CSE 4482
Computer Security Management:
Assessment and Forensics

Protection Mechanisms: Scanning and Analysis Tools

Instructor: N. Vlajic, Fall 2010

Required reading:

Management of Information Security (MIS), by Whitman & Mattord

Chapter 10, pp. 361 – 365

Recommended reading:

Principles of Information Security, by Whitman & Mattord

Chapter 7, pp. 323 - 336

Security+ Guide to Network Security Fundamentals, by Ciampa

Chapter 9, pp. 312 - 322

Learning Objectives

Upon completion of this material, you should be able to:

- List and define the major categories of scanning and analysis tools, and describe the specific tools used within each category.
- Describe, in more detail, the key techniques deployed for the purposes of port scanning, OS fingerprinting and password cracking.

Introduction

- **Why Scanners and Analysis Tools?** – **in order to secure a network, company needs to know what exactly needs securing**
  - scanners & analysis tools can find vulnerabilities in systems, and security holes in individual system components (hosts, routers, firewalls)
  - many scanners and analysis tools are developed by hacker community, and are 'freeware'
    - **same tools can also be used by network defenders to find potential vulnerabilities in the network**

Introduction (cont.)

- **Categories of Hacking Tools**

  1) Port Scanners
  2) Network Mappers
  3) Operating System Detection Tools
  4) Firewall Analysis Tools
  5) Vulnerability Scanners
  6) Packet Sniffers
  7) Wireless Sniffers
  8) Password Crackers

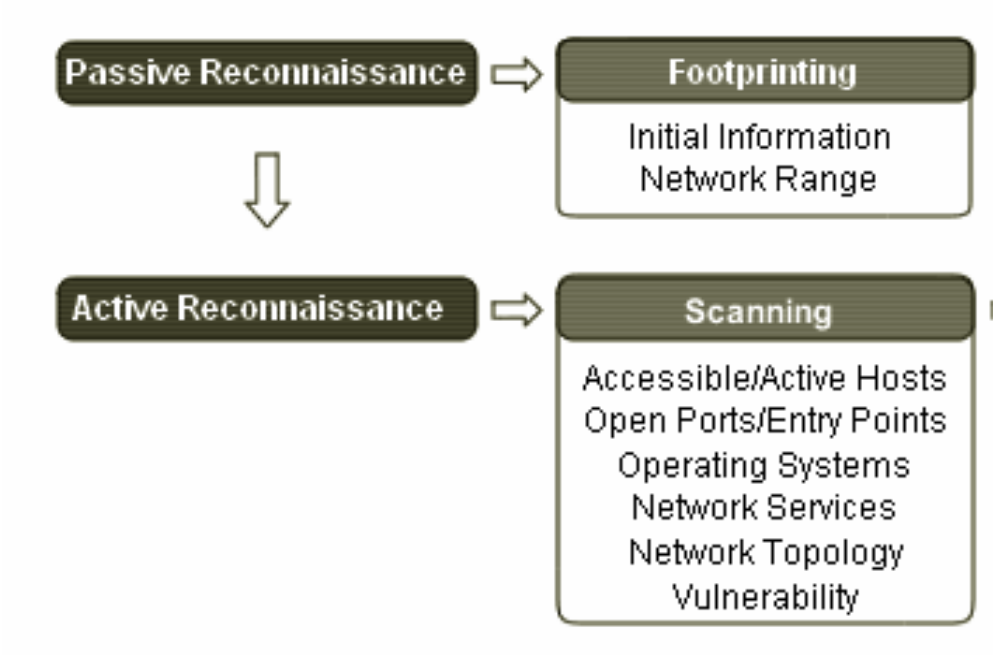
Port Scanners

- **Footprinting** – **systematic research of IP addresses owned by a target organization**
  - passive and non-intrusive process of discovering which machines are out there (part of initial attack reconnaissance)
  - Web pages are a good starting point
    - **often contain data about internal system**

examples of **'doorknob rattling'**

- **Fingerprinting (Scanning)** – **systematic scanning of a target organization's IP addresses / hosts**
  - process of discovering individual hosts as well as services running on them
  - **may include OS Fingerprinting** – determining which OS runs on a host
    - **e.g., done through detailed inspection of how a host responds to / crafts TCP packets**

Port Scanners (cont.)



Information Gathering Methodology

Port Scanners (cont.)

- **Port Scanner** – **a tool used by both attackers and defenders to identify active hosts and services on a network**
  - results of a scan on a port can be:
    - **open or accepted:** host sends reply indicating that service is available on a port
    - **closed / denied / not listening:** host sends reply indicating unavailable service on a port
    - **filtered / dropped / blocked:** there is no reply
  - popular scanners:
    - **Nmap (UNIX / Windows)** – can rapidly sweep large networks, can bypass firewalls, IDSs, ...
    - **SuperScan 4.0 (Windows) –** GUI based with additional tools in one interface, ...
    - **Advanced Port Scanner (Windows) –** small, fast, straightforward GUI, ...

Port Scanners (cont.)

Example: Advanced Port Scanner

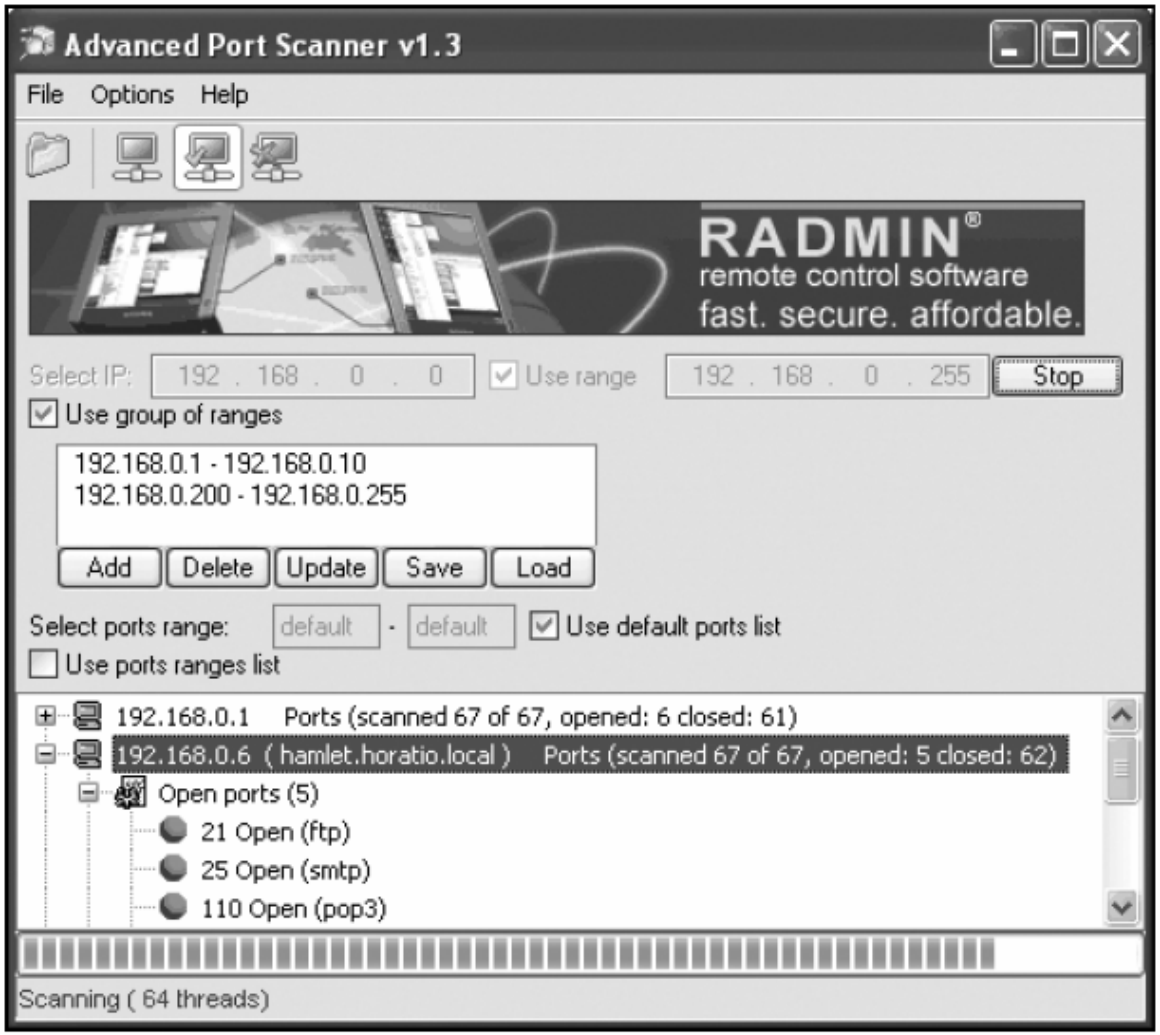


Figure 9-3 Port scanner

Port Scanners (cont.)

- **Port Scanner Techniques**

1) ICMP Ping Scan

- **not really port scanning, as ICMP is Layer 3 protocol, but useful for probing of all active hosts in a network – host scanning**
- **scanner sends a single ICMP request to a destination; an ICMP response will arrive back unless the destination is not available or ICMP protocol is filtered**
- **potentially faster than other footprinting technique – only one sent packet per machine**
- **does not provide lots of information …**

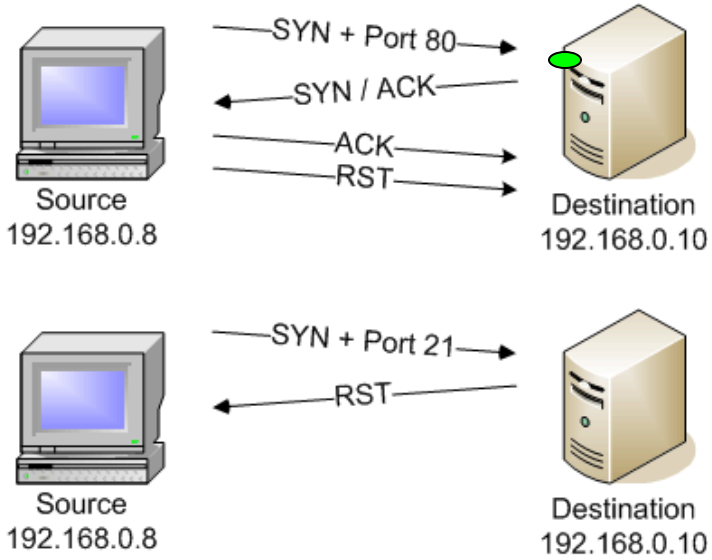
ICMP Ping scan to an open and to a closed port
http://www.networkuptime.com/nmap/page3-8.shtml

Port Scanners (cont.)

- **Port Scanning Techniques**

2) TCP connect() Scan

- **most basic form of TCP scanning**
- **OS's connect() function is used to connect to a desired port**
- **easy to implement; however, very slow and detected (loged) by most sites/firewalls**



TCP connect() scan to an open and to a closed port
http://www.networkuptime.com/nmap/page3-3.shtml

According to TCP RFC (specification):

TCP to a closed port => TCP RST arrives back
(unless RST=1)

UDP to a closed port => ICMP unreachable arrives back

Port Scanners (cont.)

- **Port Scanning Techniques**

3) TCP SYN Scan

- **'half-open' scanning**
- **instead of using OS' network function, scanner itself generates TCP-SYN packets; upon receiving a TCP-ACK, scanner immediately sends a RST to close the connection ⇒ handshake is never completed!**
- **most popular form of TCP scan since most sites do not log half-open connections - much 'quieter' than connect() scan**
- **requires programming at OS level**

TCP SYN scan to an open and to a closed port
http://www.networkuptime.com/nmap/page3-2.shtml

Port Scanners (cont.)

- Port Scanner Techniques

4) TCP FIN Scan (Stealth Scan)

- **stealth scan = scan that sends a single frame to a TCP port without any TCP TCP handshaking / additional packets**
- **TCP FIN scan sends a FIN packet; a closed port will reply with a proper RST, an open port will ignore the packet**
- **silence indicates an open port!!!**
- **UNIX vulnerable, but Microsoft is immune to this type of attack – RST sent regardless of the port state**

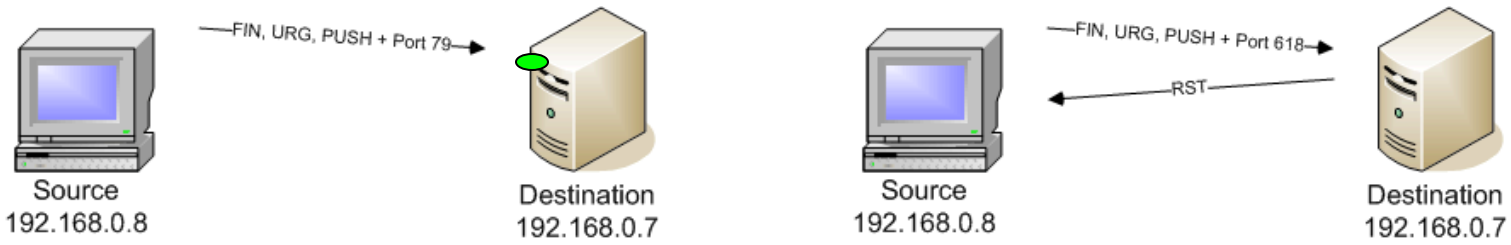
TCP FIN scan to an open and to a closed port
http://www.networkuptime.com/nmap/page3-4.shtml#3.3.1

Port Scanners (cont.)

- **Port Scanner Techniques**

5) Xmas-Tree Scan (Stealth Scan)

- **scanner sends a TCP frame with URG, PUSH and FIN flags set – Xmas tree packet (flags: 00101001, 'supper case' of FIN TCP packet)**
- **in a Xmas packet, a few flags other than RST are set to 1**
- **as in TCP FIN scan, <u>silence indicates an open port</u>!!!**



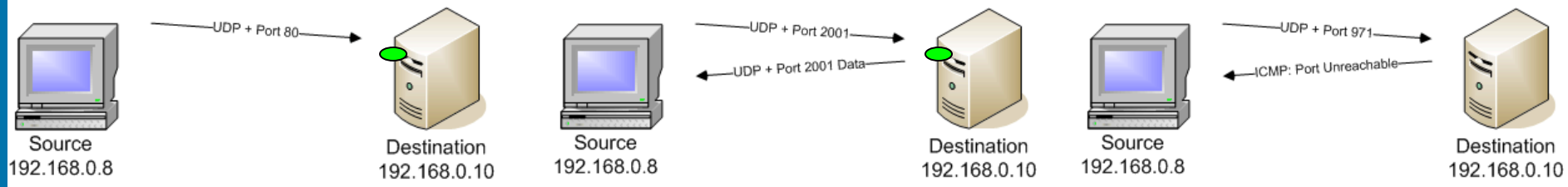
Xmas-Tree scan to an open and to a closed port
http://www.networkuptime.com/nmap/page3-5.shtml

Port Scanners (cont.)

- Port Scanner Techniques

6) UDP ICMP Scan

- previous scans find TCP ports/services; this scan looks for UDP ports/services
- scanner sends <u>empty</u> UDP datagrams – if port is listening, system sends back an error UDP message or nothing; if port is closed system sends an 'ICMP Port Unreachable'
- both UDP and ICMP are not guaranteed to arrive – **lots of false positives possible**
- also, **a rather slow scan**, as some systems limit the ICMP error message rate



UDP ICMP scan to an open and to a closed port
http://www.networkuptime.com/nmap/page3-10.shtml

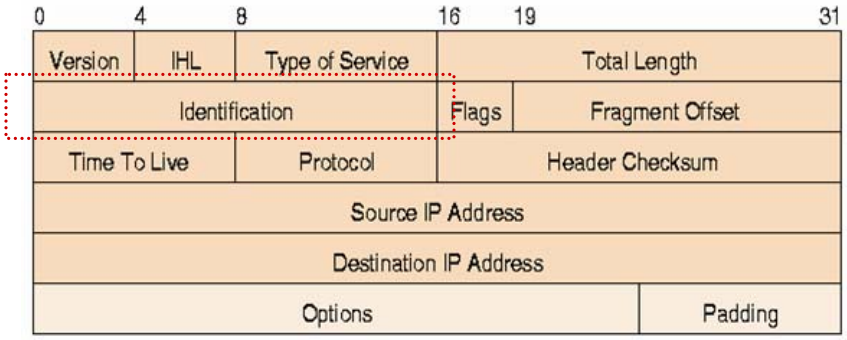
Port Scanners (cont.)

Example **Idle Scan by Nmap**

Used to:
1) hide the identity of the attacker (scanning machine); and/or
2) scan behind a firewall.

Exploits the fact that in many OSs, for every IP packet sent, **the value in packet's IP ID filed is incremented by one.**

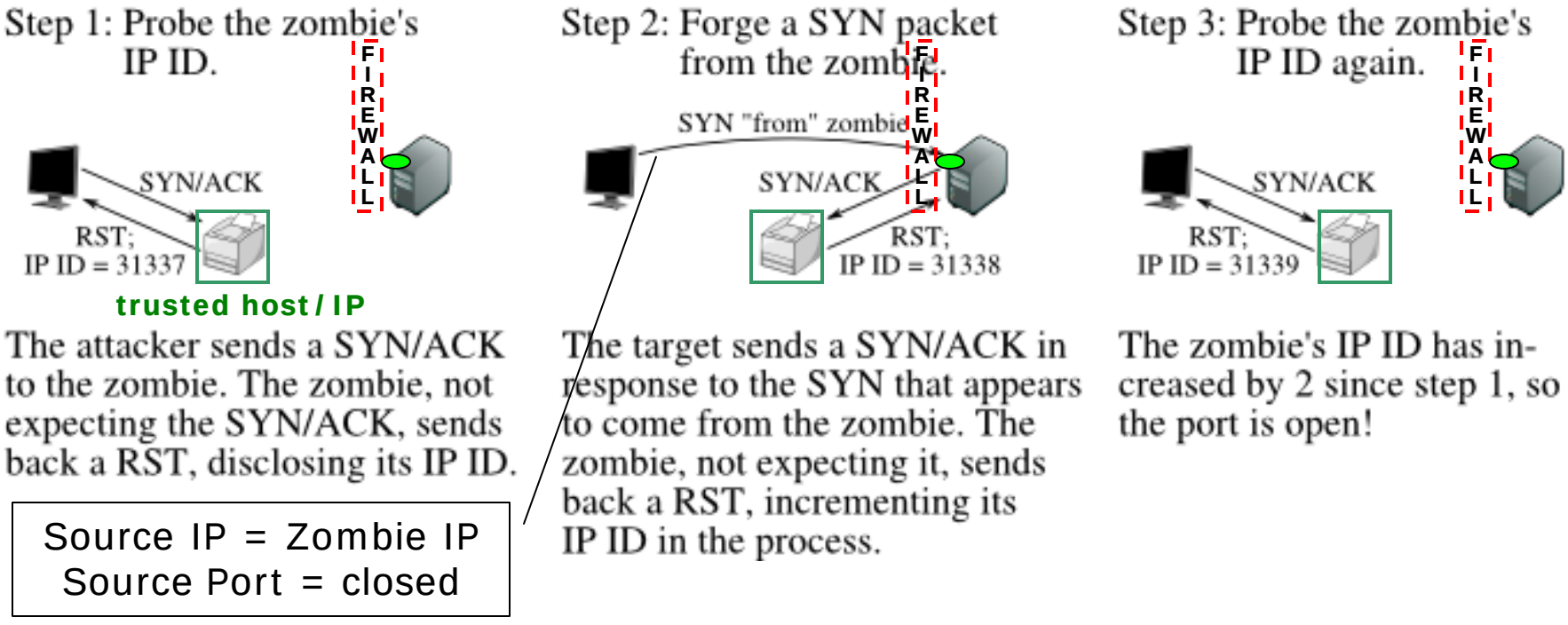
Requires the access to (communication with) at least one **zombie/dumb** host that can communicate directly with the target machine and sends/receives little traffic.



Port Scanners (cont.)

Example **Idle Scan** (cont.)

Idle scan of an open port:



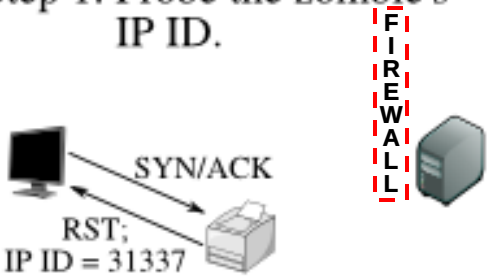
Zombie's IP ID increased by 2!

Port Scanners (cont.)

Example **Idle Scan** (cont.)

Idle scan of an closed port:

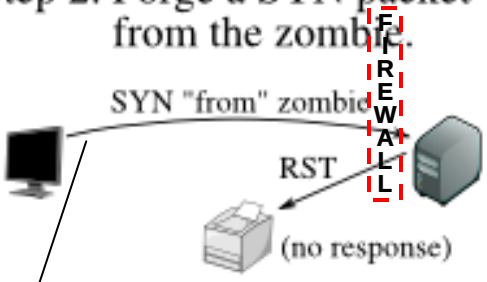
Step 1: Probe the zombie's IP ID.



The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID. This step is always the same.

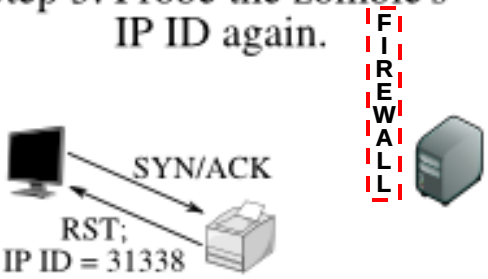
Source IP = Zombie IP
Source Port = Closed

Step 2: Forge a SYN packet from the zombie.



The target sends a RST (the port is closed) in response to the SYN that appears to come from the zombie. The zombie ignores the unsolicited RST, leaving its IP ID unchanged.

Step 3: Probe the zombie's IP ID again.



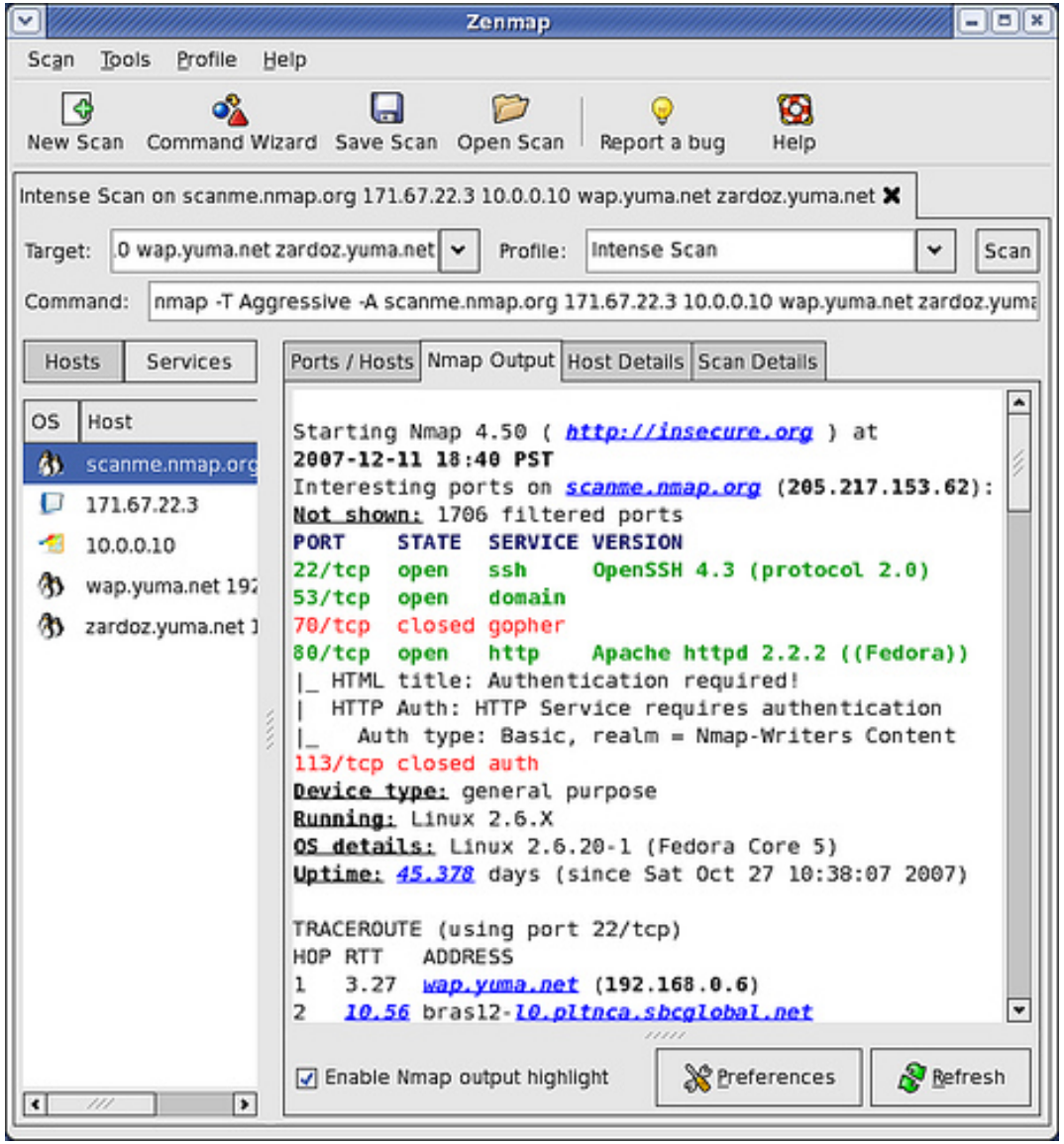
The zombie's IP ID has increased by only 1 since step 1, so the port is not open.

Zombie's IP ID increased by 1!

Port Scanners (cont.)

- **Nmap** – **Network Mapper - a security scanner written by Gordon Lyon** (aka Fyodor) **in 1997**
  - in addition to doing host and port scanning, also capable of determining:
    - OS of the target
    - name and versions of the listening services
    - type of devices
    - presence of firewall, etc.
  - runs on Windows, Linux, Solaris, Mac, etc.
  - the most respected and well-known port scanner in both **black- and white- hat** community for its efficiency, flexibility and scanning speed

Port Scanners (cont.)



Port Scanners (cont.)

- **Ethic and Legality of Port Scanning** – **fuzzy issue!**
  - scan itself is not an attack, but it is often the prelude to an attack
  - used both by hackers & defenders
  - line between scanning maliciously and scanning for administrative purposes is very vague
    - **makes creating laws regarding scanning difficult**
  - in USA and Canada the Law is not explicit, in Germany and England scanning is illegal

http://nmap.org/book/legal-issues.html

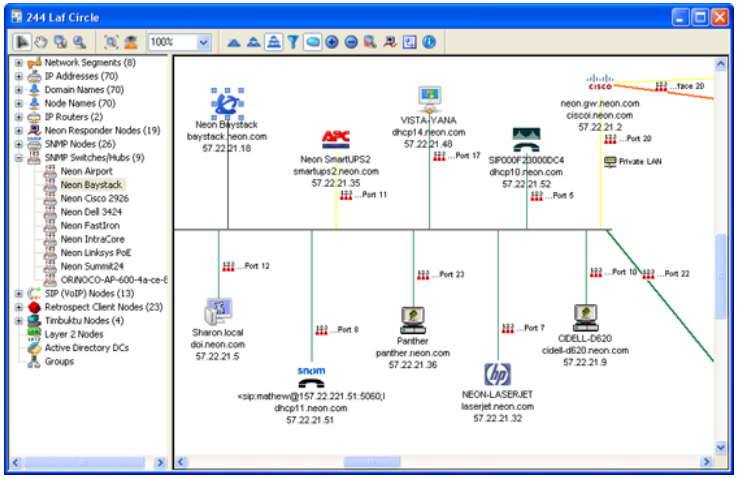
http://en.wikipedia.org/wiki/Port_scanner

- **Categories of Hacking Tools**

1) Port Scanners
2) Network Mappers
3) Operating System Detection Tools
4) Firewall Analysis Tools
5) Vulnerability Scanners
6) Packet Sniffers
7) Wireless Sniffers
8) Password Crackers

Network Mappers

- **Network Mappers** – **software tools that identify all systems connected to a network**
  - most mappers use ICMP Ping ...
  - most port scanners can be used as network mappers
  - examples of network mappers:
    - **Nmap**
    - **LanState**
    - **SolarWinds' LanSurveyor**



http://finaldownload.com/graphicsfile/screenshotimages/lansurveyor-101102.jpg

- **Categories of Hacking Tools**

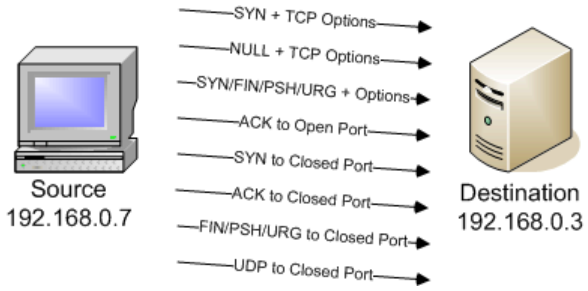
1) Port Scanners
2) Network Mappers
3) Operating System Detection Tools
4) Firewall Analysis Tools
5) Vulnerability Scanners
6) Packet Sniffers
7) Wireless Sniffers
8) Password Crackers

Operating System Detection Tools

- **OS Detection Tools** – **aka OS Fingerprinting Tools - aim to detect target host's OS**
  - knowing a host's OS is critical if one is to exploit the host's vulnerabilities (e.g. known bugs of that OS)
  - passive fingerprinting – occurs without obvious querying of host machine (e.g. obtain information through sniffing)
  - active fingerprinting – directly query host machine; replies are matched against database on known responses
  - examples of OS detection tools:
    - **Nmap**
    - **Xprobe**

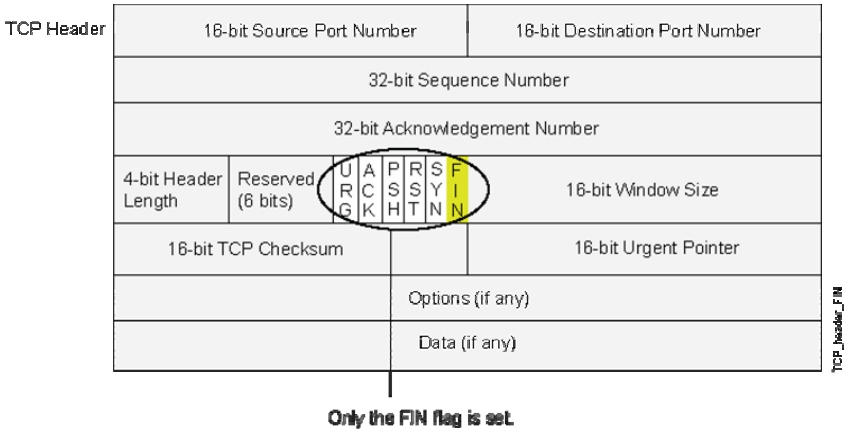
Operating System Detection Tools (cont.)

- **OS Detection Techniques in Active Fingerprinting** – **while TCP/IP stack is pretty much a fixed standard, different OS vendors interpret the standard differently**
  - (active) fingerprinting takes advantage of differences in TCP/IP implementation
    - a crafted packet is sent to a remote system to elicit a unique response from the TCP/IP stack of the underlying OS
    - the unique response is referred to as an **OS fingerprint** or **signature**
    - the attacker than carefully analyzes and and compares the fingerprint to a data-base – comprising a wide range of known OS fingerprints ...

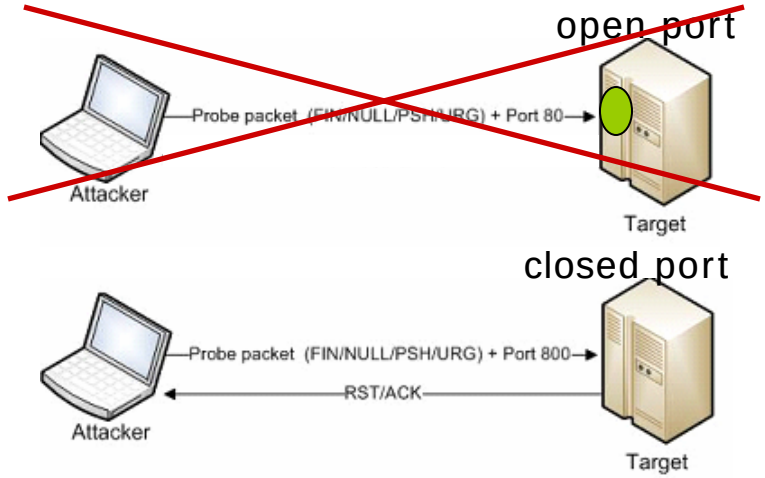


Operating System Detection Tools (cont.)

- **Nmap Fingerprint Methods**
  - **TCP FIN Probing**
    - TCP RFC requires that a system with an open port ignores (not respond to) a FIN packet if received at the start of a connect.
    - Microsoft Windows disregard this requirement and replies to the FIN packet with a RST packet



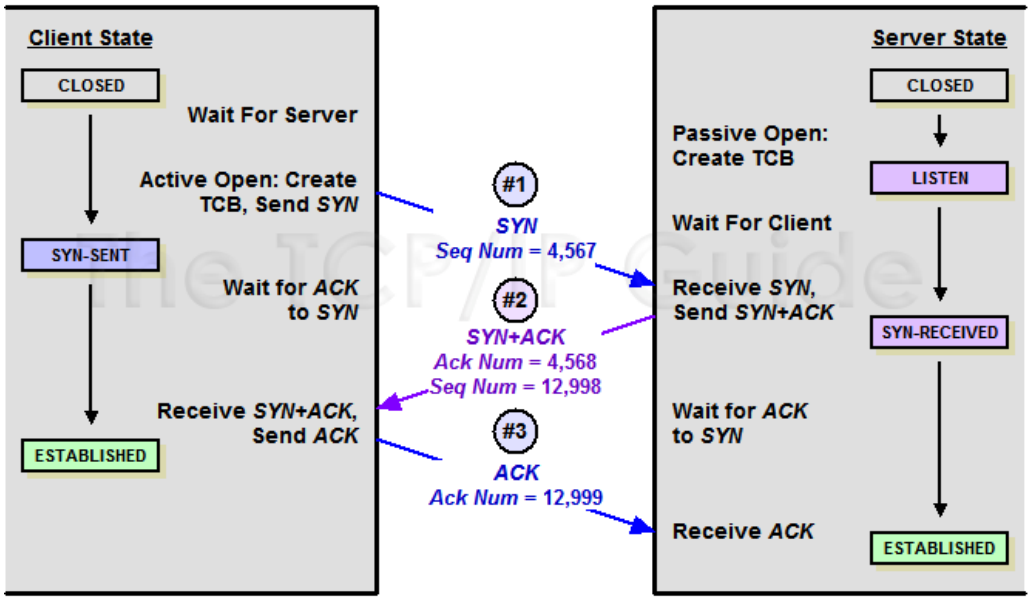
In Windows always response!



Probe a machine with a FIN packet on a port that is KNOW to be open.
If you receive a response ⇒ Windows OS!

Operating System Detection Tools (cont.)

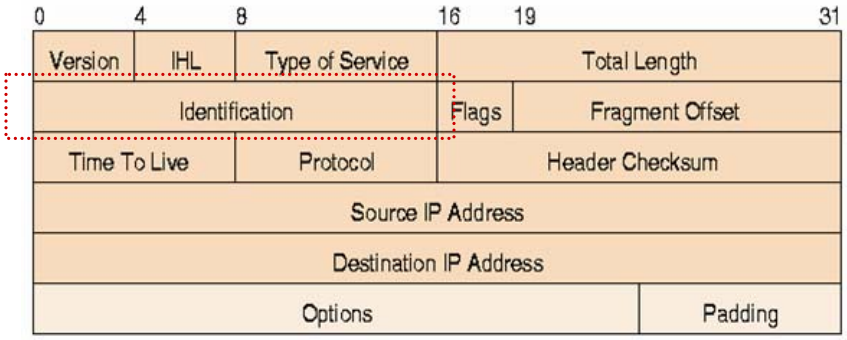
- Nmap Fingerprint Methods (cont.)
  - **TCP Initial Sequence Number (ISN)**
    - when receiving a request to establish a connection, an OS must choose an ISN to respond and continue the 3-way handshake
    - some OS choose ISN based on randomized values, while others (Windows) generate the ISN based on system's internal clock (ISN is incremented by 1 every 4 microseconds)



http://www.tcpipguide.com/free/diagrams/tcp3waysynch.png

Operating System Detection Tools (cont.)

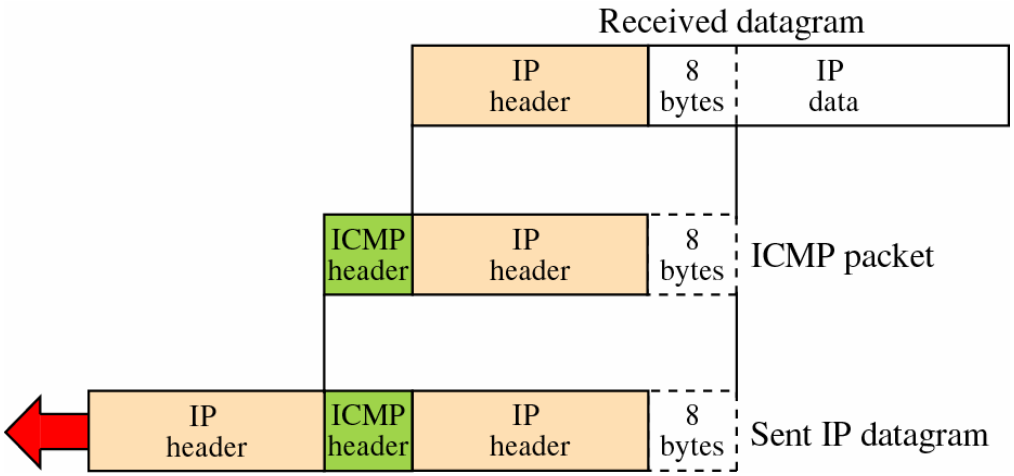
- **Nmap Fingerprint Methods (cont.)**
  - **TCP Initial Window Size**
    - some OSs are known to use a unique Window size
    - e.g. Linux 2.4 IWS=5840 bytes, Linux 2.2 IWS=32120 bytes
  - **IP ID Sampling**
    - Windows OS usually use a predictable IP ID sequence numbers, such as increasing the number by 1 or 256 for each packet
    - other OS, e.g. Linux, randomize IP ID numbers



Operating System Detection Tools (cont.)

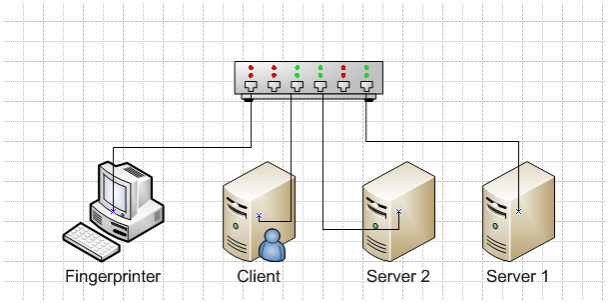
- **Nmap Fingerprint Methods (cont.)**
  - **ICMP Error Message Quoting**
    - according to ICMP RFC, OS must quote some parts of the original (ICMP) message - first 8 bytes - when generating an ICMP error message
    - Linux and Solaris include much more information than required

e.g. UDP packet to a closed port



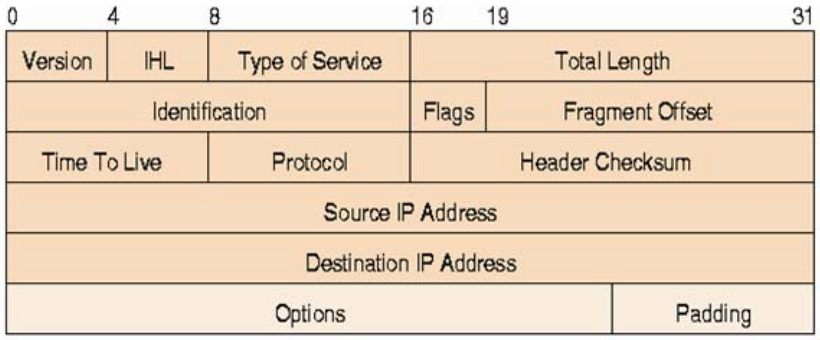
Operating System Detection Tools (cont.)

- **OS Detection Techniques in in Passive Fingerprinting** – **less intrusive way to gather information about the OS of a remote host**
  - instead of actively querying, attacker (only) sniffs remote host's packets
  - generally less precise/effective than active fingerprinting because:
    - have to accept whatever communication happens - there may not be much of it!
    - has fewer header parameters/options to work with than active fingerprinting
    - some of those parameters often get modified by firewall or proxy
  - on Nmaps 'avoided methods' list



Operating System Detection Tools (cont.)

- **Passive Fingerprint Methods**
  - **Time-to-Live (TTL) in IP packets**
    - normally **Linux** sets TTL = 64, and **Windows** TTL = 128
  - **Don't Fragment Bit in IP packet**
    - Most systems set it to 1; in **OpenBSD** set to 0
  - **Type of Service (TOS)**
    - Normally set to 0; a few OS reported using different value.
      (Generally not reliable as the TOS value is often set by application.)



Operating System Detection Tools (cont.)

Example **Idle Scan by Nmap**

Assume a sniffer/attacker has capture a packet with the following parameters:

Time-to-Live (TTL): 51
TCP Window Size: 57344
Don't Fragment Bit: 1
Type of Service (TOS): 0

How would you go about determining the host's OS?

Solution:

Use Traceroute to determine the actual number of hops between itself and the host – say you have observed 13.

Hence, original TTL = 51 + 13 = 64.

Host's OS: (likely) Linux

Operating System Detection Tools (cont.)

- **OS Detection – Counter-measures** **to reduce chances of an OS being 'fingerprinted', OS's responses to various network requests/packets must be modified**

  - **IP Personality** – a patch for Linux kernel – allows changes to TCP/IP stack
    - **IP ID field, TCP Initial Window, TCP initial Sequence Number … values can be changed**

    http://ippersonality.sourceforge.net/

  - **Morph** and **IP Scrubber** – operate in firewall manner
    - **any traffic traveling from local net. will be 'scrubbed' & any OS-related information will be removed**

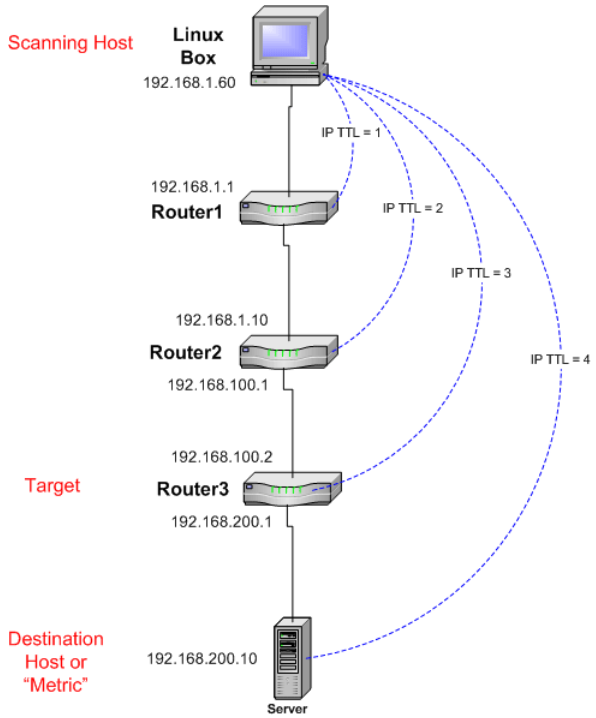
    http://www.synacklabs.net/projects/morph/

- **Categories of Hacking Tools**

1) Port Scanners
2) Network Mappers
3) Operating System Detection Tools
4) **Firewall Analysis Tools**
5) Vulnerability Scanners
6) Packet Sniffers
7) Wireless Sniffers
8) Password Crackers

Firewall Analysis Tools

- **Why Firewall Analysis?** – **help to understand / detect current set of a firewall's rules**



- **Firewalk** – detects a firewall and its respective rules, in two phases
  - **phase 1: network discovery** through a traceroute-like procedure on ANY host inside firewall's network, TTL count to the target firewall is found
  - **phase 2: firewalking / scanning** TCP/UDP probe packets with TTL of 1-hop past firewall are sent
    - (a) if firewall does NOT allow packets in - packet will be dropped & firewall either sends no message or ICMP Port Unreachable message
    - (b) if firewall DOES allow packets in, ICMP TTL Expired message is sent by the binding host

Firewall Analysis Tools

Example

We want to test whether the firewall allows traffic for host H on port P in.

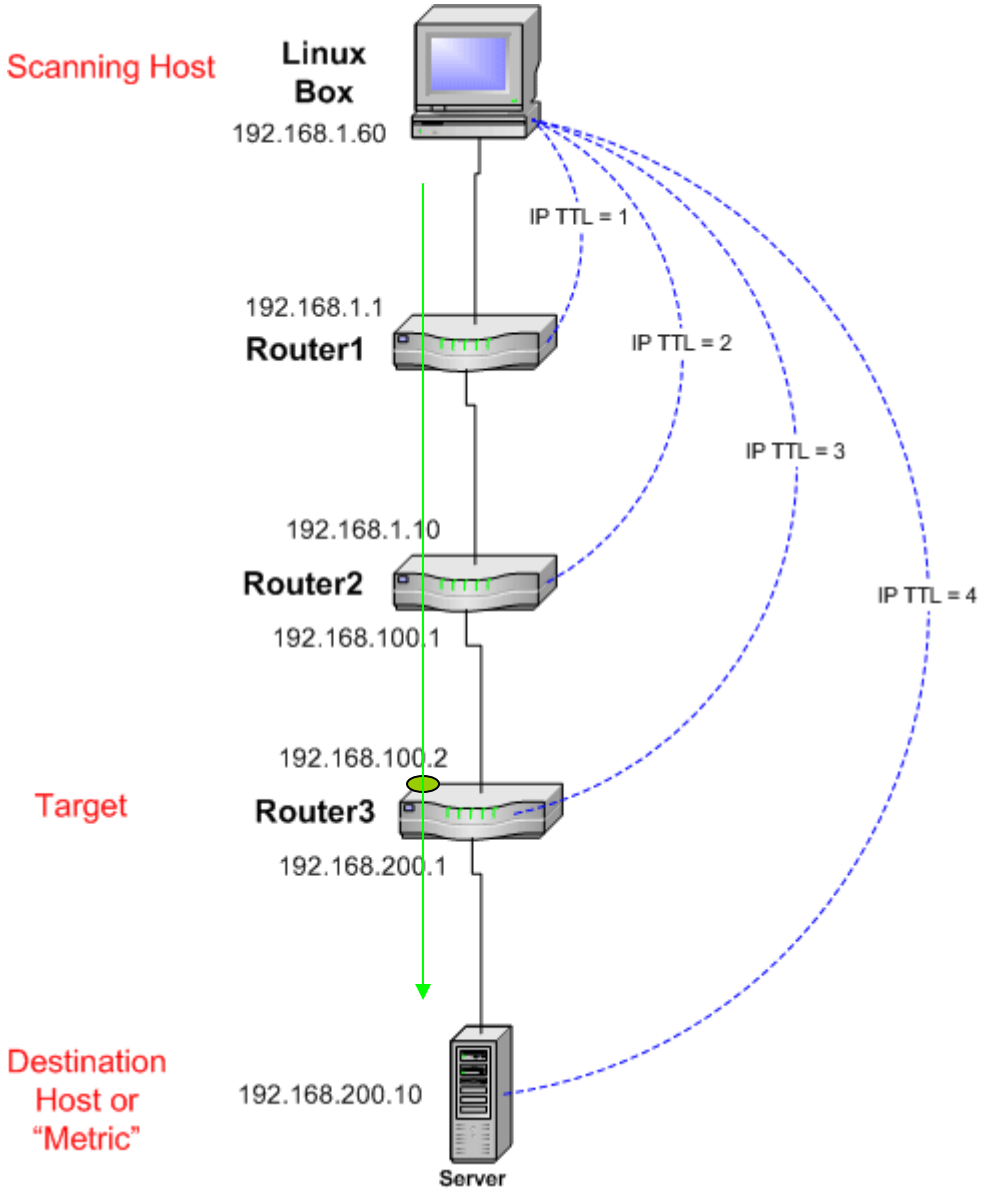
Hence, we send a UDP packet to H and port P.

No response comes back.

What can we conclude?!

a) Firewall blocks packets Intended for port P on H.

b) Firewall does not block such packet, but port P on on H is closed.



Firewall Analysis Tools (cont.)

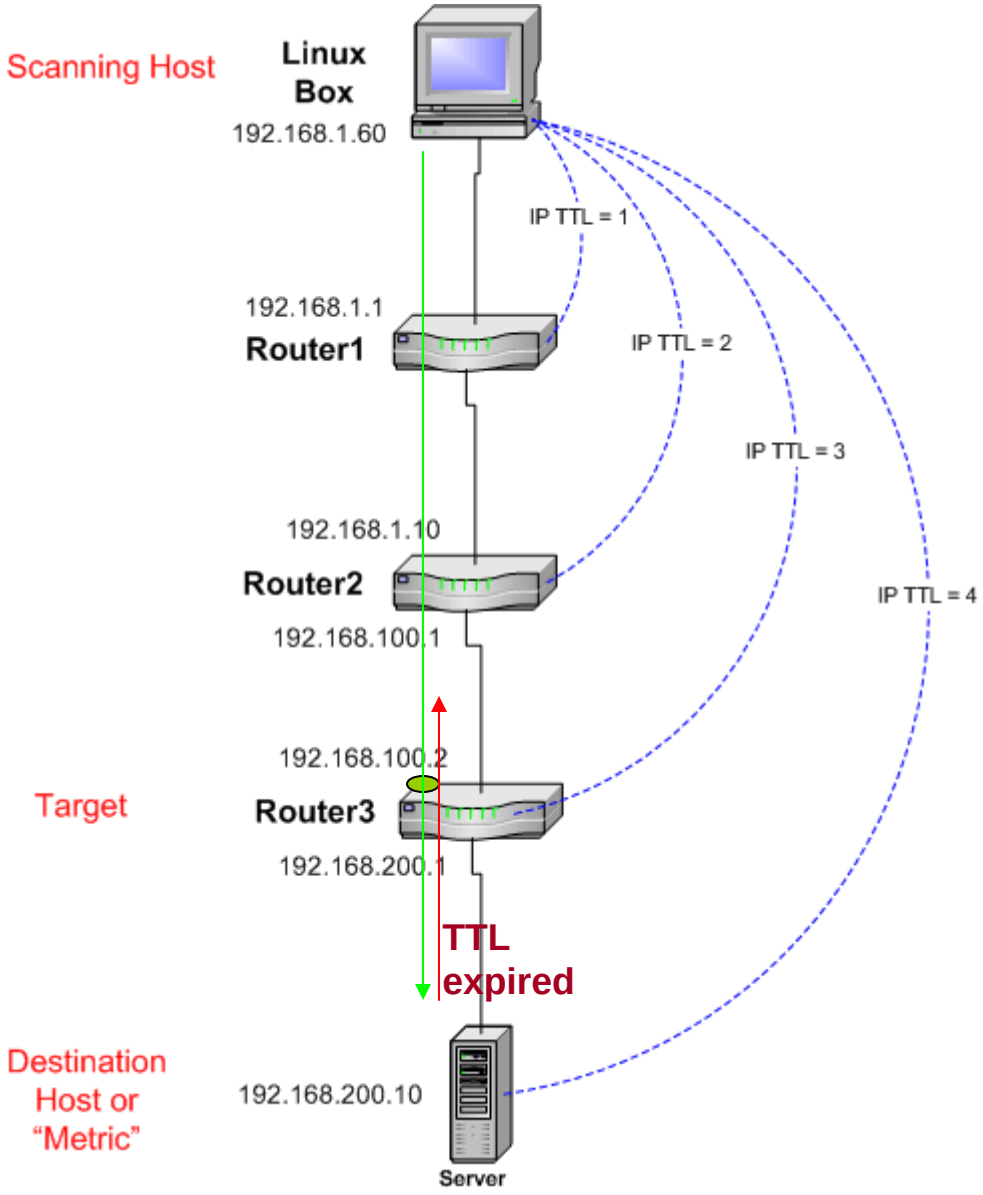
Now, assume we send the same UDP packet (to H, port P), but with $TTL = TTL_{firewall} + 1$.

Possible outcomes:

a) Firewall blocks packets intended for port P on H, thus no response arrives back.

b) Firewall lets packet(s), in but the network-layer module of H's OS sends an ICMP TTL Expired error message back.

Outcomes a) and b) are different!!!



Firewall Analysis Tools (cont.)

Example: Firewalk (http://www.e-cq.net/wp/scanning-encored.pdf)

```
[bash]# traceroute www.gotrice.com
traceroute to www.gotrice.com (203.162.168.130), 30 hops max, 38 byte
1  202.155.7.1 (202.155.7.1)  80.147 ms  74.863 ms  59.949 ms
2  202.155.7.162 (202.155.7.162)  140.144 ms  139.960 ms  139.863 ms
3  202.84.154.57 (202.84.154.57)  144.885 ms  114.808 ms  109.942 ms
4  134.159.129.174 (134.159.129.174)  380.082 ms  334.85 ms  345.20 ms
5  203.162.231.233 (203.162.231.233)  349.61 ms  344.908 ms  354.983 ms
6  203.162.95.46 (203.162.95.46)  354.922 ms  339.922 ms  349.736 ms
7  203.162.168.130 (203.162.168.130)  365.106 ms 384.931 ms 354.884 ms

[bash]# firewalk -n -p TCP -S 21,22,23,25,53,80,110,143 203.162.95.46 \
203.162.168.130

Firewalk 5.0 [gateway ACL scanner]
Firewalk state initialization completed successfully.
TCP-based scan.
Ramping phase source port: 53, destination port: 33434
Hotfoot through 203.162.95.46 using 203.162.168.130 as a metric.
Ramping Phase:
1 (TTL 1): expired [202.155.7.1]
2 (TTL 2): expired [202.155.7.162]
3 (TTL 3): expired [202.84.154.57]
4 (TTL 5): expired [134.159.129.174]
5 (TTL 5): expired [203.162.231.233]
6 (TTL 6): expired [203.162.95.46]
Binding host reached.
Scan bound at 7 hops.
Scanning Phase:
port  21: A! open (port not listen) [203.162.168.130]
port  22: A! open (port listen) [203.162.168.130]
port  23: A! open (port not listen) [203.162.168.130]
port  25: A! open (port not listen) [203.162.168.130]
port  53: A! open (port not listen) [203.162.168.130]
port  80: A! open (port listen) [203.162.168.130]
port 110: A! open (port not listen) [203.162.168.130]
port 143: A! open (port not listen) [203.162.168.130]
```

firewall / gateway router

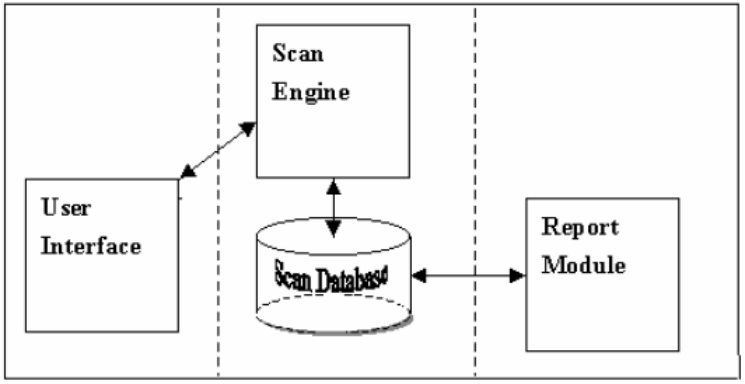
one known host on target network

- **Categories of Hacking Tools**

1) Port Scanners
2) Network Mappers
3) Operating System Detection Tools
4) Firewall Analysis Tools
5) **Vulnerability Scanners**
6) Packet Sniffers
7) Wireless Sniffers
8) Password Crackers

Vulnerability Scanners

- **Vulnerability Scanners** – **software tools that assess security vulnerabilities in networks & hosts & produce a set of scan results**
  - functionality of port scanners and more!
    - **e.g. tell you not only which ports are open, but also the name and version of software running on the port, and its vulnerabilities**
    - **components of a scanner:**



Components of Scanner

http://www.infosec.gov.hk/english/technical/files/vulnerability.pdf

Vulnerability Scanners (cont.)

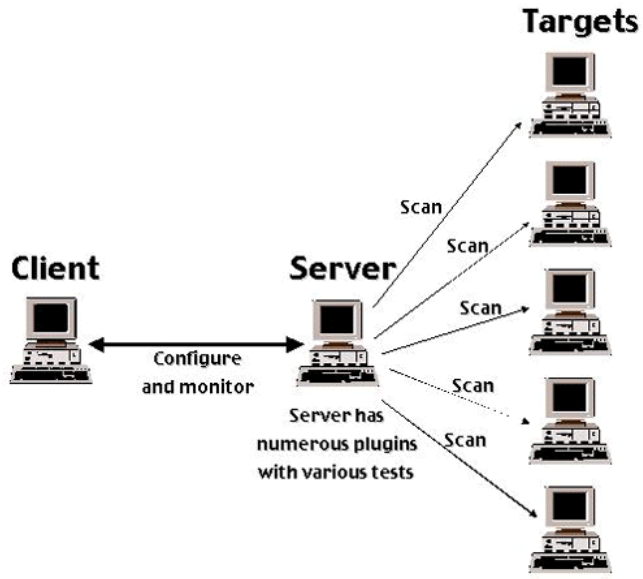
Example:

- Leader in vulnerability scanners – used by over 75,000 companies.
- Freeware!
- Can scan for vulnerabilities on either a local or a remote host.
- Comes in different flavors for UNIX, Mac and Windows.
- Able to detect:
  - open ports / available services
  - misconfigurations (e.g. missing patches)
  - default passwords
  - presence of viruses and back-door programs, etc.

Vulnerability Scanners (cont.)

Example: Nessus (cont.)

- Employs client-server architecture:
  - Nessus server includes a vulnerability database & a scanning engine.
  - Nessus client includes a user config. tool and a report-gener. tool.
  - Client & server can run on same or different machines (e.g. in case of a slow link).



- **Categories of Hacking Tools**

1) Port Scanners
2) Network Mappers
3) Operating System Detection Tools
4) Firewall Analysis Tools
5) Vulnerability Scanners
6) **Packet Sniffers**
7) Wireless Sniffers
8) Password Crackers

Packet Sniffers

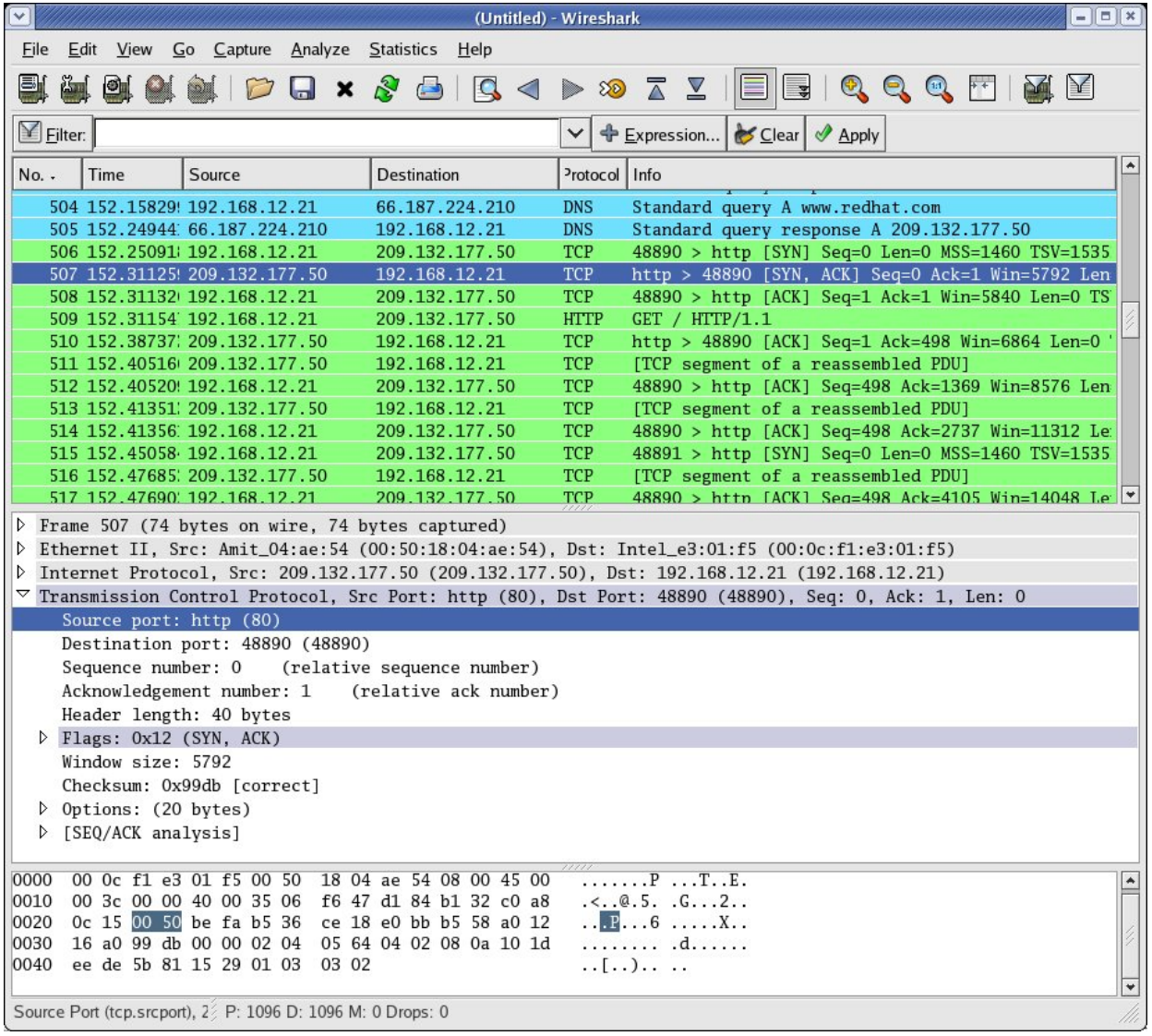
- **Packet Sniffers** – **aka network protocol analyzers – collect copies of packets from the network and decode their content**
  - common use:
    - **troubleshooting – e.g. diagnose protocol configuration mistakes**
    - **network traffic characterization – obtain a picture of type and make of network traffic to fine-tune / manage bandwidth**
    - **security analysis – e.g. detect DoS attacks by observing a large number of specific-type packets**
  - to be able to 'sniff' all LAN packets packet sniffer should be put into promiscuous mode

Packet Sniffers (cont.)

- **Packet Sniffers (cont.)**
  - warning! – simply tapping into an Internet connection constitutes a violation of the wiretapping act
  - to use a packet sniffer legally, one must:
    - **be under direct authorization of the owners of the network**
    - **have knowledge and consent of the content creators**

Packet Sniffers (cont.)

Example: Wireshark



- **Categories of Hacking Tools**

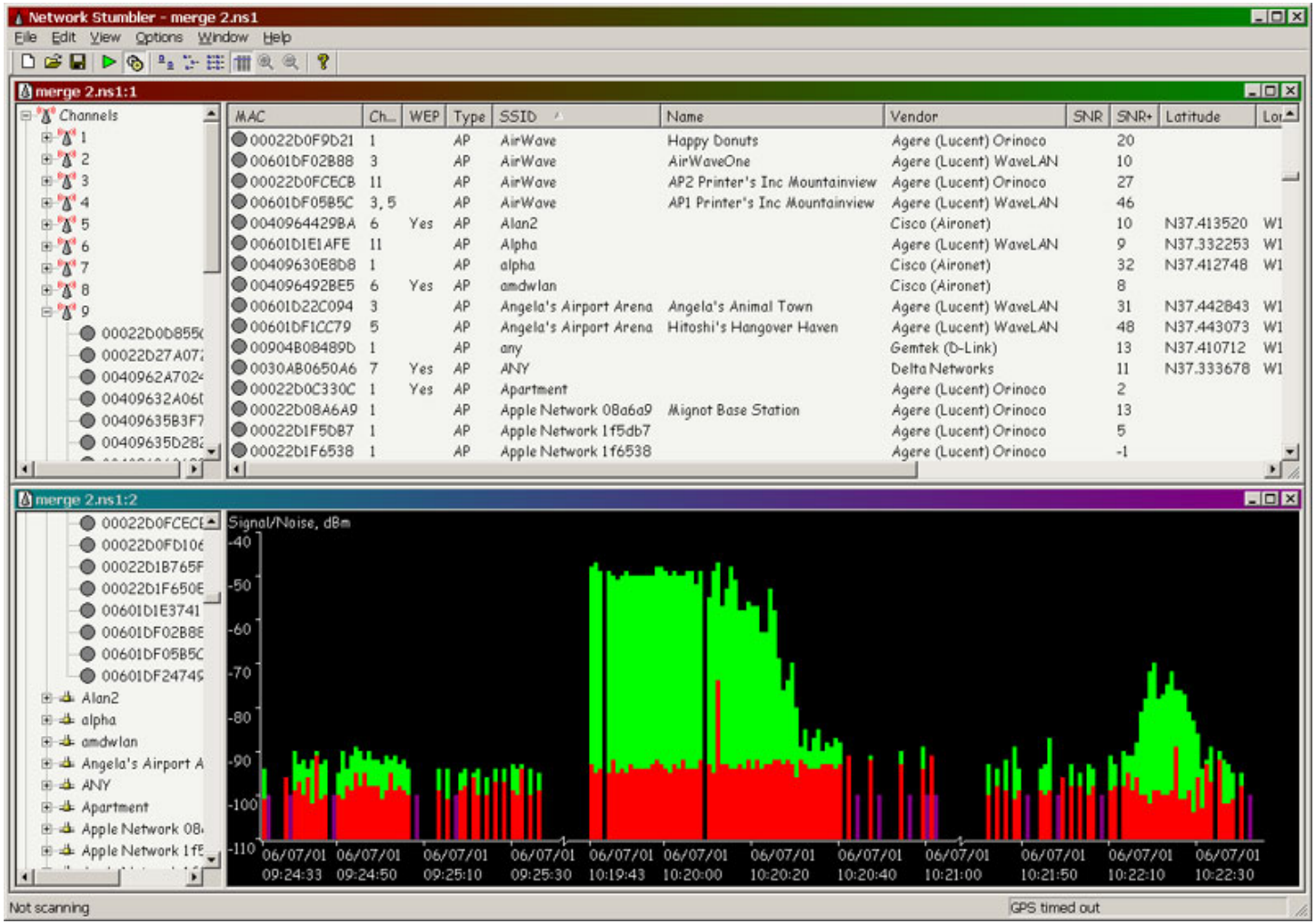
1) Port Scanners
2) Network Mappers
3) Operating System Detection Tools
4) Firewall Analysis Tools
5) Vulnerability Scanners
6) Packet Sniffers
7) **Wireless Sniffers**
8) Password Crackers

Wireless Sniffers

- **Wireless Sniffers** – **software / hardware capable of capturing & decoding packets as they pass over airwaves**
  - wireless sniffing is much easier than wired sniffing
    - **wireless medium is broadcast medium: everybody sees everything**
    - **in a wired network, the attacker must find a way to install a sniffer on a host or in a target subnet**
  - detection of wireless sniffing is extremely difficult – leaves no traceable evidence
  - name typically refers to WiFi (IEEE 802.11) sniffers

Wireless Sniffers (cont.)

Example: NetStumbler



- **Categories of Hacking Tools**

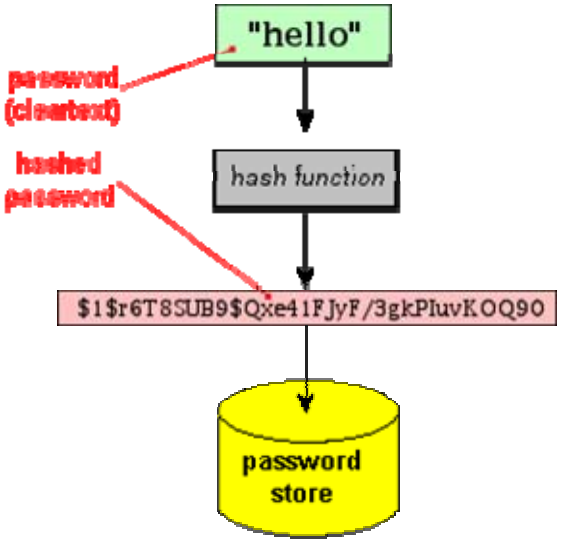
1) Port Scanners
2) Network Mappers
3) Operating System Detection Tools
4) Firewall Analysis Tools
5) Vulnerability Scanners
6) Packet Sniffers
7) Wireless Sniffers
8) Password Crackers

Password Crackers

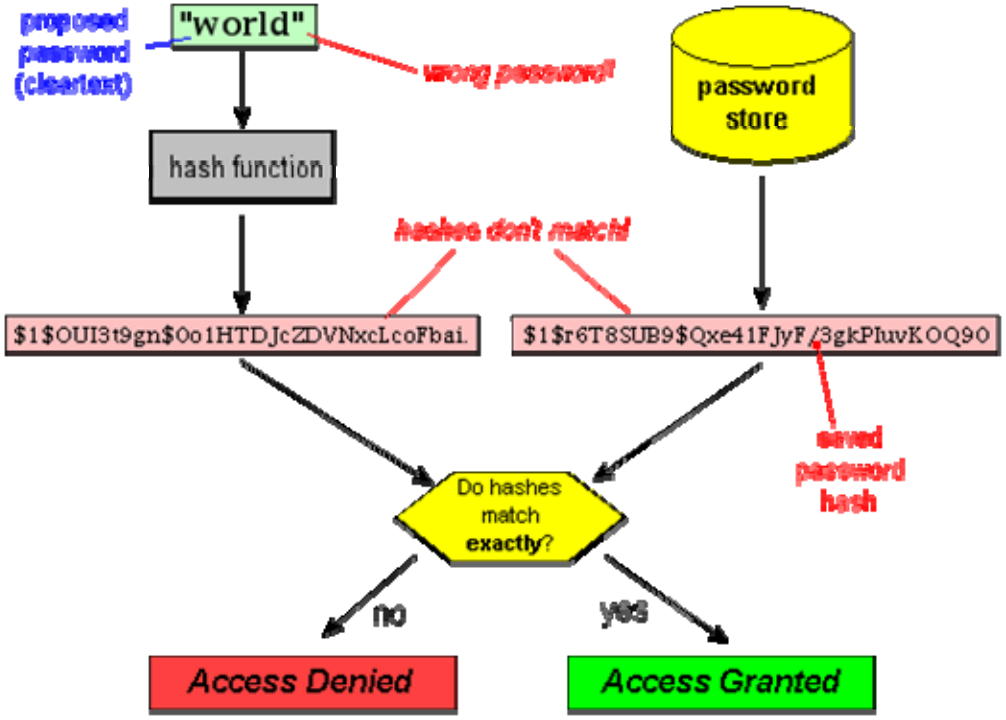
- **Password** – **a critical** (sometimes only) **defense against intruders!**
  - in most systems, passwords are stored in a protected (hash) form ⇒ snooper that gains internal access to system cannot easily retrieve/steal passwords
    - **every time a user logs in, password handling software runs the hash algorithm**
    - **if (new hash = stored hash), access is granted**
- **Password Management in Windows** – **password hashes are stored in Security Account Manager (SAM) file**
  - stored in C:\Windows\System32\config directory - cannot be accessed while computer is running (file used by OS)

Password Crackers (cont.)

Example: Password hashes



Storing hash instead of password

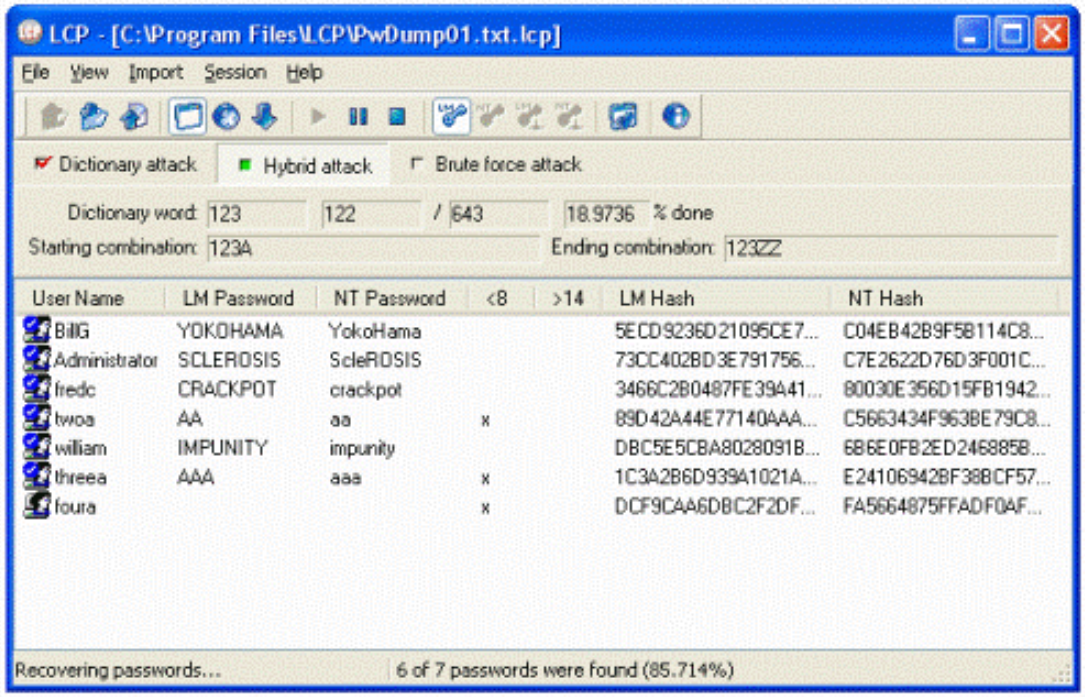


Testing a password against stored hash

http://unixwiz.net/techtips/iguide-crypto-hashes.html

Password Crackers (cont.)

- **Accessing SAM File in Windows** – **requires additional software**
  - install and run **LCP**, **pwdump**, or **FGDUMP** with Administrator privileges



http://blog.insecure.in/?tag=windows

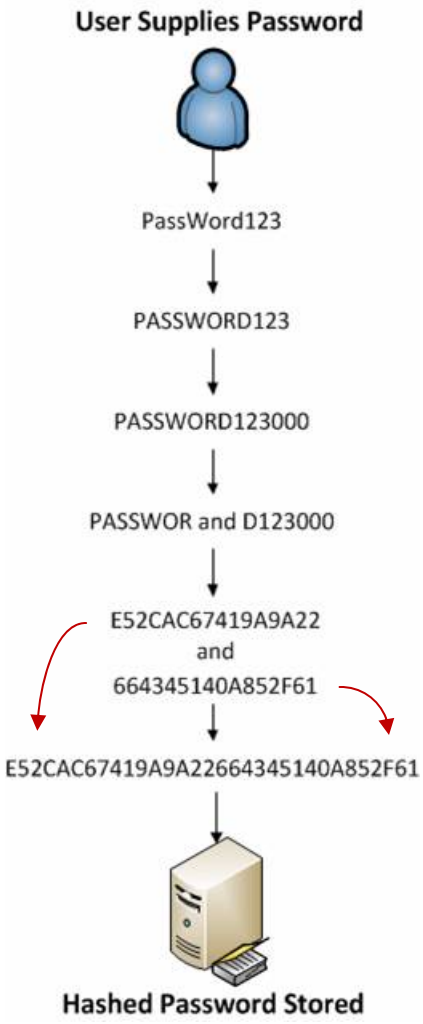
Password Crackers (cont.)

- **Password Hashing in Windows** – **Windows-based computers utilize two hashing methods**
  - LAN Manager (LM)
    - **used in earlier versions – up until Windows 2000, XP, Vista, and 7**
  - NT LAN Manager (NTLM)
    - **much stronger and harder to crack than LM hash**
    - **used in Windows 2000, XP, Vista, and 7**
    - **Windows 2000 and XP are also backward compatible – hash with both, to be able operate with older clients/servers***

      * feature that should be disabled if not necessary

Password Crackers (cont.)

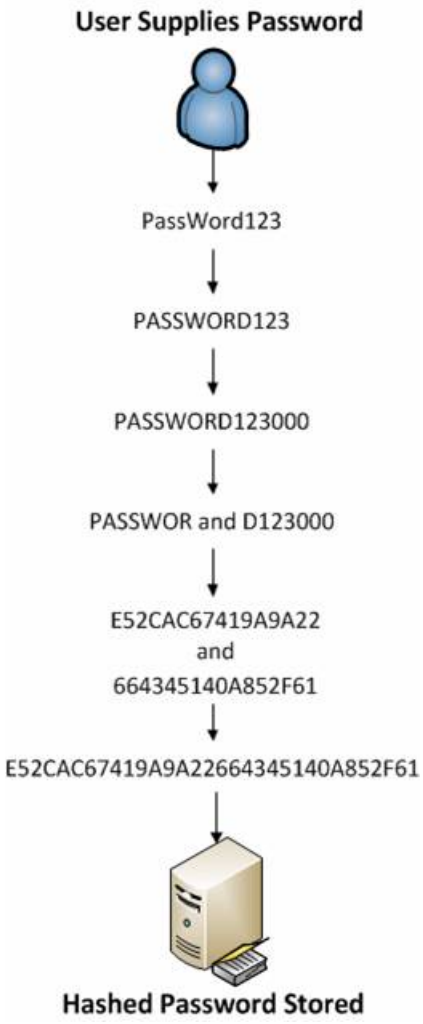
- **LM Password Hashing** – **not really a hash, but a a cryptographic value**



1) user password is converted to all uppercase
2) password has null characters added to it until it equals 14 characters
3) new password is split into two 7 character halves
4) two 7-byte halves are used to create two 64-bit (8-byte) long DES encryption keys, by inserting a null bit after every seven bits
5) each key is used to DES-encrypt the constant ASCII string "KGS!@#$%", resulting in two 16-byte long ciphertext values
6) finally, two 16 byte hashes are concatenated to form the 32-byte long hash

Password Crackers (cont.)

- **LM Password Hashes (cont.)**



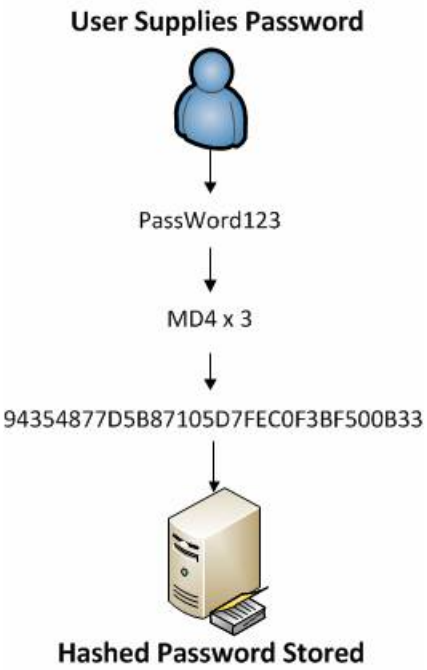
- drawbacks:

1) case insensitive – significantly reduces character set that attacker must use (n - from 95 down to 69)
2) 14-character long passwords split into two 7-character long halves – search space dramatically reduced (from n^{14} to $2*n^{7}$)

   total reduction in search space: from 95^{14} to $2*69^{7}$
3) DES encryption algorithm not considered safe anymore

Password Crackers (cont.)

- **NTLM Hashing** – **much simpler in terms of OS operations than LM**



  - applies MD4 hash algorithm 3 times
  - advantages: much 'stronger' than LN
    - **allows for distinction between upper and lower case**
    - **does not split password into smaller, easier to crack, chunks**
  - disadvantages: does not use 'salting' like in UNIX and Linux
    - **salt – random combination of 0 & 1 added to a password**
    - **every bit of salt doubles password-cracking demands on storage and computation**

Password Crackers (cont.)

- **Password Cracking** – **process of recovering passwords from data that has been stored in or transmitted by a computer**
  - brute force attack
    - **try every password** (a random combination of characters) **in real time**
    - **very slow! E.g. 8 character password of 76 possible characters = 1.1×10^{15} possibilities**
    - **2 to 3 passwords a second ⇒ 5,878,324 years to guess a password**
  - dictionary attack
    - **faster than brute force, as it uses smaller** (more likely) **search space**
    - **instead of trying every password, only common dictionary words are used**
    - **still might take long time …**

Password Crackers (cont.)

Example: Dictionary attack

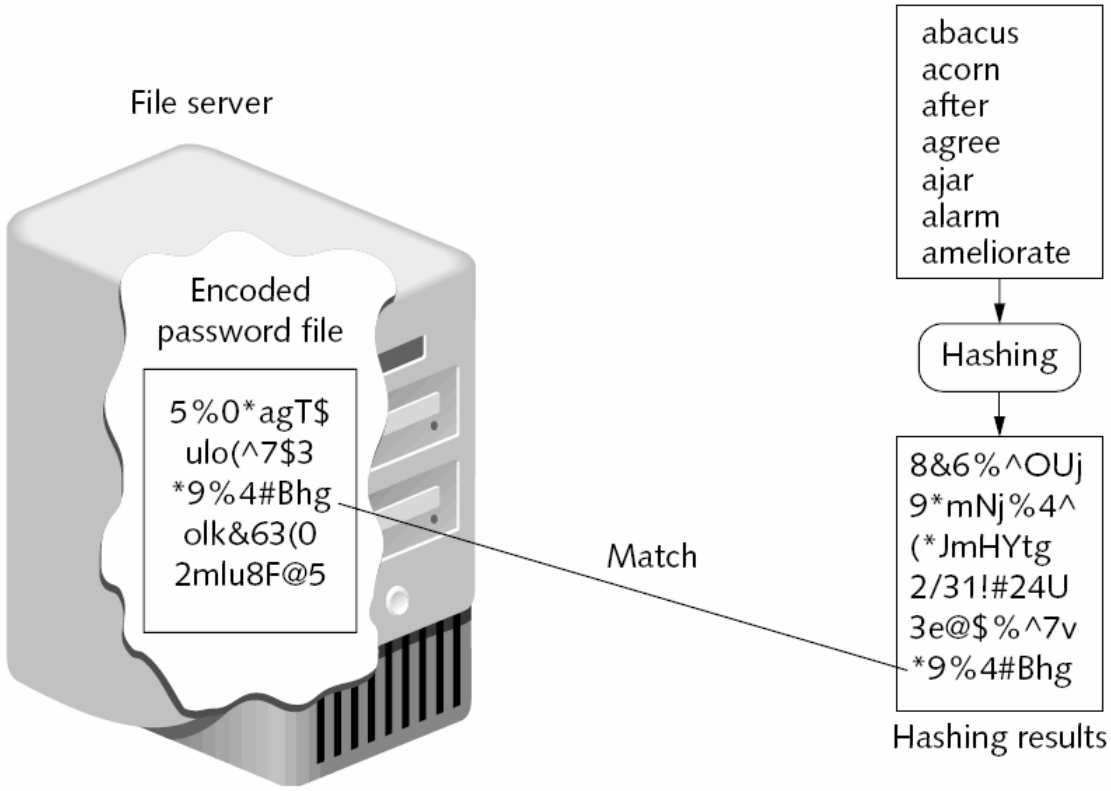


Figure 7-7 Dictionary attack

Password Crackers (cont.)

- **Password Cracking (cont.)**
  - pre-computed dictionary attack
    - **achieves a time-space tradeoff by pre-computing a list of hashes of dictionary words**
    - **pre-computed hashes are compared against those in a stolen password file**
    - **rainbow tables – effective-to-use pre-generated sets/lists of hashes**

Password Characteristics	Example	Maximum time to break using brute force	Maximum time to break using rainbow tables
8-digit password of all letters	abcdefgh	1.6 days	28 minutes
9-digit password of letters and numbers (mixed case)	AbC4E8Gh	378 years	28 minutes
10-digit password of letters and numbers (mixed case)	Ab4C7EfGh2	23,481 years	28 minutes
14-digit password of letters, numbers, and symbols	1A2*3&def456G$	6.09e + 12 years	28 minutes

Table 7-5 Times to break a hash

Password Crackers (cont.)

Example: CAIN & ABLE