CHAPTER 1

Web Application (In)security

There is no doubt that web application security is a current and very newsworthy subject. For all concerned, the stakes are high: for businesses that derive increasing revenue from Internet commerce, for users who trust web applications with sensitive information, and for criminals who can make big money by stealing payment details or compromising bank accounts. Reputation plays a critical role: few people want to do business with an insecure web site, and so few organizations want to disclose details about their own security vulnerabilities or breaches. Hence, it is not trivial to obtain reliable information about the state of web application security today.

This chapter takes a brief look at how web applications have evolved and the many benefits they provide. We present some metrics about vulnerabilities in current web applications, drawn from the authors' direct experience, demonstrating that the majority of applications are far from secure. We describe the core security problem facing web applications — that users can supply arbitrary input — and the various factors that contribute to their weak security posture. Finally, we describe the latest trends in web application security and the ways in which these may be expected to develop in the near future.

The Evolution of Web Applications

In the early days of the Internet, the World Wide Web consisted only of web *sites*. These were essentially information repositories containing static documents, and web browsers were invented as a means of retrieving and displaying those documents, as shown in Figure 1-1. The flow of interesting information was one-way, from server to browser. Most sites did not authenticate users, because there was no need to — each user was treated in the same way and presented with the same information. Any security threats arising from hosting a web site related largely to vulnerabilities in web server software (of which there were many). If an attacker compromised a web server, he would not normally gain access to any sensitive information, because the information held on the server was already open to public view. Rather, an attacker would typically modify the files on the server to deface the web site's contents, or use the server's storage and bandwidth to distribute "warez."

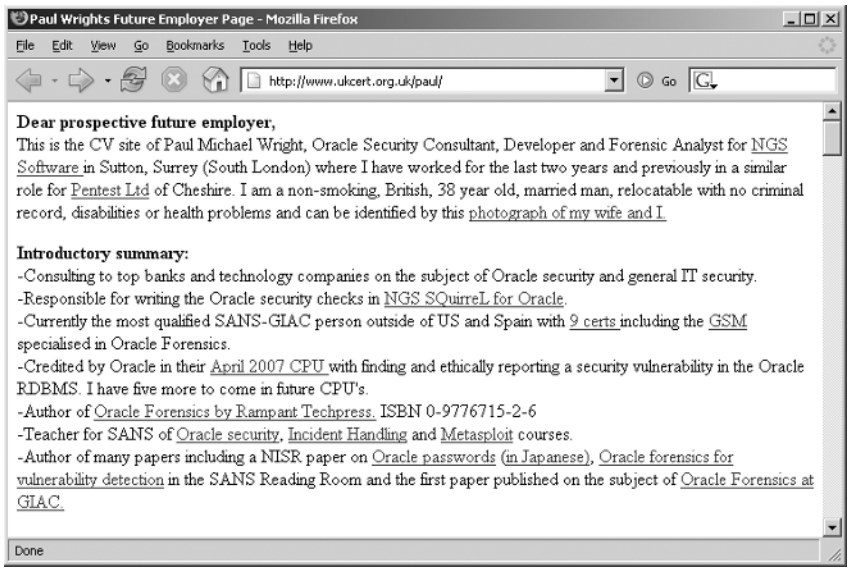

Figure 1-1: A traditional web site containing static information

Today, the World Wide Web is almost unrecognizable from its earlier form. The majority of sites on the web are in fact applications (see Figure 1-2). They are highly functional, and rely upon two-way flow of information between the server and browser. They support registration and login, financial transactions, search, and the authoring of content by users. The content presented to users is generated dynamically on the fly, and is often tailored to each specific user. Much of the information processed is private and highly sensitive. Security is

therefore a big issue: no one wants to use a web application if they believe their information will be disclosed to unauthorized parties.

Web applications bring with them new and significant security threats. Each application is different and may contain unique vulnerabilities. Most applications are developed in-house, and many by developers who have little understanding of the security problems that may arise in the code they are producing. To deliver their core functionality, web applications normally require connectivity to internal computer systems that contain highly sensitive data and are able to perform powerful business functions. Ten years ago, if you wanted to make a funds transfer, you visited your bank and someone performed it for you; today, you can visit their web application and perform it yourself. An attacker who compromises a web application may be able to steal personal information, carry out financial fraud, and perform malicious actions against other users.

Figure 1-2 A typical web application

Common Web Application Functions

Web applications have been created to perform practically every useful function one could possibly implement online. Examples of web application functions that have risen to prominence in recent years include:

- Shopping (Amazon)
- Social networking (MySpace)

- Banking (Citibank)
- Web search (Google)
- Auctions (eBay)
- Gambling (Betfair)
- Web logs (Blogger)
- Web mail (Hotmail)
- Interactive information (Wikipedia)

In addition to the public Internet, web applications have been widely adopted inside organizations to perform key business functions, including accessing HR services and managing company resources. They are also frequently used to provide an administrative interface to hardware devices such as printers, and other software such as web servers and intrusion detection systems.

Numerous applications that predated the rise of web applications have been migrated to this technology. Business applications like enterprise resource planning (ERP) software, which were previously accessed using a proprietary thick-client application, can now be accessed using a web browser. Software services such as email, which originally required a separate email client, can now be accessed via web interfaces like Outlook Web Access. This trend is continuing as traditional desktop office applications such as word processors and spreadsheets are migrated to web applications, through services like Google Apps and Microsoft Office Live.

The time is fast approaching when the only client software that most computer users will need is a web browser. A hugely diverse range of functions will have been implemented using a shared set of protocols and technologies, and in so doing will have inherited a distinctive range of common security vulnerabilities.

Benefits of Web Applications

It is not difficult to see why web applications have enjoyed such a dramatic rise to prominence. Several technical factors have worked alongside the obvious commercial incentives to drive the revolution that has occurred in the way we use the Internet:

- HTTP, the core communications protocol used to access the World Wide Web, is lightweight and connectionless. This provides resilience in the event of communication errors and avoids the need for the server to hold open a network connection to every user as was the case in many

legacy client-server applications. HTTP can also be proxied and tunneled over other protocols, allowing for secure communication in any network configuration.

- Every web user already has a browser installed on their computer. Web applications deploy their user interface dynamically to the browser, avoiding the need to distribute and manage separate client software, as was the case with pre-web applications. Changes to the interface only need to be implemented once, on the server, and take effect immediately.
- Today's browsers are highly functional, enabling rich and satisfying user interfaces to be built. Web interfaces use standard navigational and input controls that are immediately familiar to users, avoiding the need to learn how each individual application functions. Client-side scripting enables applications to push part of their processing to the client side, and browsers' capabilities can be extended in arbitrary ways using thick-client components where necessary.
- The core technologies and languages used to develop web applications are relatively simple. A wide range of platforms and development tools are available to facilitate the development of powerful applications by relative beginners, and a large quantity of open source code and other resources is available for incorporation into custom-built applications.

Web Application Security

As with any new class of technology, web applications have brought with them a new range of security vulnerabilities. The set of most commonly encountered defects has evolved somewhat over time. New attacks have been conceived that were not considered when existing applications were developed. Some problems have become less prevalent as awareness of them has increased. New technologies have been developed that have introduced new possibilities for exploitation. Some categories of flaws have largely gone away as the result of changes made to web browser software.

Throughout this evolution, compromises of prominent web applications have remained in the news, and there is no sense that a corner has been turned and that these security problems are on the wane. Arguably, web application security is today the most significant battleground between attackers and those with computer resources and data to defend, and it is likely to remain so for the foreseeable future.

"This Site Is Secure"

There is a widespread awareness that security is an "issue" for web applications. Consult the FAQ page of a typical application, and you will be reassured that it is in fact secure. For example:

> *This site is absolutely secure. It has been designed to use 128-bit Secure Socket Layer (SSL) technology to prevent unauthorized users from viewing any of your information. You may use this site with peace of mind that your data is safe with us.*

In virtually every case, web applications state that they are secure because they use SSL. Users are often urged to verify the site's certificate, admire the advanced cryptographic protocols in use, and on this basis, trust it with their personal information.

In fact, the majority of web applications are insecure, and in ways that have nothing to do with SSL. The authors of this book have tested hundreds of web applications in recent years. Figure 1-3 shows the proportions of those applications tested during 2006 and 2007 that were found to be affected by some common categories of vulnerability. These are explained briefly below:

- **Broken authentication (67%)** — This category of vulnerability encompasses various defects within the application's login mechanism, which may enable an attacker to guess weak passwords, launch a brute-force attack, or bypass the login altogether.
- **Broken access controls (78%)** — This involves cases where the application fails to properly protect access to its data and functionality, potentially enabling an attacker to view other users' sensitive data held on the server, or carry out privileged actions.
- **SQL injection (36%)** — This vulnerability enables an attacker to submit crafted input to interfere with the application's interaction with back-end databases. An attacker may be able to retrieve arbitrary data from the application, interfere with its logic, or execute commands on the database server itself.
- **Cross-site scripting (91%)** — This vulnerability enables an attacker to target other users of the application, potentially gaining access to their data, performing unauthorized actions on their behalf, or carrying out other attacks against them.
- **Information leakage (81%)** — This involves cases where an application divulges sensitive information that is of use to an attacker in developing an assault against the application, through defective error handling or other behavior.

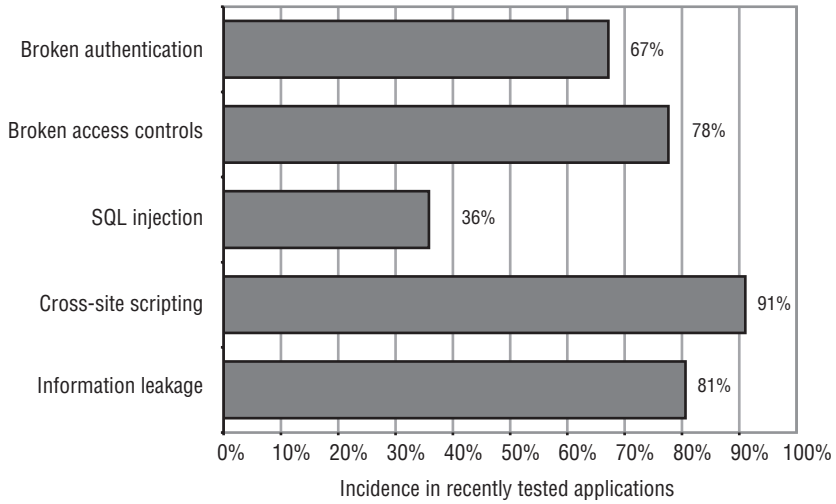

Figure 1-3 The incidence of some common web application vulnerabilities in applications recently tested by the authors (based on a sample of more than 100)

SSL is an excellent technology that protects the confidentiality and integrity of data in transit between the user's browser and the web server. It helps to defend against eavesdroppers, and it can provide assurance to the user of the identity of the web server they are dealing with. But it does not stop attacks that directly target the server or client components of an application, as most successful attacks do. Specifically, it does not prevent any of the vulnerabilities listed previously, or many others that can render an application critically exposed to attack. Regardless of whether or not they use SSL, most web applications still contain security flaws.

NOTE **Although SSL has nothing to do with the majority of web application vulnerabilities, do not infer that it is unnecessary to an application's security. Properly used, SSL provides an effective defense against several important attacks. An occasional mistake by developers is to eschew industry-standard cryptography in favor of a home-grown solution, which as a rule is more expensive and less effective. Consider the following (actual) FAQ answer, which rings even louder alarm bells than the orthodox wisdom described previously:**

This site is secure. For your safety (and our peace of mind) we do not use "standard" security procedures such as SSL but proprietary protocols which we won't disclose in detail here but permit immediate transfer of any data you submit to a completely secure location. In other words the data never stays on a server "floating in cyberspace," which allows us to keep potential malfeasants in the dark.

The Core Security Problem: Users Can Submit Arbitrary Input

As with most distributed applications, web applications face a fundamental problem which they must address in order to be secure. Because the client is outside of the application's control, users can submit completely arbitrary input to the server-side application. The application must assume that all input is potentially malicious, and must take steps to ensure that attackers cannot use crafted input to compromise the application by interfering with its logic and behavior and gaining unauthorized access to its data and functionality.

This core problem manifests itself in various ways:

- Users can interfere with any piece of data transmitted between the client and the server, including request parameters, cookies, and HTTP headers. Any security controls implemented on the client side, such as input validation checks, can be easily circumvented.
- Users can send requests in any sequence, and can submit parameters at a different stage than the application expects, more than once, or not at all. Any assumption which developers make about how users will interact with the application may be violated.
- Users are not restricted to using only a web browser to access the application. There are numerous widely available tools that operate alongside, or independently of, a browser, to help attack web applications. These tools can make requests that no browser would ordinarily make, and can generate huge numbers of requests quickly to find and exploit problems.

The majority of attacks against web applications involve sending input to the server which is crafted to cause some event that was not expected or desired by the application's designer. Some examples of submitting crafted input to achieve this objective are as follows:

- Changing the price of a product transmitted in a hidden HTML form field, to fraudulently purchase the product for a cheaper amount.
- Modifying a session token transmitted in an HTTP cookie, to hijack the session of another authenticated user.
- Removing certain parameters that are normally submitted, to exploit a logic flaw in the application's processing.
- Altering some input that will be processed by a back-end database, to inject a malicious database query and so access sensitive data.

Needless to say, SSL does nothing to stop an attacker from submitting crafted input to the server. If the application uses SSL, this simply means that

other users on the network cannot view or modify the attacker's data in transit. Because the attacker controls her end of the SSL tunnel, she can send anything she likes to the server through this tunnel. If any of the previously mentioned attacks are successful, then the application is emphatically vulnerable, regardless of what its FAQ may tell you.

Key Problem Factors

The core security problem faced by web applications arises in any situation where an application must accept and process untrusted data that may be malicious. However, in the case of web applications, there are several factors which have combined to exacerbate the problem, and which explain why so many web applications on the Internet today do such a poor job of addressing it.

Immature Security Awareness

There is a less mature level of awareness of web application security issues than there is in longer-established areas such as networks and operating systems. While most people working in IT security have a reasonable grasp of the essentials of securing networks and hardening hosts, there is still widespread confusion and misconception about many of the core concepts involved in web application security. It is common to meet experienced web application developers to whom an explanation of many basic types of flaws comes as a complete revelation.

In-House Development

Most web applications are developed in-house by an organization's own staff or contractors. Even where an application employs third-party components, these are typically customized or bolted together using new code. In this situation, every application is different and may contain its own unique defects. This stands in contrast to a typical infrastructure deployment in which an organization can purchase a best-of-breed product and install it in line with industry-standard guidelines.

Deceptive Simplicity

With today's web application platforms and development tools, it is possible for a novice programmer to create a powerful application from scratch in a short period of time. But there is a huge difference between producing code that is functional and code that is secure. Many web applications are created

by well-meaning individuals who simply lack the knowledge and experience to identify where security problems may arise.

Rapidly Evolving Threat Profile

As a result of its relative immaturity, research into web application attacks and defenses is a thriving area in which new concepts and threats are conceived at a faster rate than is now the case for older technologies. A development team that begins a project with a complete knowledge of current threats may well have lost this status by the time the application is completed and deployed.

Resource and Time Constraints

Most web application development projects are subject to strict constraints on time and resources, arising from the economics of in-house, one-off development. It is not usually possible to employ dedicated security expertise in the design or development teams, and due to project slippage security testing by specialists is often left until very late in the project's lifecycle. In the balancing of competing priorities, the need to produce a stable and functional application by a deadline normally overrides less tangible security considerations. A typical small organization may be willing to pay for only a few man-days of consulting time to evaluate a new application. A quick penetration test will often find the low-hanging fruit, but it may miss more subtle vulnerabilities that require time and patience to identify.

Overextended Technologies

Many of the core technologies employed in web applications began life when the landscape of the World Wide Web was very different, and have since been pushed far beyond the purposes for which they were originally conceived — for example, the use of JavaScript as a means of data transmission in many AJAX-based applications. As the expectations placed on web application functionality have rapidly evolved, the technologies used to implement this functionality have lagged behind the curve, with old technologies stretched and adapted to meet new requirements. Unsurprisingly, this has led to security vulnerabilities as unforeseen side effects emerge.

The New Security Perimeter

Before the rise of web applications, organizations' efforts to secure themselves against external attack were largely focused on the network perimeter. Defending this perimeter entailed hardening and patching the services that it needed to expose, and firewalling access to others.

Web applications have changed all of this. For an application to be accessible by its users, the perimeter firewall must allow inbound connections to the server over HTTP/S. And for the application to function, the server must be allowed to connect to supporting back-end systems, such as databases, mainframes, and financial and logistical systems. These systems often lie at the core of the organization's operations and reside behind several layers of network-level defenses.

If a vulnerability exists within a web application, then an attacker on the public Internet may be able to compromise the organization's core back-end systems solely by submitting crafted data from his web browser. This data will sail past all of the organization's network defenses, in just the same way as does ordinary, benign traffic to the web application.

The effect of widespread deployment of web applications is that the security perimeter of a typical organization has moved. Part of that perimeter is still embodied in firewalls and bastion hosts. But a significant part of it is now occupied by the organization's web applications. Because of the manifold ways in which web applications receive user input and pass this to sensitive back-end systems, they are the potential gateways for a wide range of attacks, and defenses against these attacks must be implemented within the applications themselves. A single line of defective code in a single web application can render an organization's internal systems vulnerable. The statistics described previously, of the incidence of vulnerabilities within this new security perimeter, should give every organization pause for thought.

NOTE **For an attacker targeting an organization, gaining access to the network or executing arbitrary commands on servers may well not be what they really want to achieve. Often, and perhaps typically, what an attacker really desires is to perform some application-level action such as stealing personal information, transferring funds, or making cheap purchases. And the relocation of the security perimeter to the application layer may greatly assist an attacker in achieving these objectives.**

For example, suppose that an attacker wishes to "hack in" to a bank's systems and steal money from users' accounts. Before the bank deployed a web application, the attacker might have needed to find a vulnerability in a publicly reachable service, exploit this to gain a toehold on the bank's DMZ, penetrate the firewall restricting access to its internal systems, map the network to find the mainframe computer, decipher the arcane protocol used to access it, and then guess some credentials in order to log in. However, if the bank deploys a vulnerable web application, then the attacker may be able to achieve the same outcome simply by modifying an account number in a hidden field of an HTML form.

A second way in which web applications have moved the security perimeter arises from the threats that users themselves face when they access a vulnerable application. A malicious attacker can leverage a benign but vulnerable web application to attack any user who visits it. If that user is located on an internal corporate network, the attacker may harness the user's browser to launch an attack against the local network from the user's trusted position. Without any cooperation from the user, the attacker may be able to carry out any action that the user could perform if she were herself malicious.

Network administrators are familiar with the idea of preventing their users from visiting malicious web sites, and end users themselves are gradually becoming more aware of this threat. But the nature of web application vulnerabilities means that a vulnerable application may present no less of a threat to its users and their organization than a web site that is overtly malicious. Correspondingly, the new security perimeter imposes a duty of care on all application owners to protect their users from attacks against them delivered via the application.

The Future of Web Application Security

Several years after their widespread adoption, web applications on the Internet today are still rife with vulnerabilities. Understanding of the security threats facing web applications, and effective ways of addressing these, remains immature within the industry. There is currently little indication that the problem factors described previously are going to go away in the near future.

That said, the details of the web application security landscape are not static. While old and well understood vulnerabilities like SQL injection continue to appear, their prevalence is gradually diminishing. Further, the instances that remain are becoming more difficult to find and exploit. Much current research is focused on developing advanced techniques for attacking more subtle manifestations of vulnerabilities which a few years ago could be easily detected and exploited using only a browser.

A second prominent trend is a gradual shift in attention from traditional attacks against the server side of the application to those that target other users. The latter kind of attack still leverages defects within the application itself, but it generally involves some kind of interaction with another user, to compromise that user's dealings with the vulnerable application. This is a trend that has been replicated in other areas of software security. As awareness of security threats matures, flaws in the server side are the first to be well understood and addressed, leaving the client side as a key battleground as the learning process continues. Of all the attacks described in this book, those against other users are evolving the most quickly, and are the focus of most current research.

Chapter Summary

In a few short years, the World Wide Web has evolved from purely static information repositories into highly functional applications that process sensitive data and perform powerful actions with real-world consequences. During this development, several factors have combined to bring about the weak security posture demonstrated by the majority of today's web applications.

Most applications face the core security problem that users can submit arbitrary input. Every aspect of the user's interaction with the application may be malicious and should be regarded as such unless proven otherwise. Failure to properly address this problem can leave applications vulnerable to attack in numerous ways.

All of the evidence about the current state of web application security indicates that this problem has not been resolved on any significant scale, and that attacks against web applications present a serious threat both to the organizations that deploy them and to the users who access them.