The Weakest Link:
Mitigating Web Application Vulnerabilities

webScurity White Paper

webScurity Inc.
Minneapolis, Minnesota USA

March 19, 2008

Contents

Section 1

Executive Summary

Insecure Web applications provide an attacker a way through even the most robust perimeter defenses. This paper describes Web application vulnerabilities, their respective impact on overall security, and techniques to mitigate them.

The door is open to customers and attackers alike

For a long time, the prevailing wisdom has been that firewalls = security. But as networks have become more and more secure, attackers are seeking out easier targets. They are increasingly discovering that manipulating applications can be much more "productive". After all, they can get in through the same "door" in the perimeter defenses deliberately left open for legitimate users such as customers, suppliers, and trading partners.

Since 2001, webScurity has helped organizations of all sizes address Web application security. Industries from financial services to healthcare - food services to distribution - have benefited from our experience and understanding of Web application security.

Section 2

Introduction

One of the most important elements of a secure e-commerce environment is the applications. Web applications written and deployed without security as a prime consideration can inadvertently:

- Expose sensitive or confidential information
- Facilitate website defacement
- Provide access to private networks
- Perpetrate Denial of Service (DoS) attacks
- Facilitate unhindered access to back-end databases

Application attacks cannot be prevented by network firewalls, intrusion-detection, or even encryption. These attacks work by exploiting the Web server and the applications it runs, meaning the attackers enter through the same open "door" in the perimeter defenses that customers use to access the website. Malicious activity cannot be distinguished from normal, everyday Web traffic.

Target Audience

This paper is targeted toward security specialists, system administrators, developers, and other individuals interested in learning the importance of Web application security.

Background

The stateless nature of HTTP doesn't help

Many of the most serious – and difficult to detect - Web application attacks take advantage of the stateless nature of the Hypertext Transfer Protocol (HTTP) and the special programming required to develop useful applications for the Web. There are two types of attacks:

- Indiscriminate - the attacker has no interest in who your organization is or what it does, but simply the fact that you have a Web server
- Targeted (discriminate) - the attacker has selected a Web site for very specific reasons which might include financial gain, publicity, business disruption, etc.

Worms are examples of indiscriminate attacks, while the various forms of parameter manipulation (such as cookie tampering discussed below) are typically targeted attacks.

Section 3

Common Web Application Attacks

Defacement

Web site defacement can largely be prevented through detailed server hardening by an experienced system administrator. However, operating systems, Web servers (HTTP servers like Apache, IIS, etc.), and other server software packages are typically shipped in an insecure configuration. It is up to a seasoned administrator to ensure all components are configured securely.

It is widely reported that the vast majority (perhaps as high as 90%) of all attacks exploit improperly configured systems.

Worms

Possibly the most common type of indiscriminate attack, worms like Code Red, Nimda, and Slapper spread at an incredible pace. Often, they can infect hundreds of thousands of servers in a matter of minutes.

Worms exploit vulnerabilities in widely-used server software. While fixes for the vulnerabilities are generally released quickly, it is little consolation to the hundreds of thousands of organizations that suffer countless damages due to downtime, lost productivity, and general business disruption.

Buffer Overruns

One of the most common attacks exploits a common programming mistake known as “buffer overruns” (also referred to as “buffer overflows”). Virtually every mainstream software product – commercial or open-source – has had a buffer overrun issue of one sort or another.

While some buffer overrun conditions can be relatively harmless, others can be as serious as allowing the attacker administrator access to the server and/or network.

URL Parameter Tampering

Web applications that use the URL query string to pass information from page-to-page (a widely-used way of maintaining state) are susceptible to parameter tampering. This attack is literally as simple as altering the query string parameter values in the browser’s address bar!

For example, consider the following URL:

http://www.website.com/show_account.asp?loggedin=true&acct=123456789

What happens if the "acct" parameter's value is changed to "223456789" and sent back to the server (as shown below)?

http://www.website.com/show_account.asp?loggedin=true&acct=223456789

Banner-grabbing

The first step in any attack is selecting the appropriate "toolbox". Two pieces of information that determine the attacker's toolbox choice are:

- Operating system
- Web server type

A simple process known as banner-grabbing provides the attacker with both these pieces of information. Using a telnet utility, the attacker can quickly determine what operating system a given server is running, as well as the type of Web server (Apache, IIS, etc.) it is using. Knowing this, the attacker can construct attacks appropriate for the environment.

Hidden Field Manipulation

Closely related to URL parameter tampering (see above), hidden field manipulation involves altering parameter values, except these are "hidden" from the user.

Hidden fields are an HTML input type (<input type="hidden" …>) that are NOT rendered by the browser and shown to the user – they are "hidden". However, because most browsers provide a "View/Source" option, it is easy for anyone to see – and alter – hidden field values.

The "acct" parameter from the example above could have been implemented as a hidden field. This would have made it less obvious and slightly increased the difficulty of altering its value, but it doesn't offer much protection against even a novice attacker!

Cookie Tampering

Like URL query string parameters and hidden fields, cookies are a popular way to maintain state information. While they are slightly more difficult to locate and alter, their relative level of obscurity compared to the other two methods is not sufficient to prevent malicious activity. Freely available tools help automate the process of cookie tampering.

SQL Injection

Data-entry fields can unknowingly provide a convenient way for an attacker to access an SQL-compliant back-end database and execute

their own SQL statements. Any user input (including hidden fields, URL query string parameters and cookie values) used by the Web application to construct SQL statements (usually part of the "where" clause) could result in the unintended execution of SQL statements "injected" by an attacker.

SQL injection can be used to bypass authentication, gain unrestricted access to data, and in extreme cases, allow an attacker to execute destructive SQL statements like "DROP TABLE users;"!

Stealth Commanding

Similar to SQL injection, stealth commanding allows an attacker to execute operating system commands on the server without proper authorization.

Any user input (including hidden fields, URL query string parameters, and cookie values) used by the Web application to construct operating system calls (shell scripts, commands, etc.) could allow an attacker to execute commands of their choice. Because most Web servers (Apache, IIS, etc.) run as a privileged group/user, the commands executed by the Web application (on behalf of the attacker) are also privileged. This is only one example of how stealth commanding can be accomplished.

Cross-site Scripting

Cross-site scripting is the act of injecting malicious HTML tags or client-side scripting code into HTML form fields. The contents of these fields are saved to a database where they are later retrieved and automatically executed by an innocent user. Essentially, it is a way to affect another user's experience.

Forceful Browsing

Forceful browsing is the act of directly accessing pages (URL's) without consideration for its context within an application. Bypassing intended application flow can lead to unauthorized access to resources which could mean a press release getting out before expected to circumventing authentication altogether.

Section 4

Inadvertently Helping the Attacker

Information that would be insignificant to a normal user is deadly in the hands of skilled attacker. Something as simple as disclosing the Web server platform (see banner grabbing discussion above) helps an attacker form the basis of their attack.

Error messages are a great source of intelligence for attackers

An attacker can also gather valuable information from error messages reported by the application. The common practice of reporting detailed error messages can aid the support process in controlled environments. However, when your audience is literally anyone in the world with an Internet connection, a different approach is needed.

For example, database error messages from a failed query frequently contain details of the environment better left undisclosed, such as the database platform, vendor, and version. They may also include details of the database's table structure and layout.

Don't give them any hints

Letting a would-be attacker know they have entered an incorrect user id or password, specifically, gives them an unnecessary advantage. Generic messages such as "Log in failed" are much safer.

Section 5

What You Can Do

Proper server hardening and up-to-date patches	Indiscriminate attacks can often be prevented through extensive server hardening and keeping patches up-to-date. However, even the latest patches only provide protection against known attacks. These measures are important, but unfortunately, are reactive rather than proactive.
Developer training, coding standards, code reviews	Build security into the development cycle by training programmers to write secure Web applications and conducting 3rd-party source code assessments. Clear standards and regular internal assessments will also help ensure secure applications.
Give away as little as possible	Disabling Web server identification and avoiding detailed error messages will slow down an attacker tremendously.
Web application firewall	Install a Web application firewall to ensure all traffic that reaches the Web server and its applications will be exactly what is expected.

Section 6

Conclusions/Recommendations

The degree to which insecure Web applications can undermine perimeter defenses cannot be over-emphasized. The most robust security measures will not protect back-end systems if holes exist in the applications.

Focused attention to the environment behind the firewall is recommended to ensure on-going security. In addition to proper server hardening, secure coding practices, and frequent assessments, a Web application firewall is the perfect compliment to existing perimeter defenses.

Section 7

Contacts

For more information, please contact info@webscurity.com.

webScurity Inc.
9298 Central Ave NE
Suite 402
Minneapolis, Minnesota USA 55434

Toll Free – 866.SCURITY (728.7489)

International/Twin Cities Metro - 1.763.786.2009

Fax – 1.763.786.3680

info@webscurity.com

www.webscurity.com

Appendix 1: References

Open Web Application Security Project

The Open Web Application Security Project (OWASP) is dedicated to finding and fighting the causes of insecure software. Everything here is free and open source. The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work. Participation in OWASP is free and open to all.

http://www.owasp.org

Web Application Security Consortium

The Web Application Security Consortium (WASC) is an international group of experts, industry practitioners, and organizational representatives who produce open source and widely agreed upon best-practice security standards for the World Wide Web.

http://www.webappsec.org