Web Application Security Overview

Software vulnerabilities are being discovered at an ever-increasing rate. For the year of 2006, the National Institute of Standards and Technology (NIST) National Vulnerability Database recorded over 6,600 vulnerabilities. A notable trend has emerged from this and other data: as operating systems become more secure there has been a shift towards the discovery of application layer vulnerabilities. A particularly disturbing consequence is that organizations have reported a corresponding increase in attacks against web applications that can provide a direct avenue to critical data assets. One managed security provider reported that the number of such attacks against their clients increased from 100-200 per day to over 8,000 per day in the first three months of 2006[1]. Given this increasingly hostile environment, the security of web applications is becoming more critical every day.

Open Web Application Security Project

Top Ten 2007

1. Cross Site Scripting (XSS)
2. Injection Flaws
3. Malicious File Execution
4. Insecure Direct Object Reference
5. Cross Site Request Forgery (CSRF, XSRF, or Session Riding)
6. Information Leakage and Improper Error Handling
7. Broken Authentication and Session Management
8. Insecure Cryptographic Storage
9. Insecure Communications
10. Failure to Restrict URL Access

Other high priority items

- Buffer Overflows, Integer Overflows, Format String Issues
- Denial of Service Attacks
- Insecure configuration management

Users who access web applications must be allowed through network perimeter security. However, a user may be an attacker in disguise and now is one step closer to data assets. The increased reliance on web applications, the blurring of the desktop and web environment, the rate of adoption of new web application technology, and the growing complexity of web applications create an environment ripe for vulnerabilities. All of these factors further emphasize the need for attention to web application security.

Web application security starts with secure network, web server, and operating system configuration. NSA has published a number of guides that are available at the sites referenced at the end of this document. However,

many problems associated with web applications arise from the design of the application itself, and not any specific configuration issue. Many resources are available for the manager, the developer, and the system administrator to guide, write, and deploy secure web applications. One well-respected industry source is the Open Web Application Security Project (OWASP), an open community dedicated to application security. OWASP's extensive library and collection of tools is freely available at http://www.owasp.org. A great place to start is the OWASP Top Ten Project (http://www.owasp.org/index.php/OWASP_Top_Ten_Project). The OWASP document provides a list of critical web application security flaws and detailed suggestions for remediation. See inset box for a brief summary[2]. Also, see the NSA follow up information sheet, “Web Application Security Vulnerabilities”, which provides more detail on the issues.

For access to a variety of configuration guidance and discussions on numerous related security topics, visit the NSA library on the Internet at http://www.nsa.gov/snac or on the SIPRNet at http://www.iad.nsa.smil.mil/resources/library.

References

1. Kirk, Jeremy. "Databases assaulted by SQL injection attacks", Network World, http://www.networkworld.com/news/2006/071906-databases-sql-injection.html
2. Reproduced in accordance with the Creative Commons License as referenced on the OWASP web page and described at http://creativecommons.org/licenses/by-sa/2.5/

Web Application Security Vulnerabilities

As operating systems become more secure, attackers are moving up the stack to target application vulnerabilities. The Open Web Application Security Project (OWASP), an open community dedicated to application security, has developed a list of the top ten web application vulnerabilities. This list serves to educate managers, developers, and administrators to these most common vulnerabilities in the hopes of improving security. The list is summarized below.

1. Cross Site Scripting (XSS) - XSS vulnerabilities exist when a web application returns user input to the browser without validating the input. This user input is passed to a client web browser where it is executed. A blog is an example application that may be vulnerable to XSS.

2. Injection Flaws - Web applications are vulnerable to injection flaws if they take user-supplied data and directly use it to construct commands or generate queries for use in an interpreter. This input can trick the interpreter into performing operations which were never intended, including reading or modifying web application data. An example would be entering a malicious database command into a web form in such a way that it would be executed.

3. Malicious File Execution - Web applications may be developed with an inherent trust of files that they utilize. If user-supplied data is used to construct references such as file system paths and URLs, it may be possible for an attacker to supply malicious input and redirect the application to process hostile data. For example, in some languages a variable may be used in an include statement, allowing the attacker to include their own malicious code.

4. Insecure Direct Object Reference - Direct object references are URL or form parameters that expose files, directories, database records, or keys. If these items are exposed by web applications and not secured with access control mechanisms, they may be manipulated to access other objects without authorization. For example, if an application stores the shopping cart ID as a hidden field on the client, the client could change the cart ID to access someone else's cart.

5. Cross Site Request Forgery (CSRF, XSRF, or Session Riding) - CSRF attacks exploit the trust or authentication relationship between web applications and client browsers. With each request, web browsers automatically send previously set cookies. Other websites can exploit this trust by causing the browser to make a request to the CSRF vulnerable website, without the victim's knowledge. For example, if a user is logged into his/her bank account while browsing other websites, a malicious website could run scripts on the client that transfer money from one bank account to another by submitting a request via an image source tag.

6. Information Leakage and Improper Error Handling - Web applications may reveal sensitive or private information if they don't handle errors properly. These vulnerabilities may be used to develop further penetrating attacks or to violate privacy. For example, when a log-on requires a username and password, the message for an invalid user should be the same as the

message for an invalid password so that an attacker can not glean information about valid user accounts on the system.

7. Broken Authentication and Session Management - Web applications may not properly protect account credentials and session tokens. This could allow attackers to compromise passwords, keys, or authentication tokens and assume other users' identities. For example, a web browser receives a cookie with session information from a website. If the session state is not destroyed, an attacker may reuse this cookie later.

8. Insecure Cryptographic Storage - Web applications may use cryptographic functions improperly to protect data and credentials. This could allow attackers to compromise data confidentiality or integrity. An example is using proven weak algorithms such as MD5.

9. Insecure Communications - Web applications must encrypt sensitive data when sent over the network. Failure to do so would allow an attacker to eavesdrop on the communications. For example, if HTTP is not sent over SSL, the unencrypted session token may be sniffed and used to hijack the session.

10. Failure to Restrict URL Access - Web servers must control access to files and directories. Attackers can use “forceful browsing” (guessing and brute force methods) to find unadvertised links and view sensitive information or perform unauthorized operations. An example is directory traversal, where an application allows a user to enter a path such as "/../../" and get access to every file on the system.

Other high priority items that are causes of many vulnerabilities:

- **Buffer Overflows, Integer Overflows, Format String Issues** - Web application components such as CGI, libraries, drivers, and server components may be vulnerable, crashed, or used to take control of a process to execute arbitrary commands.
- **Denial of Service** - Attackers may consume application resources to a point where legitimate users can no longer access the application.
- **Insecure configuration management** - Implementing a strong server configuration standard is critical to a secure web application. These servers have many configuration options that affect security and are frequently not secure out of the box.

Details on the Open Web Application Security Project Top Ten can be found at http://www.owasp.org/index.php/OWASP_Top_Ten_Project. For access to a variety of configuration guidance and discussions on numerous related security topics, visit the NSA library on SIPRNet at http://www.iad.nsa.smil.mil/resources/library or on the Internet at http://www.nsa.gov/snac.