SECURING APPLICATIONS FROM HACKERS

By
Norhazimah Abdul Malek
National ICT Security and Emergency Response Centre (NISER)
(This article was published in Computimes on 28 November 2005)

MOST companies today use the Web to do business with customers, employees, suppliers and others. This is because it is easier to maintain a Web-based application than a Windows-based one. But how can we be sure that a Web-based application is secured? Or that data is being shared only by the authorised users?

The Gartner Group estimates that 75 per cent of cyber attacks today are at the application level. And about 97 per cent of over 300 Web sites audited are vulnerable to Web application attacks. The US Federal Bureau of Investigation also reveals that 95 per cent of the companies are hacked from Web applications, and only five per cent of them are aware of the attacks (http://conference.hackinthebox.org/hitbsecconf2005kl/materials/TT-Shreeraj-Shah-Webhacking-Kungfu.pdf).

From the figures, we can deduce that most company Web sites are prone to cyber attacks, and some of these companies are not aware that their Web applications have vulnerabilities that can be exploited by hackers.

According to statistics published by the National ICT Security and Emergency Response Centre, there have been significant increases in Web defacement incidents. In the first quarter of this year, there were 256 Web defacements involving both public and private Web sites, compared to the previous quarter which recorded 42 of such incidents.

To have a secure Web application, developers of the application must know each attribute such as query string, form, cookie, script, etc, because they are vulnerable. These attributes can be exploited by an attacker and expose sensitive company information if they are not used securely.

Web Application Attacks

There are two types of Web application attacks: automated and manual. Automated attacks can be used to exploit a Web application using automated Web application attack tools such as wget, curl, blackwidow and teleport pro. Using these automated tools, crawling and attacks can be done shortly.

This type of attack can be avoided by setting “honey traps” using HTTP Module (used in pre/post-processing of requests). The attacker can be put into an infinite loop using defence trick once it is trapped.

To launch manual attacks, hackers must conduct information gathering such as address identification, port scanning, social engineering and vulnerability scanning to find out vulnerabilities that can be exploited.

Common Web application hacking methods include:

- **Source code disclosure**: The attacker uses this technique to obtain the source code of the server-side script such as active server page (ASP), Java server page (JSP) and PHP hypertext preprocessor (PHP) files, to get information on the Web application logic such as database structure, source code comments and parameters.

  There are two types of malicious code injections which may allow the source code disclosure technique to be used: client-side code injection and server-side code injection.

  An example of client-side code injection is cross-site scripting attacks that occur when the attackers embed malicious code such as script into a hyperlink. When the user clicks on the hyperlink, the malicious code will be executed at the Web server, which creates an output page containing the malicious content that can lead to internal data disclosure.

  An example of server-side code injection is remote command execution that occurs when the attacker injects PHP/ASP code which can cause arbitrary command execution on the server.

  This problem occurs because of poor design and written applications. Web developers should include exception handling in the programming so errors can be handled within the code. The errors should be logged and not displayed at the Web browser.

  All inputs such as data types, buffer sizes and metacharacters should be sanitised and validated before being passed to the internal application logic.

  To ensure that a Web application is secured from this kind of attack, the developer should follow the secure coding practices to make sure that no "active code" is injected as data contents.

- **SQL query poisoning**: Normally, Web applications send query strings and their parameters to the database server to get the requested data from the database. Attackers may take advantage of this because they can embed SQL commands inside these parameters, and this is called SQL query poisoning. This kind of attack may lead to back-end database server compromise.

  SQL query poisoning attacks occur because there is no input validation for all inputs from the client. This is a result of bad programming practice.

  A database should be configured correctly to eliminate unnecessary database users and stored procedures. Using alternative SQL query constructions such as stored procedures and prepared statements will overcome SQL query poisoning problems because the SQL string cannot be altered.

- **Session hijacking**: Hypertext transfer protocol (HTTP) connections are stateless. To keep track of an application's state when the application runs, an HTTP cookie is used. Cookies will be destroyed when the user logs out from the system.

  Nowadays, there are tools that can be used to intercept HTTP connections and alter the cookies' value, and this is called session hijacking. If the attackers successfully hijack a session, they can gain access to all of the user's data and make changes to the data.

  Session identifiers, which are unique and generated randomly, can be used to prevent such attacks. These identifiers are transmitted between the client and the server.

  To secure session identifiers, make sure that they are not stored in the hidden field, and encrypt them to prevent captured, brute-forced or reverse-engineered exploitation.

Conclusion

Web application attacks are increasing drastically because there is a lack of knowledge in securing the applications, especially during the development and deployment stages of the applications. To control or avoid this menace, we must ensure that security is being implemented not only during the coding stage, but also the deployment stage.

The operations of a Web application must be monitored by the administrator so any exploits can be detected earlier and damages can be minimised or avoided such as using an intrusion detection system to monitor and filter Web traffic.

It is also recommended for all organisations to conduct a security audit assessment to ensure that an application is secured before it is published to the public.

Acknowledgement

The author would like to thank En Zahri Yunos, Strategic Planning Manager of NISER who has given valuable inputs to this article.

Appendix A - Example of Web Application Exploits

Type of Exploit	Description	Result
Authentication Hijacking	Unsecured credential and identity management	Account hijacking and theft of service
Parameter Tampering	Modified data send to web server	Attacker gains access to all records in database
Buffer Overflow	Attackers flood server with requests that exceed buffer size	Attackers crash and take control of server
Command Injection	Web application passes malicious commands to back-end server	Attackers gain access to data
Cookie Snooping	Attacker decodes user credentials	Attacker can log on as user and gain access to unauthorized information
SQL Injection	Web application passes malicious command to database	Attacker can modify data
Cookie Poisoning	Attacker manipulates cookies passed from server to browser	Attacker can gain access and modify data
Cross-site Scripting	Malicious code is executed when user clicks on a URL	User credentials and information can be stolen
Invalid Parameters	Malicious data accepted without validation	Attacker can hijack client accounts and steal data
Forceful Browsing	Client accesses unauthorized URL	Attacker accesses off-limit directories

10 Most Aggressive Web Application Exploit – Paul Desmond, Network World. "All-out blitz against Web app attacks". *May 17, 2004. URL: http://www.networkworld.com/techinsider/2004/0517techinsidermain.html?page=1*