Volume 3, Issue 1, January 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Review on Web Application Security Vulnerabilities

Ashwani Garg
Assistant Professor, Deptt. Of CSE
Panipat Institute of Engg. & Technology, Samalkha

Shekhar Singh
Assistant Professor, Deptt. Of CSE
Panipat Institute of Engg. & Technology, Samalkha

Abstract— Web applications are important, common distributed systems whose current security relies primarily on server-side mechanisms. Web applications provide end users with client access to server functionality through a set of Web pages. These pages often contain script code to be executed dynamically within the client Web browser. Most Web applications aim to enforce simple, intuitive security policies, such as, for Web-based email, disallowing any scripts in untrusted email messages. Even so, Web applications are currently subject to a plethora of successful attacks, such as cross-site scripting, cookie theft, session riding, browser hijacking, and the recent self-propagating worms in Web-based email and social networking sites . This paper looks at five common Web application vulnerabilities, their examples and countermeasures to eliminate common security exploits and to secure the emerging class of rich, cross-domain Web applications.

Keywords— Web Vulnerabilities, Web Security Flaws.

I. INTRODUCTION

The increased usage of the Internet and network technology has changed the focus in assessing computer environments. The traditional approach considered the location of hardware and equipment first and then the data stored on the hardware. Also, these assessments were primarily at specific points in time and primarily compliance-based reviews. With network-based technology (a.k.a. net centric technology), the primary concern is on the network and the contents of information or objects. Also, an assessment of network technology is focused on the implementation and management of real-time controls to meet the business needs and to provide continuous monitoring [2]. Security is not a one-time event. It is insufficient to secure your code just once. A secure coding initiative must deal with all stages of a program's lifecycle. Secure web applications are only possible when a secure SDLC is used. Secure programs are secure by design, during development, and by default.

II. REVIEW FROM LITERATURE

No language can prevent insecure code, although there are language features which could aid or hinder a security-conscious developer [1]. Insecure software is already undermining our financial, healthcare, defence, energy, and other critical infrastructure. As our digital infrastructure gets increasingly complex and interconnected the difficulty of achieving application security increases exponentially. We can no longer afford to tolerate relatively simple security problems like those presented below. The vulnerabilities [12] explained in this paper are:

1. Remote code execution
2. SQL injection
3. Format string vulnerabilities
4. Cross Site Scripting (XSS)
5. Username enumeration

III. VULNERABILITIES

2.1 Remote code execution

As the name suggests, this vulnerability allows an attacker to run arbitrary, system level code on the vulnerable server and retrieve any desired information contained therein. Improper coding errors lead to this vulnerability [6,7]. At times, it is difficult to discover this vulnerability during penetration testing assignments but such problems are often revealed while doing a source code review. However, when testing Web applications is is important to remember that exploitation of this vulnerability can lead to total system compromise with the same rights as the Web server itself [13].
Rating: Highly Critical
Here we will look at such type of critical vulnerability:

- Exploiting register_globals in PHP: Register_globals is a PHP setting that controls the availability of "superglobal" variables in a PHP script (such as data posted from a user's form, URL-encoded data, or data from cookies). In earlier releases of PHP, register_globals was set to "on" by default, which made a developer's life easier - but this lead to less secure coding and was widely exploited. When register_globals is set to "on" in

php.ini, it can allow a user to initialize several previously uninitialized variables remotely. Many a times an uninitialized parameter is used to include unwanted files from an attacker, and this could lead to the execution of arbitrary files from local/remote locations. For example:

```
require ($page . ".php");
```

Here if the $page parameter is not initialized and if register_globals is set to "on," the server will be vulnerable to remote code execution by including any arbitrary file in the $page parameter. Now let's look at the exploit code:

http://www.vulnsite.com/index.php?page=http://www.attacker.com/attack.txt

In this way, the file "http://www.attacker.com/attack.txt" will be included and executed on the server. It is a very simple but effective attack.

Countermeasures:

1. More recent PHP versions have register_globals set to off by default; however some users will change the default setting for applications that require it. This register can be set to "on" or "off" either in a php.ini file or in a .htaccess file. The variable should be properly initialized if this register is set to "on." Administrators who are unsure should question application developers who insist on using register_globals.
2. It is an absolute must to sanitize all user input before processing it. As far as possible, avoid using shell commands. However, if they are required, ensure that only filtered data is used to construct the string to be executed and make sure to escape the output.

2.2 SQL Injection

SQL injection is a very old approach but it's still popular among attackers. This technique allows an attacker to retrieve crucial information from a Web server's database [18]. Depending on the application's security measures, the impact of this attack can vary from basic information disclosure to remote code execution and total system compromise [3,5].

Rating: Moderate to Highly Critical

Here is an example of vulnerable code in which the user-supplied input is directly used in a SQL query:

- <form action="sql.php" method="POST" /> Name: <input type="text" name="name" />
 <input type="submit" value="Add Comment" /> </form> <?php $query = "SELECT * FROM users WHERE username = '{$_POST['username']}'"; $result = mysql_query($query); ?>

The script will work normally when the username doesn't contain any malicious characters. In other words, when submitting a non-malicious username (steve) the query becomes:

```
$query = "SELECT * FROM users WHERE username = 'steve'";
```

However, a malicious SQL injection query will result in the following attempt:

```
$query = "SELECT * FROM users WHERE username = '' or '1=1'";
```

As the "or" condition is always true, the mysql_query function returns records from the database. A similar example, using AND and a SQL command to generate a specific error message, is shown in the URL below in Figure 1.

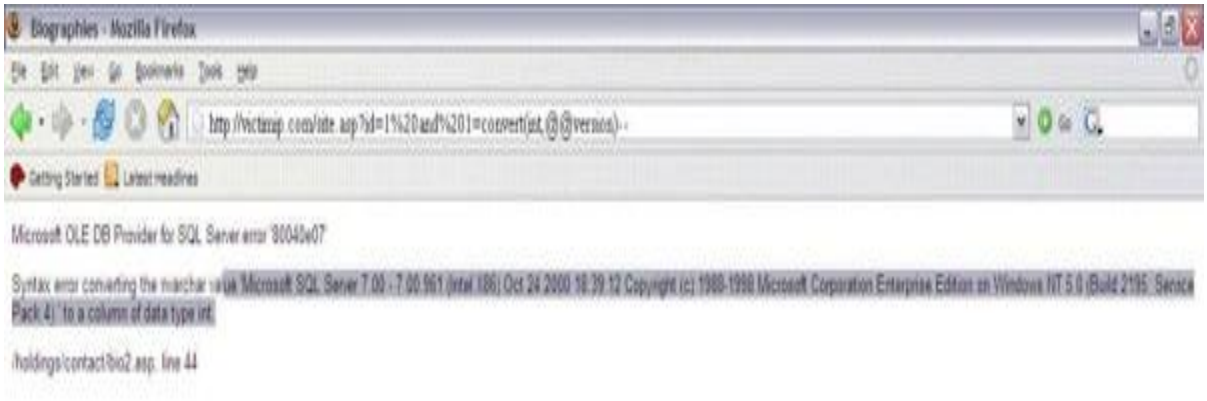

Figure 1. Error message displaying the MS SQL server version.

It is obvious that these error messages help an attacker to get a hold of the information which they are looking for (such as the database name, table name, usernames, password hashes etc). Thus displaying customized error messages may be a good workaround for this problem, however, there is another attack technique known as Blind SQL Injection where the attacker is still able to perform a SQL injection even when the application does not reveal any database server error message containing useful information for the attacker.

Countermeasures:

1. Avoid connecting to the database as a superuser or as the database owner. Always use customized database users with the bare minimum required privileges required to perform the assigned task.
2. If the PHP magic_quotes_gpc function is on, then all the POST, GET, COOKIE data is escaped automatically.
3. PHP has two functions for MySQL that sanitize user input: addslashes (an older approach) and mysql_real_escape_string (the recommended method). This function comes from PHP >= 4.3.0, so you should check first if this function exists and that you're running the latest version of PHP 4 or 5. MySQL_real_escape_string prepends backslashes to the following characters: \x00, \n, \r, \, ', "and \x1a.

2.3 Format String Vulnerabilities

This vulnerability results from the use of unfiltered user input as the format string parameter in certain Perl or C functions that perform formatting, such as C's printf().A malicious user may use the %s and %x format tokens, among others, to print data from the stack or possibly other locations in memory [4,8]. One may also write arbitrary data to arbitrary locations using the %n format token, which commands printf() and similar functions to write back the number of bytes formatted. This is assuming that the corresponding argument exists and is of type int *.Format string vulnerability attacks fall into three general categories: denial of service, reading and writing [14,15].

Rating: Moderate to Highly Critical

Here is the piece of code in miniserv.pl which was the cause of vulnerability in Webmin:

- if ($use_syslog && !$validated) { syslog("crit", ($nonexist ? "Non-existent" : $expired ? "Expired" : "Invalid"). " login as $authuser from $acpthost"); }

In this example, the user supplied data is within the format specification of the syslog call.
The vectors for a simple DoS (Denial of Service) of the Web server are to use the %n and %0(large number)d inside of the username parameter, with the former causing a write protection fault within Perl – and leading to script abortion. The latter causes a large amount of memory to be allocated inside of the perl process.

Countermeasure:

Edit the source code so that the input is properly verified.

2.4 Cross Site Scripting(XSS)

The success of this attack requires the victim to execute a malicious URL which may be crafted in such a manner to appear to be legitimate at first look. When visiting such a crafted URL, an attacker can effectively execute something malicious in the victim's browser [9,16]. Some malicious Javascript, for example, will be run in the context of the web site which possesses the XSS bug [19].

Rating: Less to Moderately Critical

Here is a sample piece of code which is vulnerable to XSS attack:

- <form action="search.php" method="GET" /> Welcome!! Enter your name: <input type="text" name="name_1" />
 <input type="submit" value="Go" />
 </form> <?php echo " Your Name
"; echo ($_GET[name_1]); ?>

In this example, the value passed to the variable 'name_1' is not sanitized before echoing it back to the user. This can be exploited to execute any arbitrary script.
Here is some example exploit code:

http://victim_site/clean.php?name_1=<script>code</script> or

http://victim_site/clean.php?name_1=<script>alert(document.cookie);</script>

Countermeasures:
The above code can be edited in the following manner to avoid XSS attacks:

- <?php $html= htmlentities($_GET['name_1'],ENT_QUOTES, 'UTF-8'); echo "<p>Your Name
"; echo ($html); ?>

2.5 Username enumeration

Username enumeration is a type of attack where the backend validation script tells the attacker if the supplied username is correct or not[10,11,17]. Exploiting this vulnerability helps the attacker to experiment with different usernames and determine valid ones with the help of these different error messages.

Rating: Less Critical

Here is an example of login screen:

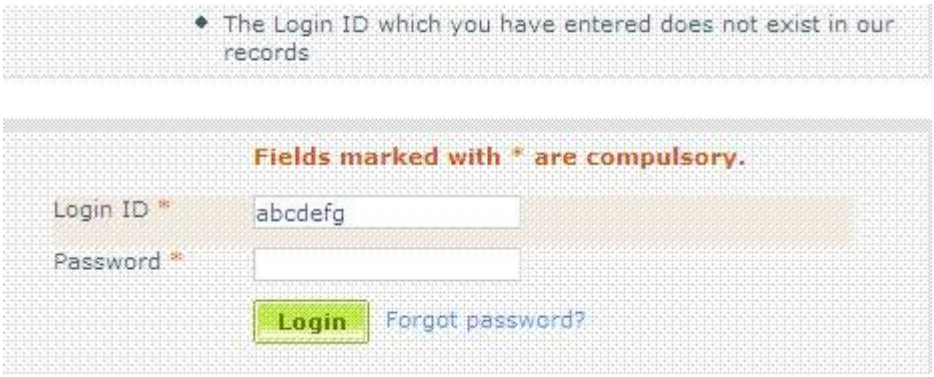

Figure 2. Sample login screen.

Below the response given when a valid username is guessed correctly:

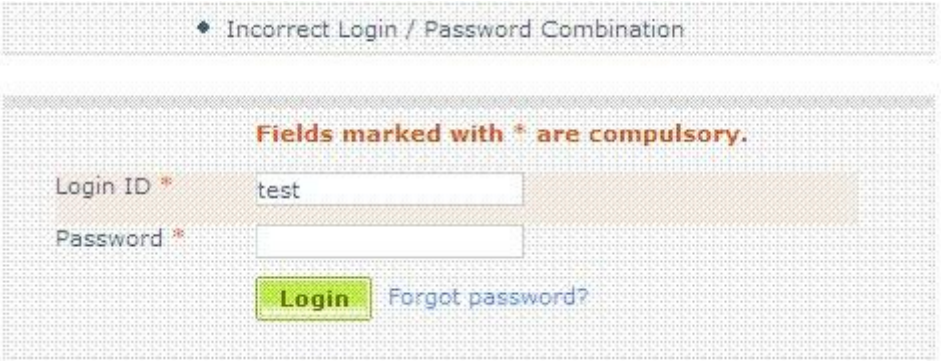

Figure 3. Valid username with incorrect password.

Username enumeration can help an attacker who attempts to use some trivial usernames with easily guessable passwords, such as test/test, admin/admin, guest/guest, and so on. These accounts are often created by developers for testing purposes, and many times the accounts are never disabled or the developer forgets to change the password.
During pen testing assignments, the authors of this article have found such accounts are not only common and have easily guessable passwords, but at times they also contain sensitive information like valid credit card numbers, passport numbers, and so on. Needless to say, these could be crucial details for social engineering attacks.

Countermeasures:
Display consistent error messages to prevent disclosure of valid usernames. Make sure if trivial accounts have been created for testing purposes that their passwords are either not trivial or these accounts are absolutely removed after testing is over - and before the application is put online.

IV. CONCLUSIONS

Web applications reach out to a larger, less-trusted user base than legacy client-server applications, and yet they are more vulnerable to attacks. Many companies are starting to take initiatives to prevent these types of break-ins. Code reviews, extensive penetration testing, and intrusion detection systems are just a few ways that companies are battling a growing problem.Unfortunately, most of the solutions available today are using negative security logic (working with a list of attacks and trying to prevent against them). Negative security logic solutions can prevent known, generalized

attacks, but are ineffective against the kind of targeted, malicious hacker activity outlined in this paper. In this paper,we have demonstrated five common web application vulnerabilities, their countermeasures and their criticality. If there is a consistent message among each of these attacks, the key to mitigate these vulnerabilities is to sanitize user's input before processing it.

REFERENCES

[1] OWASP Top 10 Web Application Vulnerabilities.http:// www.applicure.com/blog/owasp-top-10-2010
[2] E. Chien. Malicious Yahooligans. http://www. symantec.com/avcenter/reference/malicious. yahooligans.pdf, 2006.
[3] S. Di Paola. Wisec security. http://www.wisec.it/ sectou.php?id=44c7949f6de03, 2006.
[4] Ú. Erlingsson and F. B. Schneider. IRM enforcement of Java stack inspection. In Proc. IEEE Security and Privacy, 2000.
[5] O. Hallaraker and G. Vigna. Detecting malicious JavaScript code in Mozilla. In Proc. IEEE Conf. on Engineering of Complex Computer Systems, 2005.
[6] B. Hoffman. Ajax security. http://www.spidynamics. com/assets/documents/AJAXdangers.pdf, 2006.
[7] T. Jim, N. Swamy, and M. Hicks. Defeating script injection attacks with browser-enforced embedded policies. In WWW, 2007.
[8] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic. Noxes: A client-side solution for mitigating cross-site scripting attacks. In ACM Symp. on Applied Computing, 2006.
[9] M. Johns. SessionSafe: Implementing XSS immune session handling. In Proc. ESORICS, 2006.
[10] B. Livshits and M. S. Lam. Finding security errors in Java programs with static analysis. In Proc. Usenix Security Symp., 2005.
[11] Microsoft ASP.NET AJAX. http://ajax.asp.net.Y. Minamide. Static approximation of dynamically generated Web pages. In Proc. WWW, 2005.
[12] MITRE. Common vulnerabilities and exposures. http:// cve.mitre.org/cve/, 2007.
[13] Open Web Application Security Project. The ten most critical Web application security vulnerabilities. http://umn.dl.sourceforge.net/sourceforge/ owasp/OWASPTopTen2004.pdf, 2004.
[14] T. Pixley. DOM level 2 events specification. http://www. w3.org/TR/DOM-Level-2-Events, 2000.
[15] C. Reis, J. Dunagan, H. Wang, O. Dubrovsky, and S. Esmeir. BrowserShield: Vulnerability-driven filtering of dynamic HTML. In Proc. OSDI, 2006.
[16] RSnake. XSS (Cross Site Scripting) cheat sheet. http://ha ckers.org/xss.html, 2006.
[17] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. ACM Transactions on Computer Systems, 2(4):277–288, Nov. 1984.
[18] Z. Su and G. Wassermann. The essence of command injection attacks in Web applications. In Proc. POPL, 2006.
[19] D. Yu, A. Chander, N. Islam, and I. Serikov. JavaScript instrumentation for browser security. In POPL, 2007

AUTHORS

Ashwani Garg received his B.tech and M.tech Degree in Computer Science and Engineering from Kurkshetra University.He is a UGC- NET June-2012 qualifier. He has publications in many different international journals. His current research interest includes Number System Conversions.

Shekhar Singh has done his M.tech degree in Computer Science and Engineering from GGIPU, Delhi.His Current research interest includes digital image processing, pattern matching, Computer vision.

