International Journal of Advance Engineering and Research Development (IJAERD)
Volume 1,Issue 3, April 2014, e-ISSN: 2348 - 4470 , print-ISSN:2348-6406

# A Review on Security Exposures on Web Application

[1]PATEL SUDHIRKUMAR JAGADISHBHAI,
MCA, Ph.D. (Pursuing) Mewar University, Chhittogarh. (Raj)
*Asst. Prof & Head - D.L.Patel Inst. of Mgt & Tech. MCA College, Himatnagar, Gujarat, India*
*Affiliated to Gujarat Technological University (G.T.U) Ahmedabad.*
patel_sudhirkumar@gtu.edu.in , sudhirkumar.patel@gmail.com

**Abstract:** Now a day, Web applications are essential, general distributed systems whose existing security relies mostly on server-side mechanisms. Web applications offer end users during client access to server functionality during a set of Web pages. These pages repeatedly enclose script code to be executed vigorously within the client Web browser. The majority Web applications intend to impose perceptive security policies, simple, such as, for Web-based email, prohibiting any scripts in untrustworthy emails. Still, Web applications are at present subject matter to a embarrassment of successful attacks, such as the recent self-propagating worms, session riding, cookie theft, browser hijacking, and cross-site scripting in Web-based email and social networking web-sites . This paper looks at five universal Web application exposures, their countermeasures and paradigms to reduce ordinary security exploits and to protect the talented class of rich, cross-domain Web applications.

## I. Introduction

The enlarged procedure of the network technology and Internet has transformed the spotlight in measuring computer atmospheres. The usual approach considered the location of data stored on the hardware first and then the hardware and tools. Also, these assessments were predominantly at explicit points in time and predominantly compliance-based reassess. Through network-based technology, the crucial unease is the contents of information or objects and on the network. Also, a review of network technology is decisive on the management and implementation of real-time reins to gather the business requirements and to suggest constant monitoring [2]. Security is not a once incident. It is inadequate to protect your code just once. A protected coding proposal must treaty with all phases of a program's lifecycle. Protected web applications are only possible when a Protected SDLC is used. Protected programs are protected by default, throughout development, also by design.

## II. Literature Review

No language can put a stop to vulnerable code, even if there are language characteristics which could help or hold back a security-conscious developer [1]. Exposed software is by now decaying our healthcare, financial, defence, energy, and other significant infrastructure. As our digital infrastructure acquires more and more interrelated and intricate the difficulty of attaining application precautions enlarges exponentially. We can no longer manage to put up with comparatively straightforward security troubles like those offered underneath. The exposures [12] described in this paper are:

1. Username enumeration
2. Arrange string exposures
3. SQL injection
4. Remote code implementation
5. Cross Site Scripting (XSS)

## III. Exposures

### 3.1 Username enumeration

Username enumeration is a kind of attack where the backend confirmation script informs the attacker if the abounding username is right or not [10, 11, 17]. Utilizing these exposures assists the attacker to

testing with changed usernames and decides suitable ones with the assist of these diverse defect messages.
Rating: Fewer Crucial
In case of login screen, underneath the response agreed when a suitable username is estimated accurately. Username enumeration can assist attackers who attempt to use a few slight usernames with effortlessly guessable passwords, such as admin/admin, test/test, guest/guest, and so on. These accounts are frequently shaped by developers for trying intentions, and a lot of times the accounts are certainly not stopped or the developer not remembers to alter the password. Unnecessary to say, these could be vital particulars for public engineering attacks.

**Preventive:**
Display reliable defect messages to stop revelation of suitable usernames. Make sure if unimportant accounts have been shaped for testing reasons that their passwords are either these accounts are completely separate after testing is ended or not trifling - and ahead of the application is set online.

### 3.2 Arrange String Exposures
This exposure marks from the utilize of unfiltered user input as the arrangement string parameter in assured C or Perl functions that execute formatting, such as C's printf(). A wicked user may apply the %x and %s format tokens, along with others, to print data from the probably or stack other locations in recall [4, 8]. One might be also write randomly data to randomly locations via the %n arrange token, which orders printf() and related functions to write down rear the number of bytes formatted. This is presumptuous that the equivalent argument survives and is of type int *. Format string exposures assaults fall into three common groups: writing, reading and denial of service [14,15].

Rating: Sensible to Extremely Crucial

Now is the part of code in miniserv.pl which was the cause of exposures in Webmin:

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

In this paradigm, the client abounding data is inside the format requirement of the syslog call. The vectors for a easy Denial of Service - DoS of the Web server are to utilize the %0 and %n (large number) contained by the username, with the previous grounds a write security fault inside Perl – and most important to script abortion. The last causes a big quantity of memory to be owed within of the perl practice.

**Preventive:**
Amend the source code thus that the input is accurately confirmed.

### 3.3 SQL Injection
SQL injection is a dreadfully older approach but it's unmoving trendy surrounded by attackers. This procedure permits an attacker to regain essential information from a Web server's database [18]. Depending on the application's sanctuary process, the brunt of this assault can differ from essential information revelation to remote code implementation and total system negotiation [3, 5]. Evaluation: Judicious to Extremely Crucial at this time, it is an example of exposure code in which the user-supplied effort is frankly used in a SQL query:

- `<form action="xyz.php" method="POST" />`

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

The librettos will exertion usually when the username doesn't enclose any wicked characters. In former style, when yielding a non-wicked username (abc) the query suits: $query = "SELECT * FROM users WHERE username = 'abc'"; However, a wicked SQL injection query will outcome in the subsequent effort: $query = "SELECT * FROM users WHERE username = '' or '1=1'"; As the "or" clause is constantly true, the mysql_query function returns records as of the database. A parallel case, using AND & a SQL command to produce an explicit defect note.

It is observable that these defect notes assist an attacker to obtain a grasp of the information which they are appearing for (such as the table name, password hashes, usernames, database name etc). Thus showing adapted defect messages may be an excellent workaround for this crisis, though, there is an extra attack method recognized as Blind SQL Injection where the attacker is still capable to achieve a SQL injection still when the application does not divulge any database server defect message including valuable information for the attacker.

**Preventive:**

1. Evade concerning to the database as the database owner or as an admin. All the time utilize adapted database users with the exposed least mandatory privileges mandatory to complete the consigned duty.

2. PHP has two functions for MySQL that disinfect user input: add slashes (an elder approach) and mysql_real_escape_string. This function arrive from PHP >= 4.3.0, so we should ensure initial if this function subsists and that we're running the latest version of PHP 5 or 4. MySQL_real_escape_string prepends back-slashes to the subsequent characters: \n,\x00, \, \r, ', \x1a and ".

If the PHP magic_quotes_gpc function is on, then all the GET, POST, COOKIE data is runaway robotically.

## 3.4 Remote code implementation

As the name recommends, this exposures permits an attacker to run randomly, system altitude code on the susceptible server and salvage any required information enclosed therein. Improper coding mistakes lead to these exposures [6, 7]. Sometimes, it is complicated to determine these exposures through dispersion testing assignments except such troubles are repetitively exposed while doing a source code reassess. However, when testing Web applications is vital to memorize that utilization of these exposures can escort to entirety system concession with the similar rights as the Web server itself [13].

Rating: Extremely Crucial

Now we will stare such kind of crucial exposures:

Developing register_globals in PHP: Register_globals is a PHP scenery that organizes the accessibility of "superglobal" variables in a PHP script (for example data from cookies or URL-encoded data, data posted from a user's form). In advance liberates of PHP, register_globals was put to "on" by evade, which prepared a developer's life easier - but this escort to less sheltered coding and was extensively oppressed. When register_globals is located to "on" in php.ini, it can permit a user to initialize quite a few formerly uninitialized variables tenuously. Many a times an uninitialized parameter is used to contain redundant files from an attacker, and this could escort to the execution of random files from remote /local positions. For example:

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

Here if the $page factor is not initialized and if register_globals is put to "on," the server will be susceptible to isolated code functioning by counting any random file in the $page parameter. Now let's appear at the utilize cipher:

```
http://www.expsite.com/index.php?page=http://www.enemy.com/enemy.txt
```

In this manner, the file "http://www.enemy.com/enemy.txt" will be integrated and performed on the server. It is an especially uncomplicated but efficient attack.

**Preventive:**

1. Latest PHP editions have register_globals put to off by evade; still several users will alter the evade setting for applications that oblige it. This register can be put to "on" or "off" either in a php.ini file or in an .htaccess file. The uneven should be accurately initialized if this register is set to "on." Proprietors who are hesitant should enquiry application developers who persist on by register_globals.

2. It is a complete must to disinfect all users input prior to dispensation it. So far as probable, keep away from using shell instructions. However, if they are necessary, certify that only sieved data is used to construct the string to be performed and take care to break away as of the output.

### 3.5 Cross Site Scripting(XSS)

The sensations of this attack necessitate the victim to perform a wicked URL which might be skilled in such a way to emerge to be genuine at first appear. While visiting such a skilled URL, an attacker can successfully carry out a little wicked in the victim's browser [9, 16]. Several wicked Javascript, for pattern, will be run in the perspective of the web site which possesses the XSS bug [19]. Rating: Less to fairly crucial, here is a example section of code which is exposures to XSS attack:

-
  ```
  <form action="find.php" method="GET" />
          Welcome!! <p>Enter your name: <input type="text" name="n1" /><br />
                      <input type="submit" value="Go" /></p><br>
  </form>
  <?php echo "<p>Your Name <br />"; echo ($_GET[n1]); ?>
  ```

In this paradigm, the value accepted to the variable 'n1' is not sanitary prior to loud it rear to the user. This can be oppressed to carry out any random script. Now are several examples utilizing code:

```
http://victim_site/clean.php?n1=<script>code</script>
or
http://victim_site/clean.php?n1=<script>alert(document.cookie);</script>
```

**Preventive:**
The exceeding code can be shortened in the subsequent approach to keep away from XSS attacks:

- <?php $html= htmlentities($_GET['n1'],ENT_QUOTES, 'UTF-8'); echo "<p>Your Name<br />"; echo ($html); ?>

## IV. CONCLUSIONS

Web applications accomplish out to a big, less-confidential user base than inheritance client-server applications, and so far they are more exposures to attacks. A lot of companies are preliminary to take proposals to check these types of break-ins. Code reconsiders, wide diffusion testing, and interruption exposure systems are just a few traditions that companies are combating a rising problem. Regrettably, most of the explanations obtainable today are using harmful security logic. Harmful security logic explanations can avoid known, widespread attacks, but are unsuccessful beside the kind of embattled, wicked hacker movement sketched in this paper. In this paper, we have confirmed five ordinary web application exposures, their preventives and their criticality. If there is a reliable message with each of these attacks, the type to alleviate these exposures is to disinfect user's input ahead of dispensation 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