Investigating Options of Securing Web Application

T.Meemeskul and P.S.Dowland

Network Research Group, University of Plymouth, Plymouth, United Kingdom
e-mail: info@cscan.org

Abstract

This research paper represents and explains the web application's major vulnerability as well as displaying easy functions to secure web applications. In the background it shows the brief internet threat report. The approach is based on the assumption of "Network Good – Application Bad". The Results of test show the reason why security options are not able to secure web applications.

Keywords

Security Awareness, Vulnerability, Web Attack, Security Option

1 Introduction

Web technology has enabled a new direct channel for communicating between client and server over Hyper Text Transfer Protocol (HTTP). Many visionary businesses take advantage of web technology to expand their network to users. It provides more alternative choice for developers selecting technology to operate their computer systems. In the same way, it motivates attackers to keep an eye on new web technologies and investigate loopholes of the application that enables port 80 to run all the time. Vulnerability begins with several developers, those who are inexperienced and have a limited knowledge of web technologies leaving them to remain unaware of security threats, more and more achieve various solutions to implement web systems but in limited time only. Without the principles of web security, the appearing gateway from failure in configuration and programming attracts attackers to break the web system. In addition, the limits of web technology itself raise the risks, the attackers approach is also more various than the protecting function.

Security options are designed to protect and defend as safeguard from unauthorised access which intends to break a web system. Many software vendors release security options for supporting unaware web developers, web administrators and web clients who are the key aspect involved with web security. However, security options unpreserved all the characteristics of web attacks even web applications and web servers run behind a firewall. HTTP port 80 is still enabling any HTTP request from web clients. The web application source code which interfaces with the web server and back-end database is available to be a device for attackers breaking into the web system. The best options of securing web application has become security awareness from web developers, web administrators and web clients

2 Background

With web technologies, the communication service to the web is established when user's mention input into the Uniform Resource Locator (URL) in the Web browser, intern Web browsers running on the Application Layer open HTTP port 80, to send HTTP requests to web servers. User's input transforms to be a data packet to pass through the Transport Layer, Network Layer and Data Link Layer. The packet which runs inside a communication cable would be sent to the web server as mentioned in the URL. When web servers receive HTTP request, it would transmit them to web applications for executing data. Web applications would query data in database and return it back to the web server. Web servers would send the response to the users request back as hypertext or plain ASCII text and it would then terminate the connection after it receives an acknowledgement packet from the web client or after a server timeout due to 15 seconds without activity. If the connection is broken while the web server is responding, web servers would contribute an error message as text in HTML syntax (Cisco 2005).

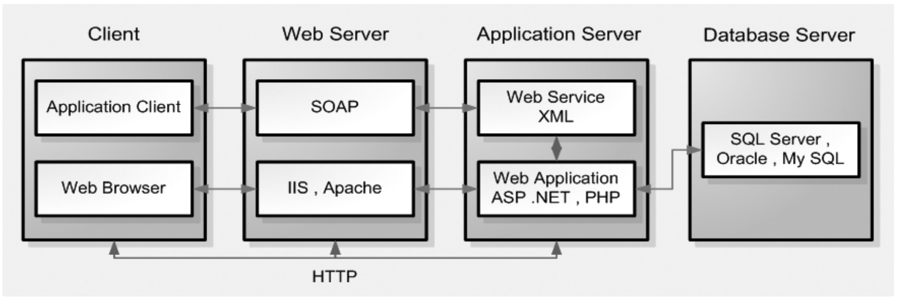
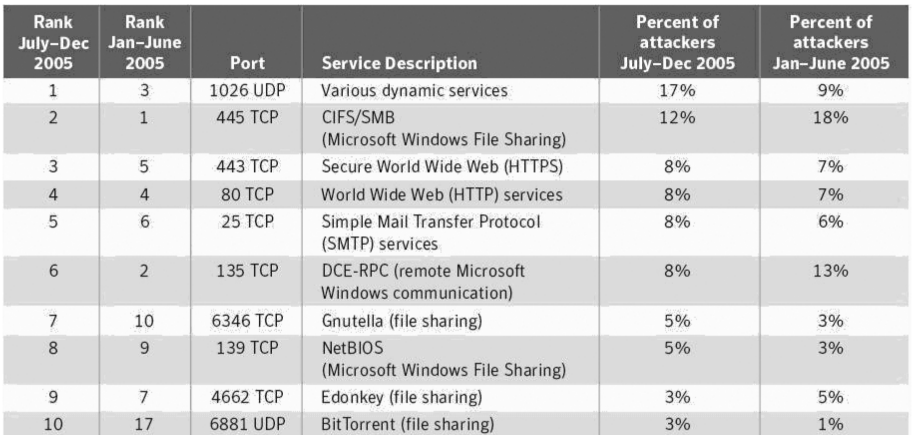


Figure 1: Web Mechanism

Now, many options are available for securing the connection between HTTP servers and the web clients. Secure Socket Layer (SSL) released to protect the private connection between web client and web servers by the use of cryptography protocol. In addition, the release of web application firewalls to also support HTTP servers for filtering HTTP request and controlling Buffer Overflow. In the same way, web developing tools released validation tools for validating input into the web application. However, more concentration from web developers focuses on other points rather than source code.

Basically, the major vulnerability of web applications is Invalidated Input (Huseby 2004). It is a vulnerability that web developers remain unaware about due to difficulty in determining input scope. Attackers are able to use input which affects the code in a web application attack of web system. This is because web developers do not determine the scope of input received from users. Attack can investigate loopholes in source code and submit input which affects the commands in the source code to obtain data back. It is possible for this vulnerability to occur with all the web pages who provide textbox or textarea to receive data from users. Invalidated Input relates to forced browsing, command insertion, cross site scripting, buffer overflows, format string attacks, SQL injection, cookie poisoning, and hidden field manipulation vulnerability.



Rank July–Dec 2005	Rank Jan–June 2005	Port	Service Description	Percent of attackers July–Dec 2005	Percent of attackers Jan–June 2005
1	3	1026 UDP	Various dynamic services	17%	9%
2	1	445 TCP	CIFS/SMB (Microsoft Windows File Sharing)	12%	18%
3	5	443 TCP	Secure World Wide Web (HTTPS)	8%	7%
4	4	80 TCP	World Wide Web (HTTP) services	8%	7%
5	6	25 TCP	Simple Mail Transfer Protocol (SMTP) services	8%	6%
6	2	135 TCP	DCE-RPC (remote Microsoft Windows communication)	8%	13%
7	10	6346 TCP	Gnutella (file sharing)	5%	3%
8	9	139 TCP	NetBIOS (Microsoft Windows File Sharing)	5%	3%
9	7	4662 TCP	Edonkey (file sharing)	3%	5%
10	17	6881 UDP	BitTorrent (file sharing)	3%	1%

Figure 2: Top Attack Port from Symantec Internet Threat Report 2005

In the Internet Threat Report 2005 from Symantec represent HTTPS port 443 is ranked number 3 and HTTP port 80 is ranked number 4. However, the percentage of attacks for them are equal, they had 8% of overall attacks each (Symantec 2006). The result is possible to put into 2 points of view; Firstly, SSL is not secure, it might use self sign digital certificates and attackers can use sniffer software, to find the certificate code. Therefore, attackers are able to use man in the middle to attack web sites. Secondly SSL and networks are secure but web applications contain vulnerabilities. In this case the open loophole for attackers to use is Invalidated Input to attack the web.

3 Methodology

The vulnerability in the web application occurs because web developers are concern with the security of the connection between HTTP server and web client rather than source code. The vulnerability that's contained on web servers is difficult to solve. It is a big project that can be divided into many web developers programs. If only one web developer is unaware about security and leaves even one line of code that contains vulnerabilities, it means attackers are able to attack the whole web system. For web developers, they should be concerned with their source code before securing other applications or networks.

3.1 Approach

This research test is based on the assumption of “Network Good; Application Bad”. The test would implement web systems by installing Apache HTTP server in Linux Fedora, all data would be stored in MySQL database connected with a PHP web application. All connection ports were closed by firewall except HTTP port 80 and HTTPS port 443. The authentication web page runs on HTTPS that uses OpenSSL certificate.

```
<?php
$con = mysql_connect("localhost","root","Timmy_21");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
$user=$_POST['uname'];
$password=$_POST['password'];
$link = mysql_connect("localhost", "root", "Timmy_21");
mysql_select_db("timdb", $link);
$result = mysql_query("SELECT username,password FROM timusers
WHERE username='$user' and password ='$password'", $link);
$num_rows = mysql_num_rows($result);
echo "$num_rows Rows\n";
if ($num_rows == 0 )
   echo "Please enter the right password";
else
   echo "Found in database";
mysql_close($con);
?>
```

For the test, after entering input into the login form, form data uses the POST method to send data passed to the standard stream which is the connection for transferring input or output between computer applications. Form data would transform to $user variable, $password variable before sending variables into the SQL query command. If the data matches the data in the database, the web page would represent the number of rows which are matching. If the data is not matching, the result would be represented to users as “Please enter the right password”.

The first test is sending the right username and password in there, the results found are normal. The Second time test is sending SQL code rather than username and password. The code is ‘or ‘1’ = ‘1. The results are in the following section.

4 Experimental result and Discussion

The results from previous section represent that the SQL code which entered was found in database. The SQL query in the source code executes input to be like this:

```
SELECT username,password FROM timusers WHERE Admin =  '  '  or
'1' = '1' and Timmy = '  '  or '1' = '1'
```

Actually, the database stores the username “Admin” and password “Timmy”. This code is following the tutorial online and some books. It means many authors aimed to teach new web developers how to understand to use the command only. They are not supporting any principle of security to web developers, not even easy functions such as preg_match. In PHP, function preg_match is used to compare matching input that passes to check it in function preg_match (Zandstra 2002). For example with the code below, before sending the code to query in the database, there should be input filtering such as character querying, for characters that possibly affect the source code.

```
<?php
function validateinput($uname2,$password2)
{
       if   (preg_match("/^[A-Za-z0-9_.=+-]+@([a-z0-9-]+\.)+([a-
z]{2,6})$/",$uname2))
       {
              if           (preg_match("/[A-Za-z0-9!@#%&*)(}{_-
]/",$password2))
              {
                     $uname3=addslashes($uname2);
                     $password3=addslashes($password2);
                     $link = mysql_connect("localhost", "root",
"Timmy_21");
                     mysql_select_db("timdb", $link);
                     $result       =        mysql_query("SELECT
username,password  FROM  timusers  WHERE  username='$uname3'  and
password ='$password3'", $link);
                     $num_rows = mysql_num_rows($result);
                     if ($num_rows == 0 )
                            header("Location: 404.html");
                     else
                            header("Location:
http://www.plymouth.ac.uk");
                     mysql_close($con);
              }
       }
       else
       {
              header("Location: 404.html");
              die();
       }
}
$uname2=$_POST['uname'];
$password2=$_POST['password'];
validateinput($uname2,$password2);
?>
```

In the function preg match, it requires an email for the username. If the username those users enter is not an email, the web site would redirect users to an error page provided. Basically, the source code above uses double filter input, after passing function preg_match, it uses function addslashes before transforming user’s data into

a variable for querying. Better ways could web developers not sending any input to query in the SQL query command because most of attackers are able to break the basic code of SQL. Web developers should transform data that's queried in the database into variables and compare them with user's input rather than send user's input to query the data.
ASP.NET, Visual Studio 2005 provides a validation tool to filter the user's input (Ladka 2002). The RegularExpressionValidator from ASP.NET is able to validate input before submitting data to process. It works with object that web developers require to validate input. This method is better than the function from PHP because it is an automatic tool. For PHP, web developers still have to program it by hand.

```
<asp:RegularExpressionValidator
ID="RegularExpressionValidator1" runat="server"
ErrorMessage="Invalid Username"
Style="z-index: 101; left: 252px; position: absolute; top:
16px" ControlToValidate="txtUsername"
ValidationExpression='^[A-Za-z0-9_.=+-]+@([a-z0-9-]+\.)+([a-
z]{2,6})$'></asp:RegularExpressionValidator>
<asp:RegularExpressionValidator
ID="RegularExpressionValidator2" runat="server"
ErrorMessage="Invalid password"
Style="z-index: 100; left: 252px; position: absolute; top:
61px" ValidationExpression="(?!^[0-9]*$)(?!^[a-zA-Z]*$)^([a-zA-
Z0-9]{8,10})$"
ControlToValidate="txtPassword"></asp:RegularExpressionValidato
r>
```

The code above shows that in the txtUsername and txtPassword strict the input. The input for txtUsername should be an email only. For the txtPassword, it must be between 8 and 10 characters, contain at least one digit and one alphabetic character, and must not contain special characters. That is the recommendation from Microsoft. However, very strong passwords should include special character because they are not query character. The advantage of determining input is not only securing the web but when the web was attacked, it is easy to investigate the problem. Web developers are able to cut out the problems from input and find problems in other resources.

5 Conclusion

Obviously, all web vulnerabilities that occur are involved with security awareness. Many web developers misunderstand that options could secure their application even if it contains vulnerabilities. They are concerned more about options to secure their mistakes. They are not aware enough to improve their codes while programming. Actually, the first option to secure web applications is security awareness from web developers, and it's the best option and where they should begin. This research tries to investigate many options for securing web applications but the answer shows problems occur while programming web pages for testing. Security awareness from all the entities involved with the web application is the best option found from this research. To improve web security for future, it should start now. The first tutorial of web developers should be basic concepts of web security rather than how to use connection string to connect with the database. For users, before teaching them to know how to use a search engine, change the ideas to how to set a strong password. In the authentication page they should provide a link for teaching users to set a strong password. That is the future plan for reducing the percentage of web attacks.

6 References

Cisco (2005), Cisco *Network Academy Program CCNA 1 and 2 Companion Guide*, Cisco Press, USA, ISBN 1-58713-150-1.

Huseby, S.(2004), *Innocent Code*, Wiley, England, ISBN 0-470-85744-7

Ladka, P. (2002), *ASP .NET for Web Designer*, New Riders, USA, ISBN 0-7357-1262-X.

Symantec (2006), 'Internet Threat Report July –December 2005', available from: http://www.symantec.com/enterprise/threatreport/index.jsp. date visited: 31 July 2006

Zandstra, M. (2002), *Tech yourself PHP in 24 hours*, SAMS, USA, ISBN 0-672-32311-7.