CYBER SECURITY INTERNSHIP - TASK 1 REPORT

Task Title: Web Application Security Testing

Track code: FUTURE_CS_01

Intern Name: Sarita Sharma

Aim:

Perform an in-depth evaluation of web application security using the Damn Vulnerable Web Application (DVWA) in a secure lab setting. The aim is to explore and demonstrate various security flaws including SQL injection, cross-site scripting(XSS) and authentication loopholes via brute force techniques. This testing will help assess the potential consequences of these vulnerabilities and recommend effective strategies to enhance web application protection.

DVWA SQL Injection Vulnerability Exploitation

Target Environment:

- **Application:** DVWA (Damn Vulnerable Web Application)
- **Security Level:** Low
- **SQL Database:** MySQL
- **Module:** SQL Injection

Objectives:

To demonstrate and document SQL Injection attacks on a vulnerable input field in DVWA to:

1. Retrieve user data.
2. List database tables.
3. Enumerate table columns.

Setting Up DVWA in Kali Linux

Requirements:

- Apache web server
- MySQL/Maria DB
- PHP
- DVWA source code (from GitHub)

Login to DVWA

- URL: http://localhost/DVWA/login.ph
  Default credentials:
- **Username:** admin
- **Password:** password

DVWA is ready!

You can now start testing vulnerabilities like:

- SQL Injection
- XSS
- CSRF

Basic SQL Injection Test

- **Input:**

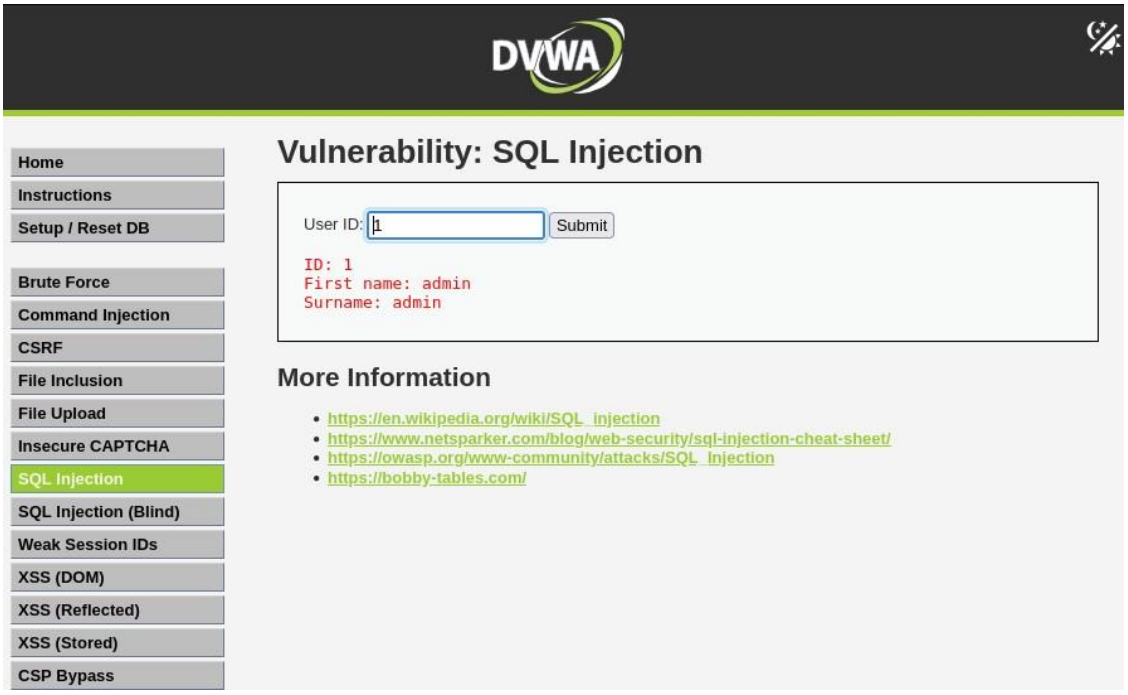
  1

- **Result:**
  Displays record for `ID: 1`:

  First name: admin
  Surname: admin

- **Observation:**
  The backend executes:

  SELECT * FROM users WHERE id = '1'
  This confirms the input is being inserted directly into a SQL query without sanitization.



Screenshot: SqlIn_1.png

Bypass Authentication & Dump All Users

- **Input:**

1' or '1'='1

- **Result:**

Dumps all users from the database:

Admin, Gordon Brown, Hack Me, Pablo Picasso, Bob Smith

- **Underlying Query:**

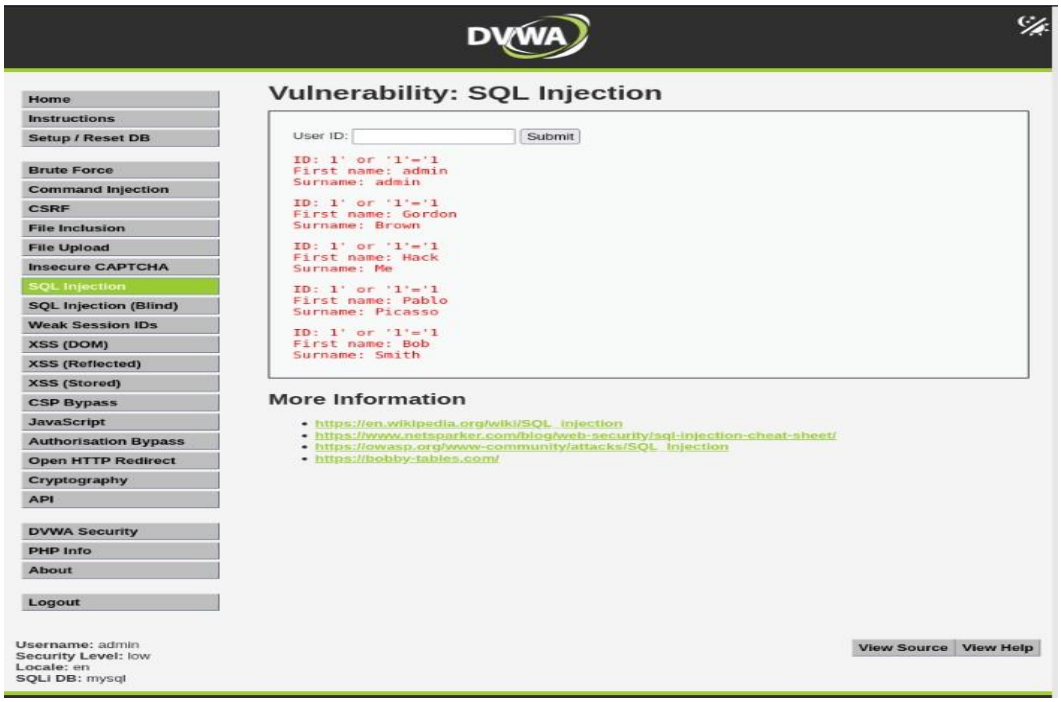
SELECT * FROM users WHERE id = '1' OR '1'='1'

- **Impact:**

  •Authentication bypass

  • Full table enumeration

  •Critical breach of confidentiality



Screenshot: SqlIn_2.png

Extract Table Names

- **Input:**

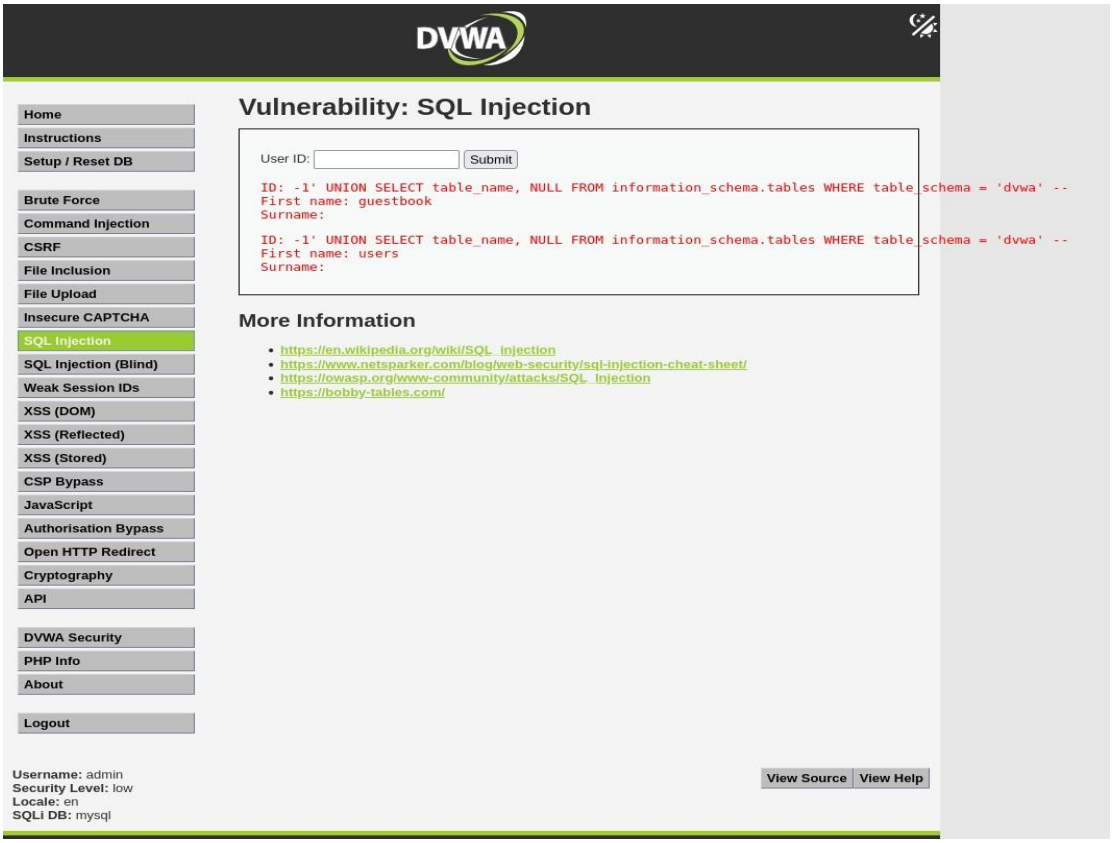
-1' UNION SELECT table_name, NULL FROM information_schema.tables WHERE table_schema = 'dvwa' –

- **Result:**

  Reveals the following tables:
  - guestbook
  - users

- **Impact:**

  - Database schema enumeration
  - Attackers now know which tables to target next



Screenshot: SqlIn_3.png

Extract Column Names from users Table

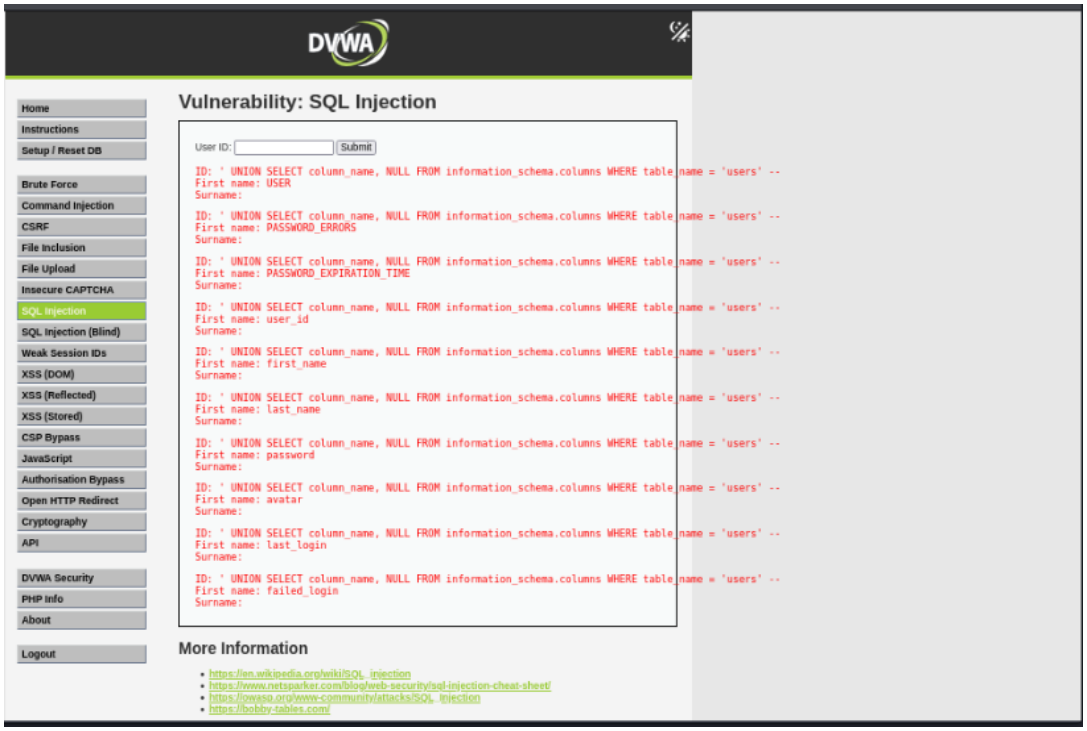
- **Input:**

  -1' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name = 'users' –

- **Result:**
  Enumerates column names:
  - user_id
  - first_name
  - Sur_name

- **Impact:**

  Full mapping of user table columns, including sensitive fields like:
  - First name of all the users

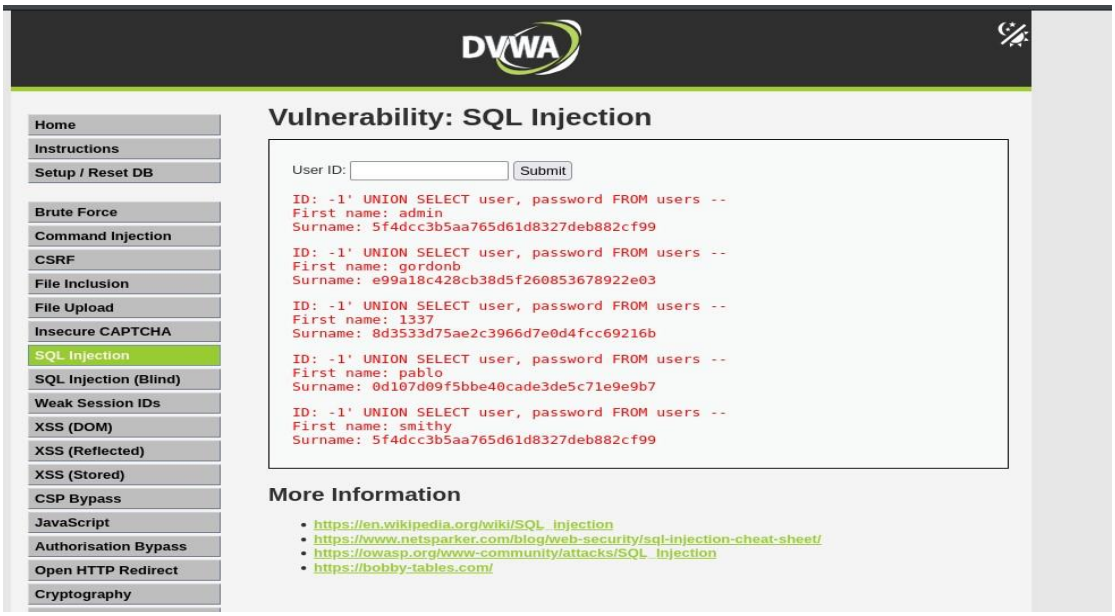


Screenshot:SqlIn_4.png

Extract Usernames and Passwords from users Table

- **Input:**

  -1' UNION SELECT user, password FROM users –



Screenshot:SqlIn_5.png

- **Result:**

  Enumerates sensitive user credentials:

  admin | 5f4dcc3b5aa765d61d8327deb882cf99
  gordonb | e99a18c428cb38d5f260853678922e03
  pablo | 0d107d09f5bbe40cade3de5c71e9e9b7

- **Impact:**

  Full compromise of authentication data:

  - Reveals usernames of all registered users
  - Leaks MD5-hashed passwords, which can be cracked easily o Enables unauthorized login and privilege escalation (e.g., admin access)

What was done: Data extraction techniques involved crafting multiple SQL injection queries to uncover hidden database structures, including identifying tables and their fields, as well as retrieving critical records like usernames and encrypted passwords.

Conclusion: Exploiting the SQL injection flaw made it possible to retrieve confidential user details and internal database structure. This exposure indicates that the application lacks proper input validation and fails to implement secure coding practices like parameterized statements.

DVWA XSS Vulnerability Testing

Overview:

This report outlines the identification and exploitation of three types of Cross-Site Scripting (XSS) vulnerabilities using DVWA:

- Reflected XSS
- Stored XSS
- DOM-Based XSS

---

1. Reflected XSS:

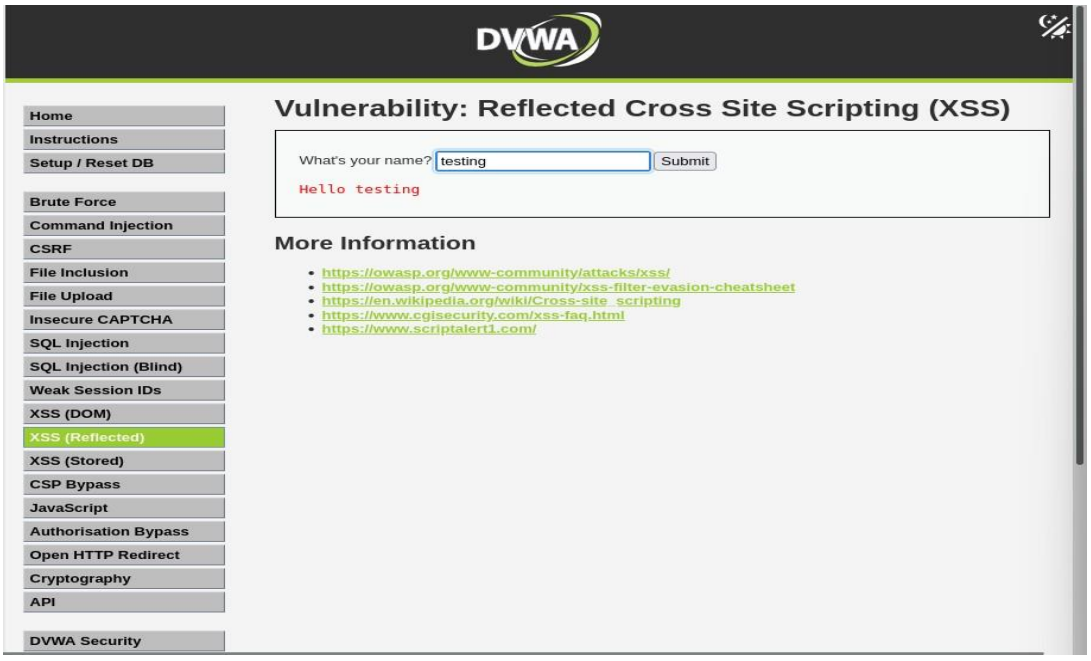
Description:
Reflected XSS occurs when user input is reflected immediately in the page response without proper validation or encoding.

Steps Taken:

- Navigated to the "XSS (Reflected)" module.
- Entered the string `testing` to test input reflection.
- Injected payload: `<script>alert('XSS')</script>`

Result:

- Payload executed immediately in the browser.



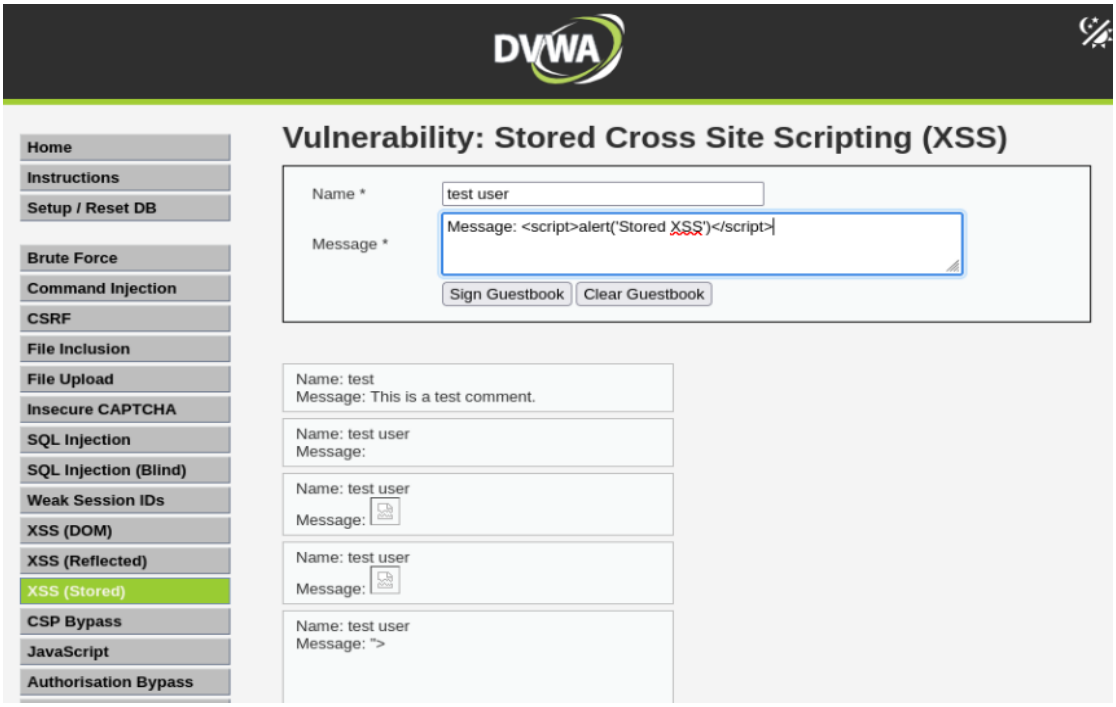
Screenshot: XSSr_1.png

2. Stored XSS

Description:
Stored XSS involves injecting a malicious script that gets saved on the server (e.g., in a database) and executes whenever the data is viewed.

Steps Taken:

- Navigated to "XSS (Stored)" module.
- Input Name: `test user`
- Input Message: `<script>alert('Stored XSS')</script>`

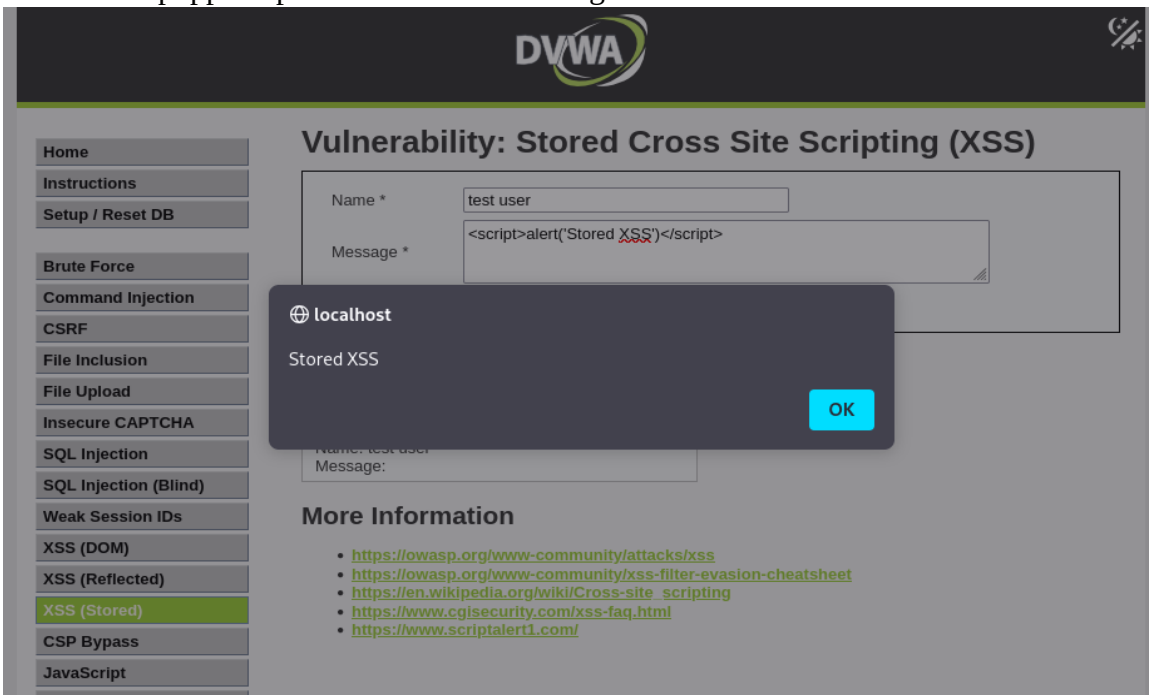


Screenshot: XSSs_1.png

- Submitted the form.

Result:

- Alert box popped up when the stored message was rendered.



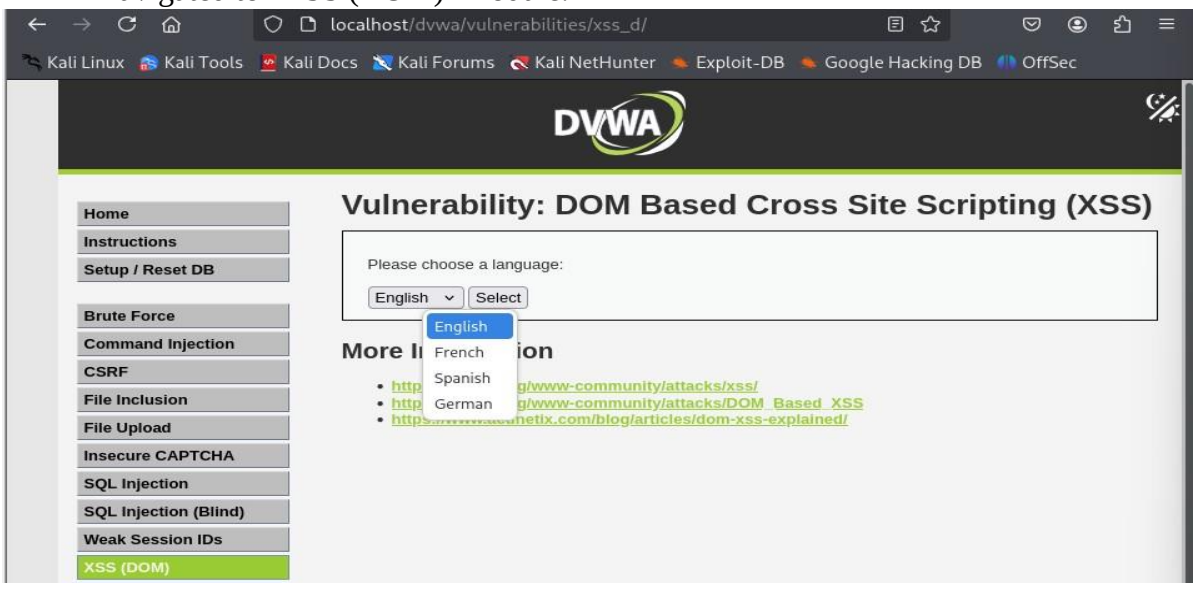
Screenshot: XSSs_2.png

3. DOM-Based XSS

Description:
DOM-Based XSS occurs on the client-side where JavaScript manipulates the DOM using untrusted data (e.g., URL parameters).

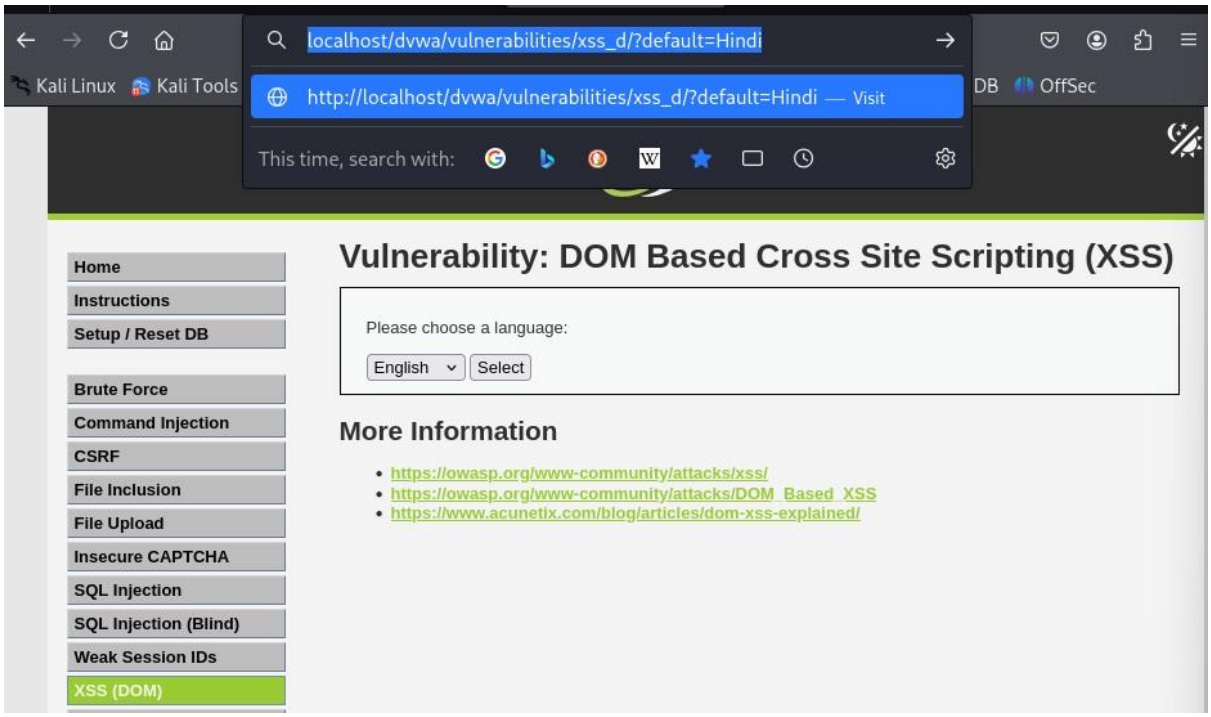
Steps Taken:

- Navigated to "XSS (DOM)" module.



Screenshot: XSSd_1.png

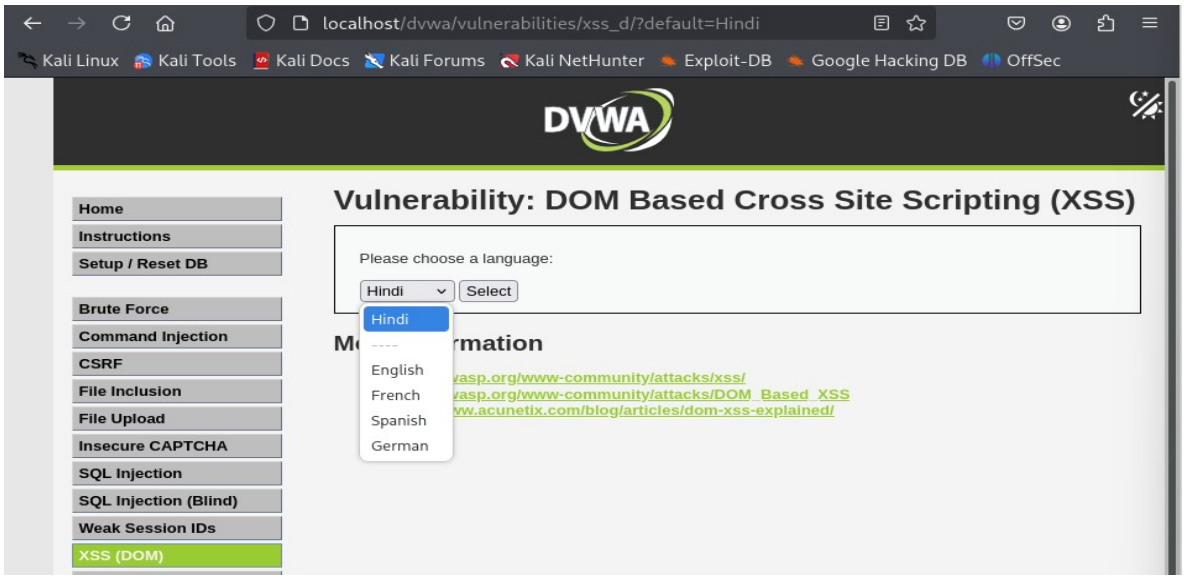
- Modified URL to: ?default=Hindi and observed behavior.



Screenshot: XSSd_2.png

Result:

- JavaScript executed due to lack of sanitization in JavaScript code.



Screenshot: XSSd_3.png

Summary Table:

XSS Type	Stored on Server	Trigger Location	Risk Level	Example Field
Reflected XSS	No	Immediate response	Medium	Search, Forms
Stored XSS	Yes	On viewing saved data	High	Comments, Messages
DOM-Based XSS	No	Client-side script code	High	URL Parameters

Conclusion:

Each type of XSS attack was successfully carried out, showing how inadequate input validation and missing output encoding can cause serious vulnerabilities on the client side.

Web Application Security Assessment

Environment: Kali Linux, DVWA (Difficulty: Impossible), Sample Web Application

---

Brute Force Attack Simulation on DVWA using Burp Suite

Objective:
The goal is to replicate a brute force attack on a vulnerable web application to learn how attackers take advantage of weak login protections by using Burp Suite's Intruder tool.

Tools Used:

- Burp Suite
- DVWA (Damn Vulnerable Web Application)
- Kali Linux
- Custom Password Wordlist

Procedure:

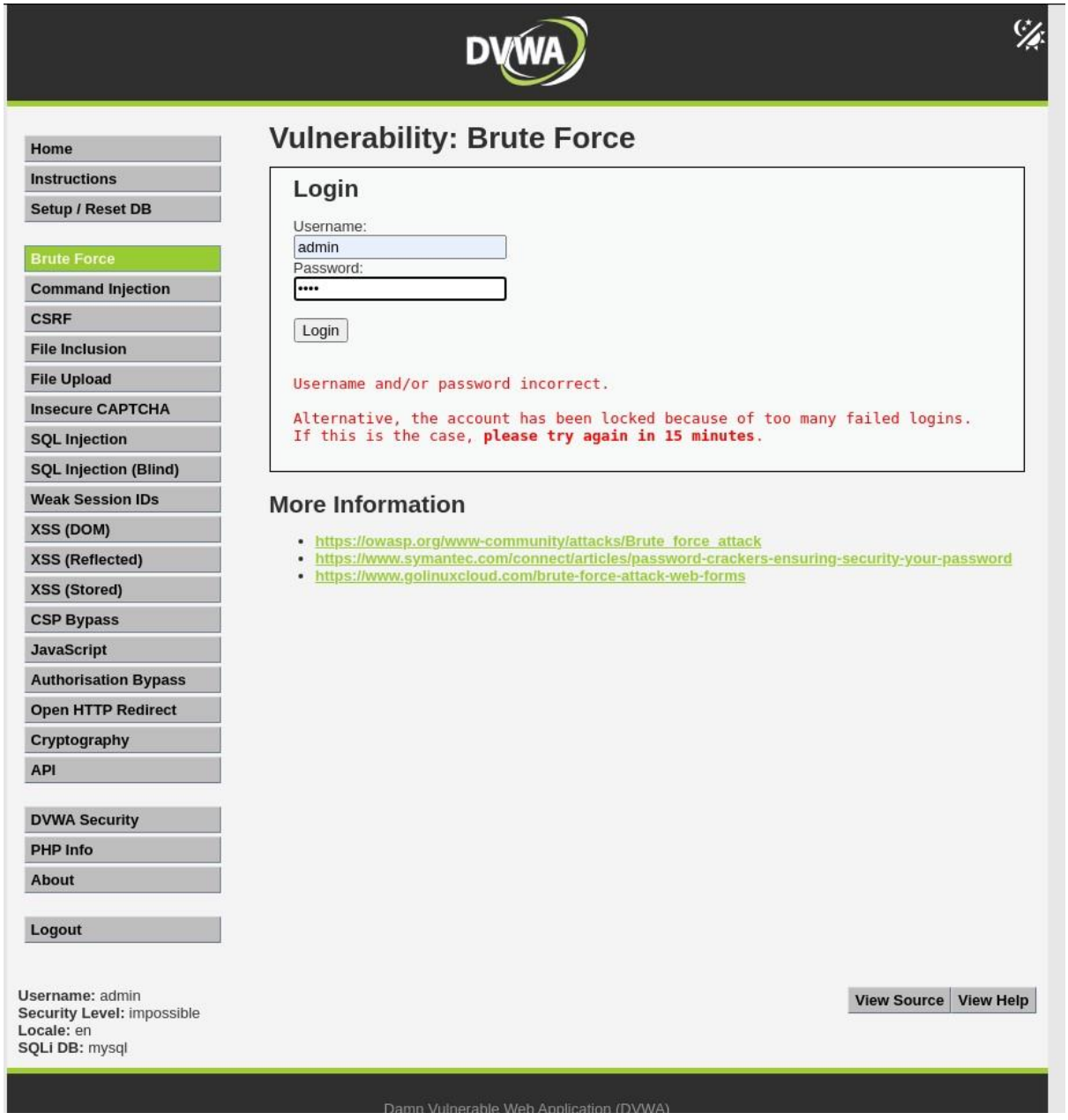
1. Setup:

- DVWA configured to 'Impossible' difficulty level
- Logged into DVWA with valid admin credentials

Screenshot: Brfo_1.png

2. Target Identification:

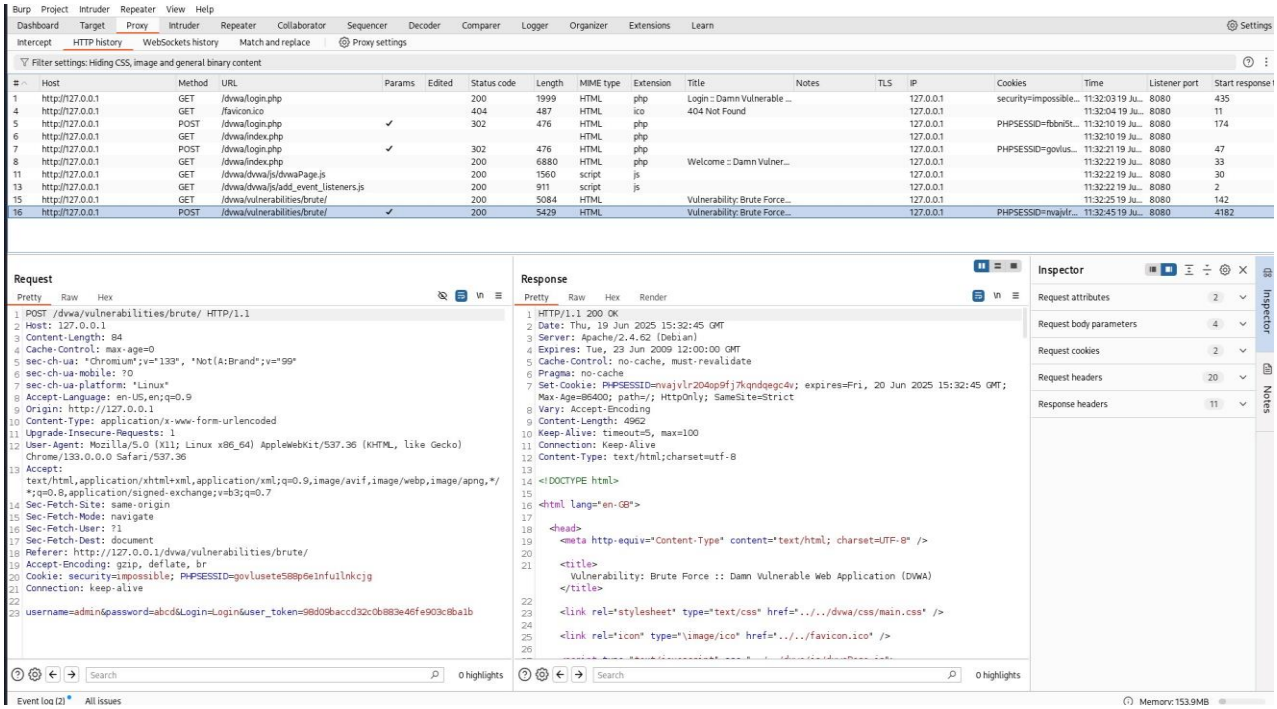
- Navigated to the "Brute Force" module in DVWA
- Attempted a failed login to capture the POST request



Screenshot: Brfo_2.png

3. Intercept & Analyze Request:

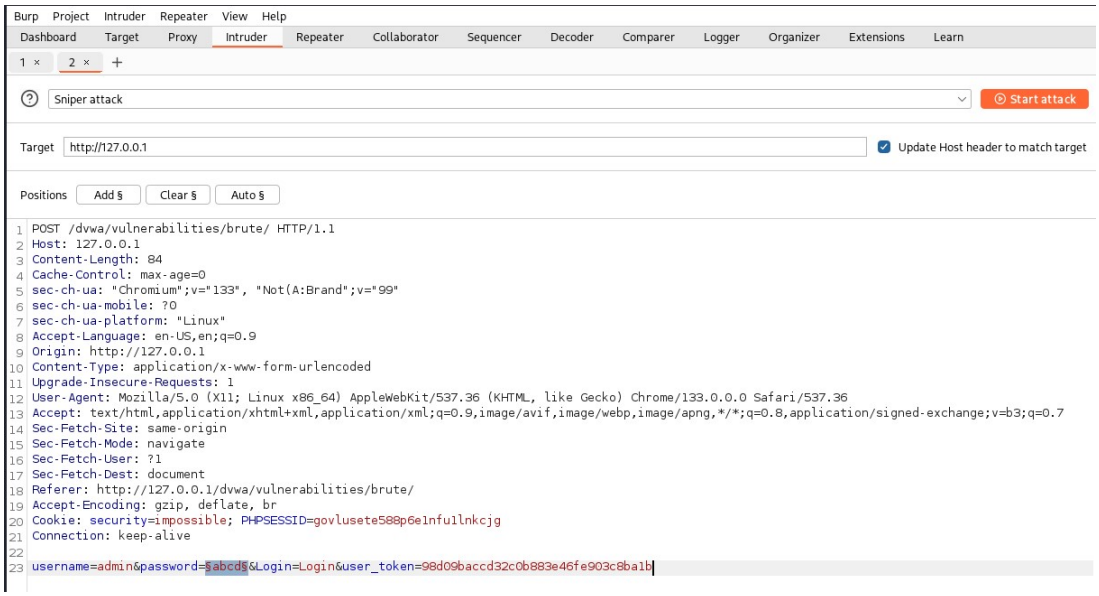
- Opened Burp Suite and enabled the Intercept option
- Captured the login POST request and sent it to Intruder



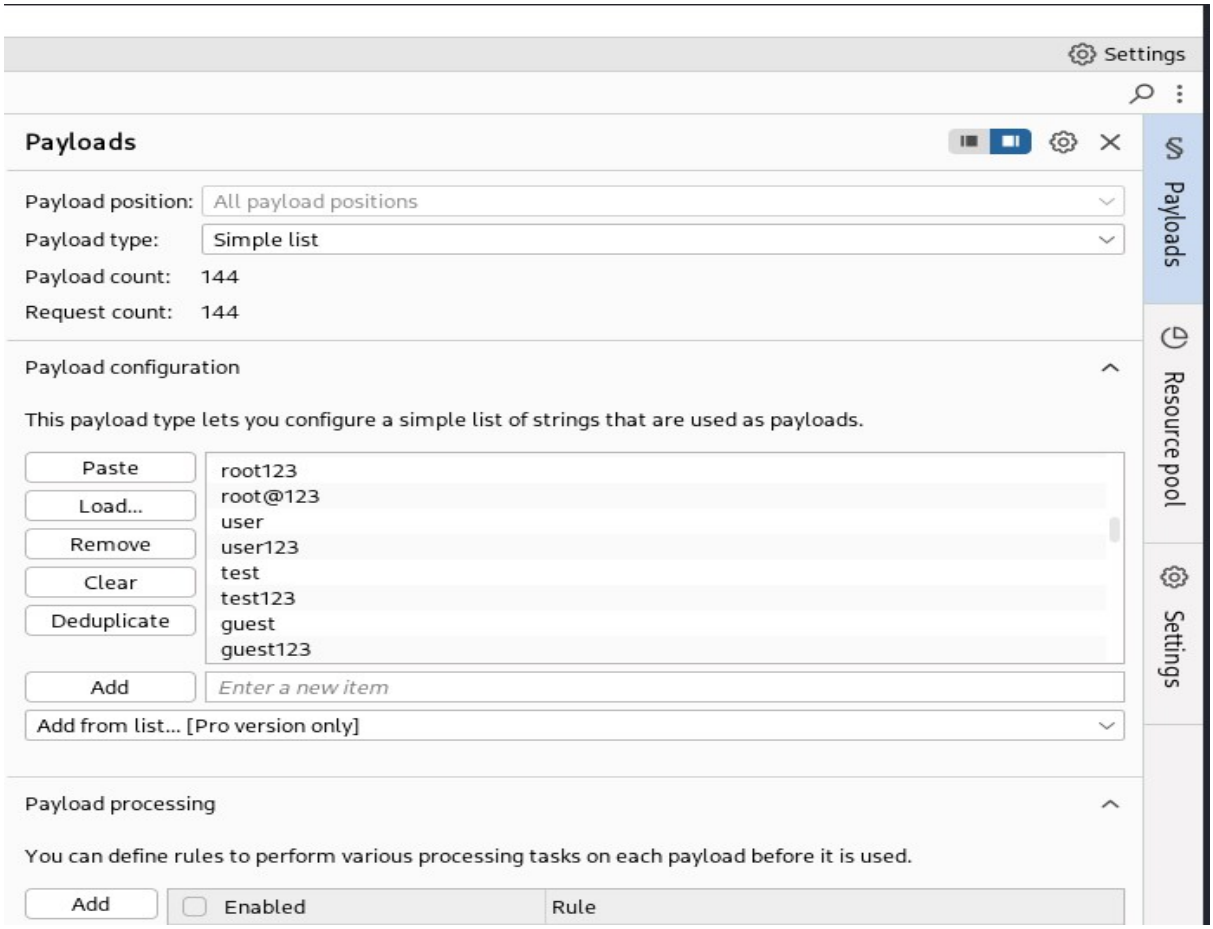
Screenshot: Brfo_3.png

4. Configure Burp Intruder:

- Set payload position for the password field



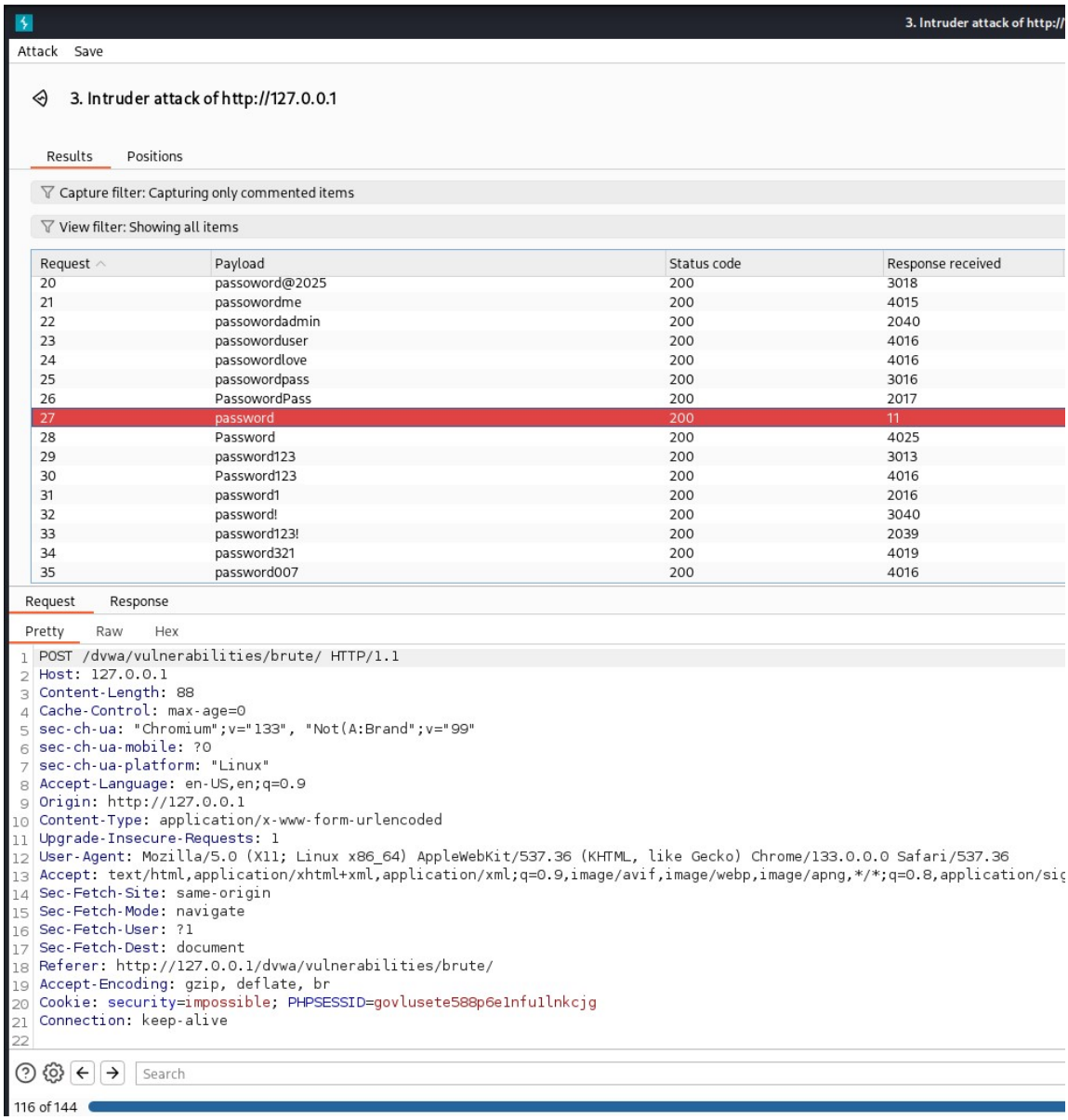
- Loaded a custom wordlist of potential passwords



Screenshot: Brfo_4.png

5. Execute Attack:

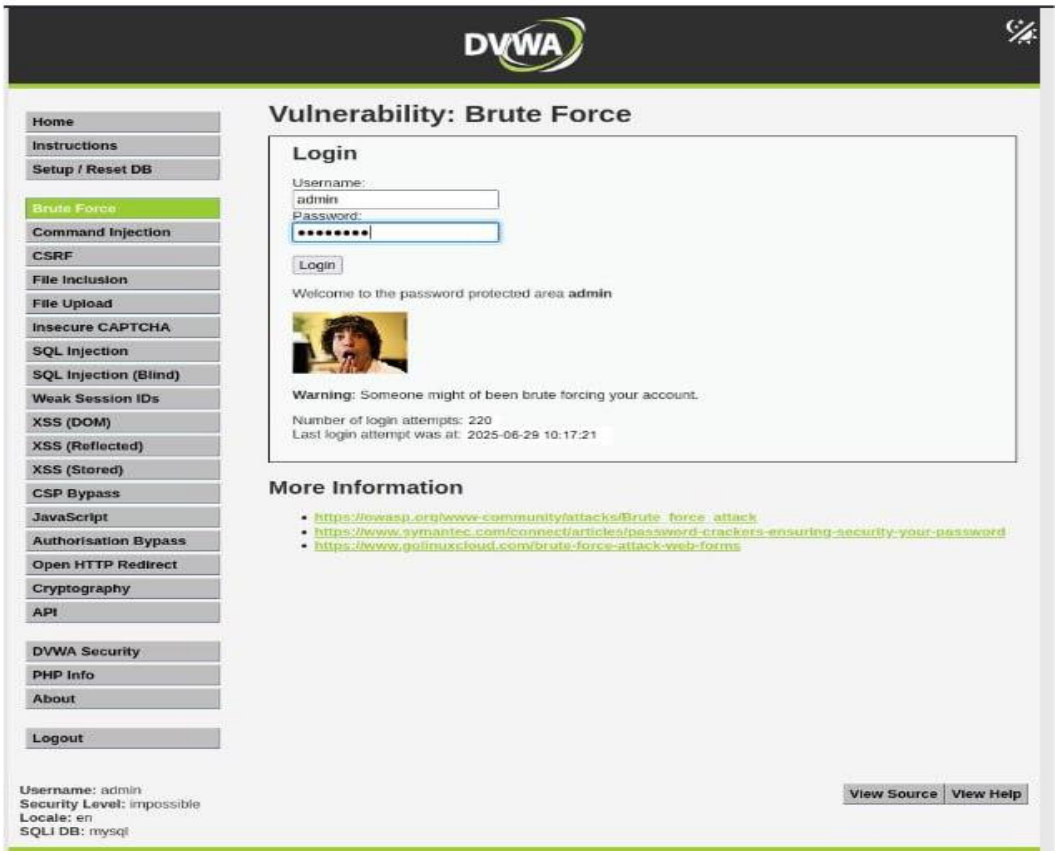
- Launched attack via Burp Intruder
- Monitored response lengths and status codes
- Identified correct password based on unique response behavior



Screenshot: Brfo_5.png

Result:

Successful login detected using the password: **password**



Screenshot: Brfo_6.png

Analysis & Observations:

- The success of the login attempt was identified by observing changes in response codes and the size of the returned content.
- The absence of mechanisms like CAPTCHA or rate-limiting allowed the attack to work effectively, even with maximum security settings enabled.

Security Recommendations:

- Implement account lockout after multiple failed attempts
- Enforce CAPTCHA to mitigate automated attacks
- Apply strong password policies and multi-factor authentication

Learning Outcomes:

- Deepened understanding of brute force attack methodologies
- Enhanced skills in HTTP request analysis and manipulation
- Practical exposure to Burp Suite’s Intruder module

Ethical Note:

This activity was carried out in a secure, controlled lab setup using deliberately vulnerable applications strictly for learning purposes. Gaining unauthorized access to real-world systems is both illegal and unethical.

Brute Force Authentication Attack:

- **What was done:** A brute-force attack was carried out using Burp Suite Intruder, applying a custom-built password list to test login vulnerabilities.

- **Conclusion:** The absence of security features such as rate limiting, CAPTCHA, and account lockout made the application susceptible to brute-force attacks, even under the "Impossible" difficulty level. This highlights the importance of implementing strong authentication safeguards.

Overall Conclusion:

The security assessment performed on DVWA effectively showcased how widely known vulnerabilities like SQL Injection, Cross-Site Scripting (XSS), and insufficient authentication mechanisms can be leveraged to breach a web application. Each simulated attack exposed serious weaknesses that could result in data theft, unauthorized access, and significant reputational and financial consequences in real environments.

This evaluation emphasizes the importance of:
• Implementing strict input validation and proper output encoding
• Following secure coding techniques (e.g., using prepared statements)
• Applying strong authentication systems with features like rate limiting, CAPTCHA, and multi-factor authentication
• Conducting routine security reviews and penetration testing

References:

[1] D. Hunt, Damn Vulnerable Web Application (DVWA). GitHub. [Online]. Available: https://github.com/digininja/DVWA [Accessed: Jul. 14, 2025].

[2] OWASP Foundation, Cross Site Scripting (XSS). OWASP. [Online]. Available: https://owasp.org/www-community/attacks/xss/ [Accessed: Jul. 14, 2025].

[3] PortSwigger Ltd., Burp Suite Documentation. PortSwigger. [Online]. Available: https://portswigger.net/burp/documentation [Accessed: Jul. 14, 2025].