Task Five -
DVWA Report

Done by

Meera Saju

Introduction

This report presents an in-depth analysis of SQL Injection vulnerabilities discovered in the Damn Vulnerable Web Application (DVWA), a purposely insecure web platform designed for security professionals to practice ethical hacking in a controlled, legal setting. The primary focus of the report is to showcase the exploitation of SQL Injection flaws across three distinct security levels—Low, Medium, and High. It outlines the steps involved in carrying out successful SQL injections, examines the security measures implemented at each level, and proposes effective mitigation strategies to protect real-world web applications from similar attacks.

DVWA Installation

To install and access DVWA, I used the PentestLab GitHub repository. Below are the steps I followed to clone and set up the environment:

1. Create the Target Directory

First, I created a directory named `pentestlab` and navigated into it:

```
mkdir pentestlab
cd pentestlab
```

2. Clone the DVWA Repository

I then cloned the DVWA repository from GitHub using the following command:

```
git clone https://github.com/eystsen/pentestlab.git
```

3. List the Directory Contents

To verify the files and scripts available in the `pentestlab` directory, I executed:

```
ls
```

This revealed the presence of the main script, `pentestlab.sh`, which is used to manage and launch various vulnerable web applications.

4. Start the DVWA Lab

To deploy DVWA and its required services (Apache, MySQL, PHP), I ran the following command:

```
./pentestlab.sh start dvwa
```

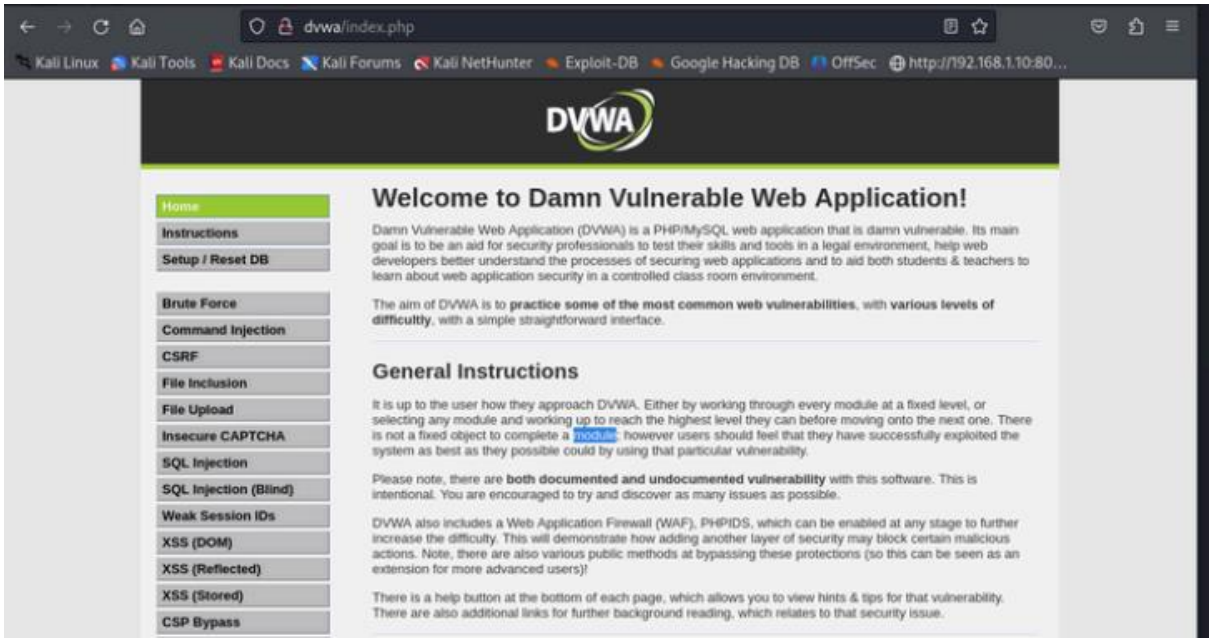
The DVWA login page is now accessible at:

`http://DVWA/`

Credentials:

- Username: admin
- Password: password

This successfully launched DVWA, allowing for further testing and exploitation.

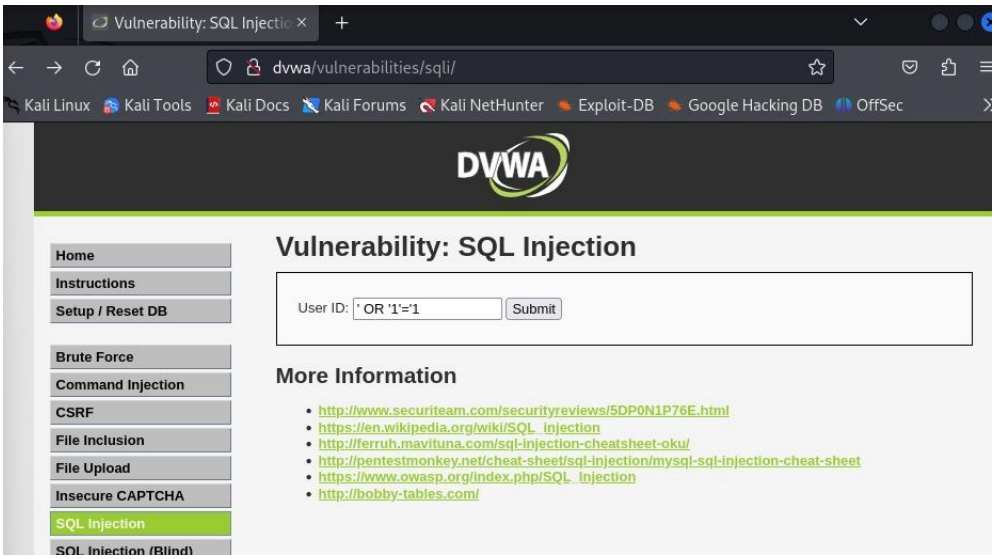


LOW LEVEL SQL INJECTION

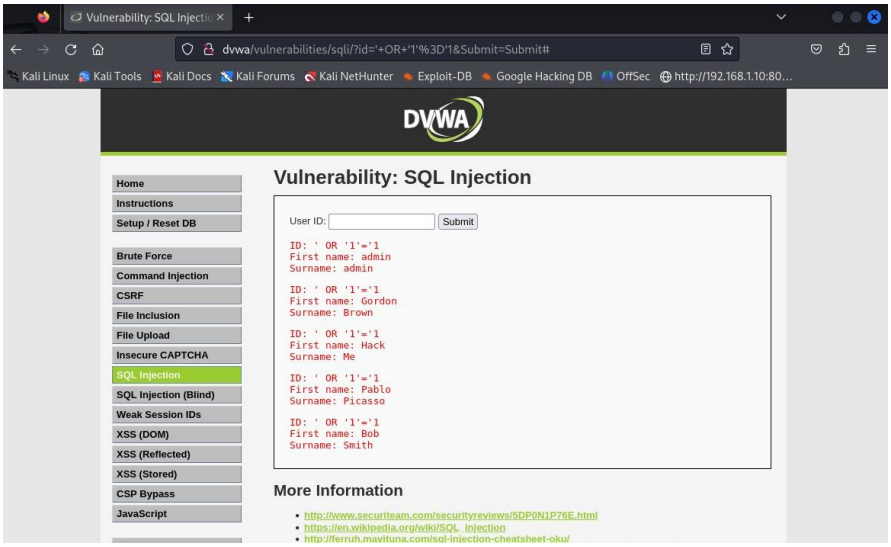
By default, SQL injection in DVWA was set to the **Low security** level. There was a field available where SQL queries could be injected. I initiated a simple exploitation using the query:

' OR '1'='1

in the **User ID** field.



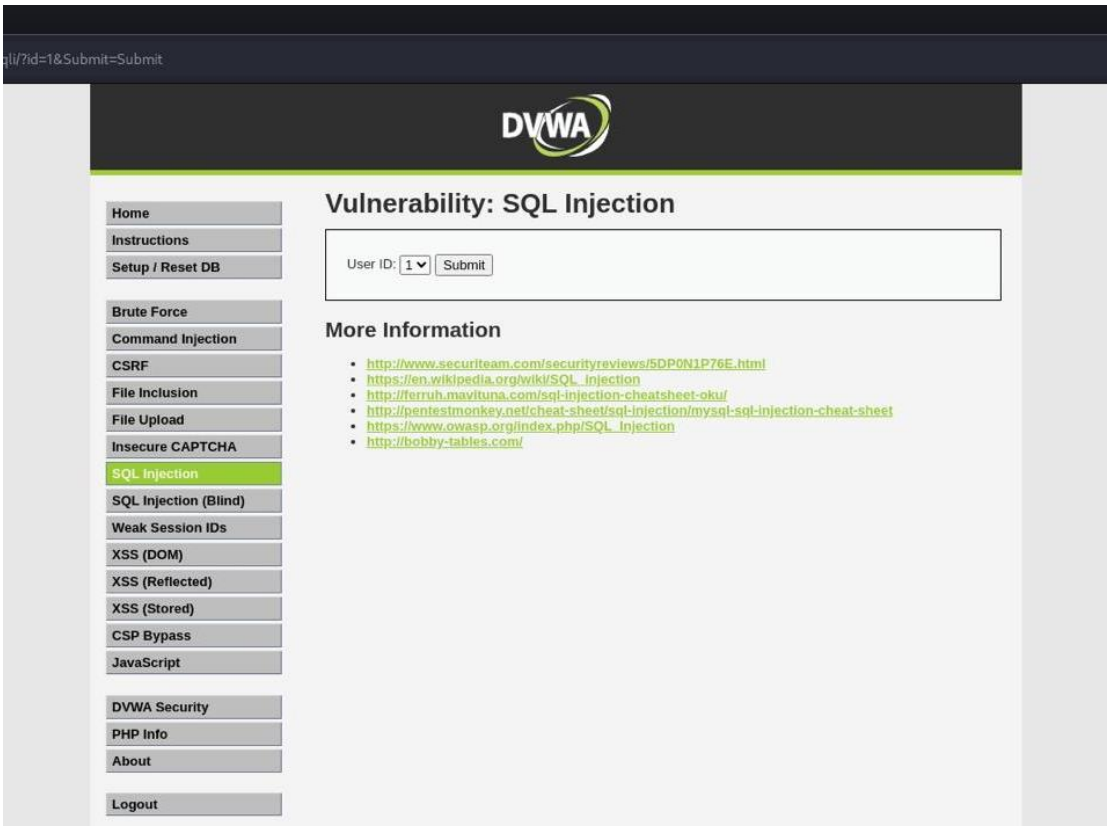
This allowed me to bypass authentication and retrieve information from the database, successfully demonstrating the vulnerability.



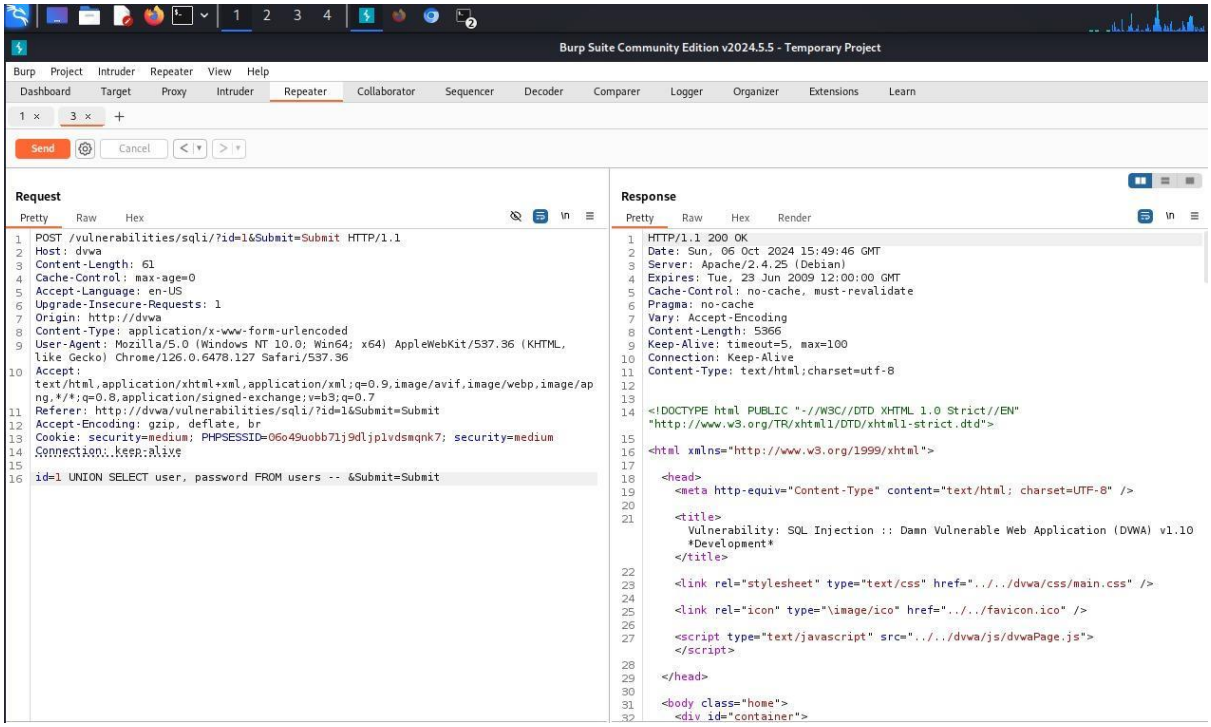
MEDIUM LEVEL SQL INJECTION

At the medium security level, I used Burp Suite to perform an SQL injection. I began by opening the browser within Burp Suite and accessing DVWA. After logging in, I entered a random input into the target field and submitted it.

Next, I navigated to the HTTP history in Burp Suite, where I found the GET request generated after the form submission. Initially, the security level was set to "Low", so I modified the request to "Medium" and sent it. I then opened the response in the browser, which redirected me to the Medium-level SQL injection task page.



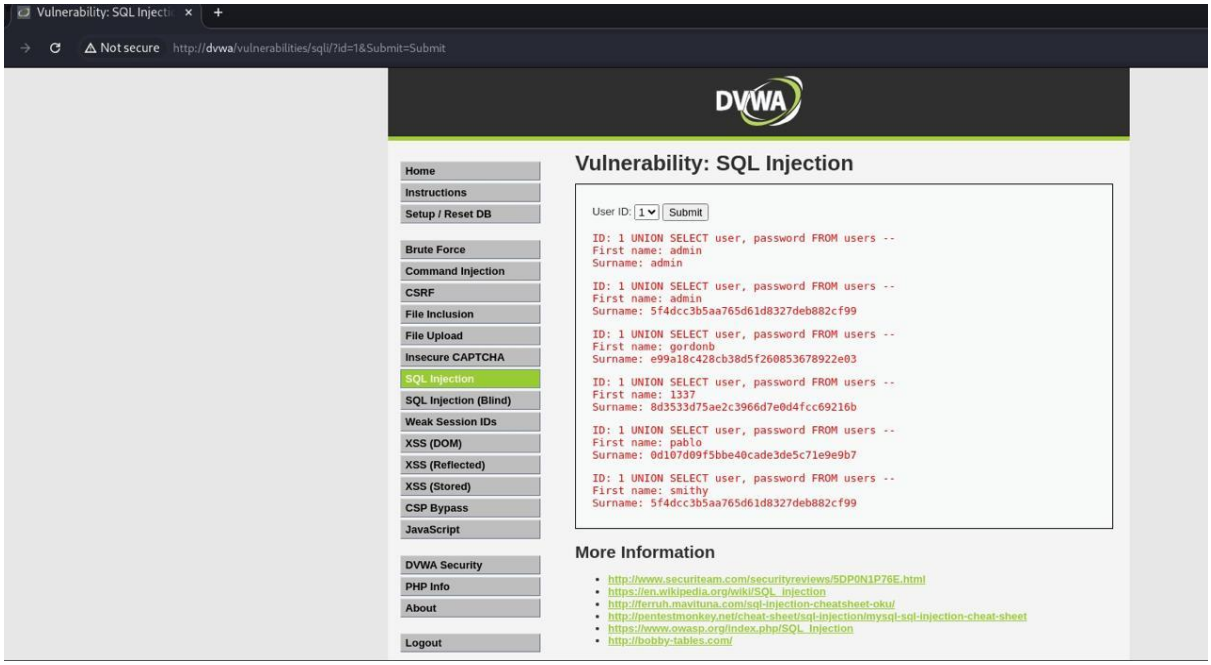
On the task page, I selected "1" and submitted the form. Going back to Burp Suite, I found a POST request related to the submission. I sent this request to the Repeater, changed the security level from "Low" to "Medium", and edited the payload.



For the SQL injection, I inserted the following query in the `id` field:

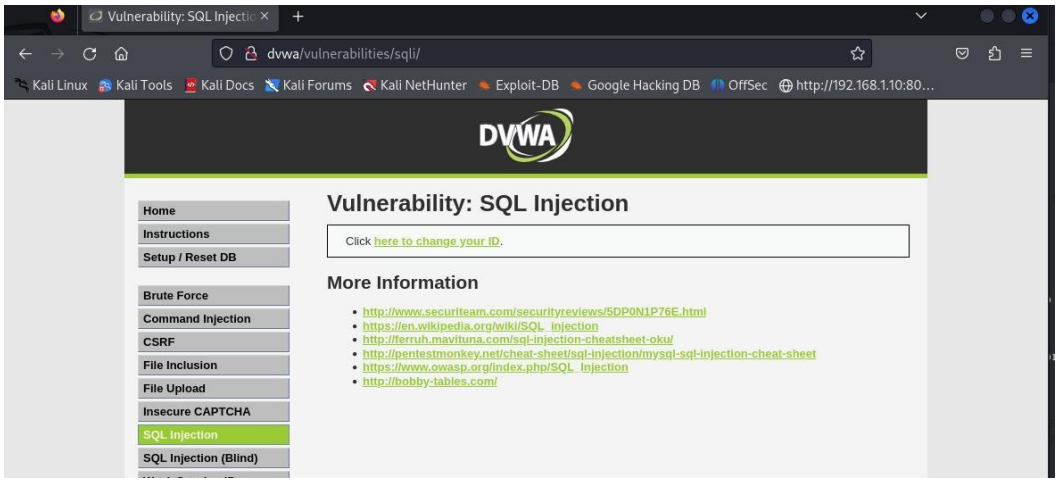
1 UNION SELECT user, password FROM users –

In the response, I successfully retrieved the usernames and passwords of the users, demonstrating the SQL injection vulnerability.



HIGH LEVEL SQL INJECTION

To test SQL injection at the High security level in DVWA, I first set the security level to High in the DVWA Security settings. This redirected me to the appropriate page for the high-level SQL injection task.



On the new page, I was able to inject SQL queries. To retrieve more information, I used the following injection:

1' UNION SELECT user, password FROM users #

After executing the query, I successfully obtained additional user details, demonstrating that SQL injection was still possible at the high security level.