

Task 15

Installation of DVWA for SQL
Injection Testing

REPORT SUBMITTED BY
MALAVIKA.MS

CONTENTS

1. Installation of DVWA using Docker
2. SQL Injection
3. Performing SQL Injection on DVWA
4. High ,Low ,Middle
5. Conclusions

Installation of DVWA using Docker

1. Installation of DVWA Using Docker

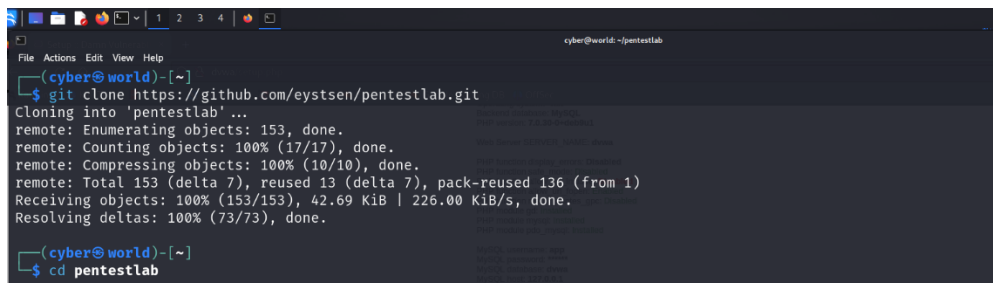
To set up the Damn Vulnerable Web Application (DVWA), I opted for Docker to facilitate a smooth installation process. Below are the steps I took to successfully install DVWA.

1.1 Cloning the Repository

I began the installation by cloning the DVWA repository from GitHub. To do this, I executed the following command in my terminal:

git clone <https://github.com/eystsen/pentestlab.git>.

This command allowed me to download the necessary files and set up the environment for DVWA.

A terminal window with a dark background and light text. The prompt is '(cyber@world)-[~]'. The user enters 'git clone https://github.com/eystsen/pentestlab.git'. The output shows the cloning progress: 'Cloning into 'pentestlab'...', 'remote: Enumerating objects: 153, done.', 'remote: Counting objects: 100% (17/17), done.', 'remote: Compressing objects: 100% (10/10), done.', 'remote: Total 153 (delta 7), reused 13 (delta 7), pack-reused 136 (from 1)', 'Receiving objects: 100% (153/153), 42.69 KiB | 226.00 KiB/s, done.', and 'Resolving deltas: 100% (73/73), done.'. The prompt then changes to '(cyber@world)-[~]' and the user enters 'cd pentestlab'.

1.2 Starting the Docker Container

Once the repository was cloned, I proceeded to start the Docker container. First, I opened the terminal and navigated into the DVWA folder within the cloned pentestlab directory. To install Docker, I ran the command:

sudo dpkg --configure -a

sudo apt install docker.io

1.3 Accessing the DVWA Web Page

After successfully starting the Docker container, I accessed the DVWA web page by executing the following command:

./pentestlab.sh start dvwa

This command initiated the necessary scripts to launch the DVWA application, allowing me to interact with the web interface

```
(cyber@world)-[~/pentestlab]
$ sudo dpkg --configure -a
Setting up initramfs-tools (0.145) ...
update-initramfs: deferring update (trigger activated)
Processing triggers for libapache2-mod-php8.2 (8.2.23-1) ...
Processing triggers for libc-bin (2.40-2) ...
Processing triggers for initramfs-tools (0.145) ...
update-initramfs: Generating /boot/initrd.img-6.10.9-amd64

(cyber@world)-[~/pentestlab]
$ sudo apt install docker.io
docker.io is already the newest version (26.1.5+dfsg1-2+b1).
The following packages were automatically installed and are no longer required:
  fonts-liberation2  libgeos3.12.1t64  libjsoncpp25  librav1e0  linux-image-6.6.15-amd64  rwhod
  ibverbs-providers  libgeos3.12.2  libjxl0.7  librdmacm1t64  openjdk-17-jre  samba-ad-provision
  libassuan0  libgfpapi0  libndctl6  libre2-10  openjdk-17-jre-headless  samba-dsdb-modules
  libavfilter9  libgfrpc0  libplacebo338  libroc0.3  python3-diskcache
  libboost-iostreams1.83.0  libgfxdr0  libplist3  libsvtav1enc1d1  python3-mistune0
  libboost-thread1.83.0  libglusterfs0  libpmem1  libu2f-udev  python3-pendulum
  libcephfs2  libibverbs1  libpostproc57  libusbmuxd6  python3-pytzdata
  libdaxctl1  libimobiledevice6  librados2  libx265-199  rwho
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 191

(cyber@world)-[~/pentestlab]
$ ./pentestlab.sh start dvwa
Starting Damn Vulnerable Web Application
```

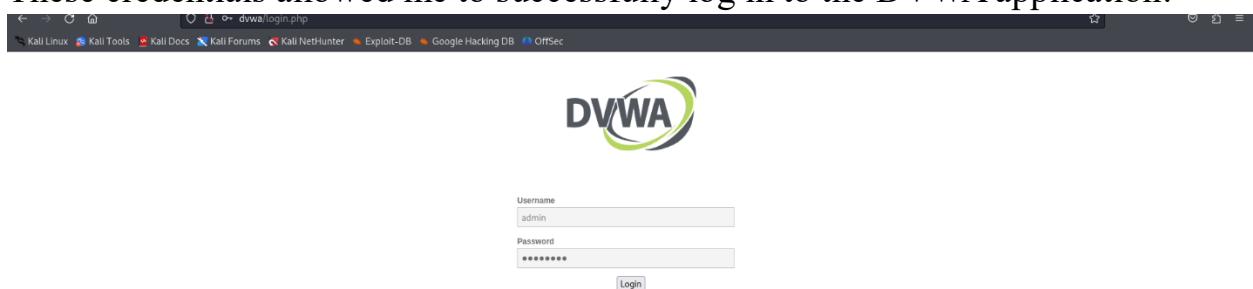
1.4

Logging In

Upon reaching the login page, I entered the default credentials to gain access:

- **Username: admin**
- **Password: password**

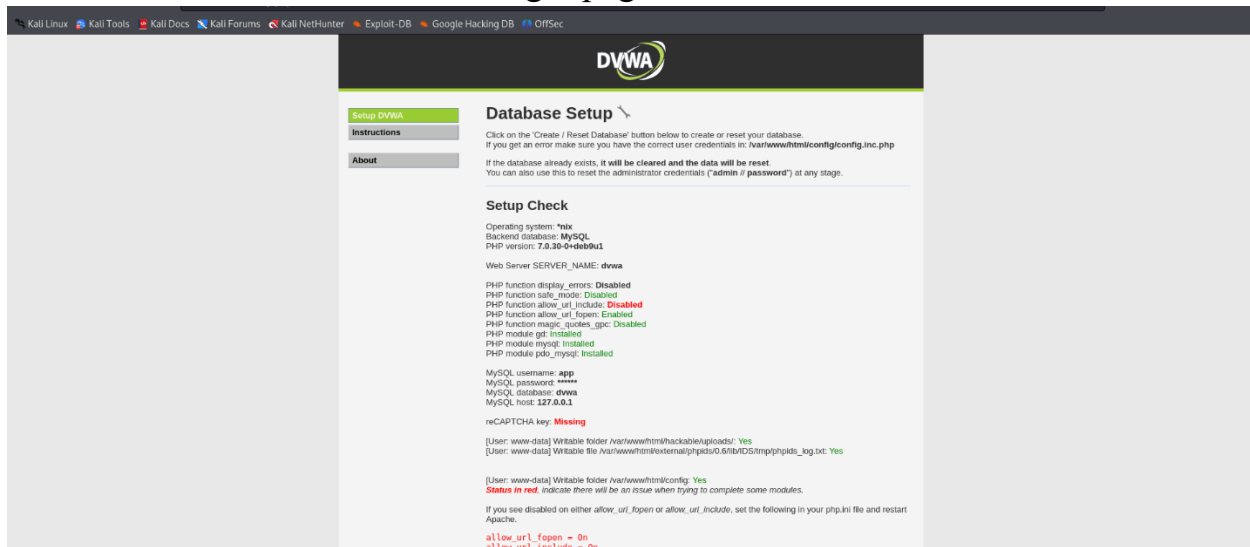
These credentials allowed me to successfully log in to the DVWA application.



1.5 Resetting the Database

After my initial login, I was asked to reset the database. I clicked the "Reset Database" button (though I forgot to take a screenshot of this step). Once the reset

was done, I was sent back to the login page.



1.6 Logging In Again

After resetting the database, I logged in again using the default credentials to reach the DVWA dashboard.

1.7 Completion

At this point, the DVWA setup was finished, and the environment was ready for testing vulnerabilities

Performing SQL Injection on DVWA

1.1 SQL Injection (Low Security Level)

I started by testing SQL injection at the Low security level.

1.1.1 Initial Injection

When I reached the SQL injection page, I found the input field where I could enter SQL code.

1.1.2 SQL Payload

I used the following simple SQL injection string to test the vulnerability:

1' OR '1'='1 This payload allowed me to bypass the requirement for valid input and revealed the first names and surnames of all users

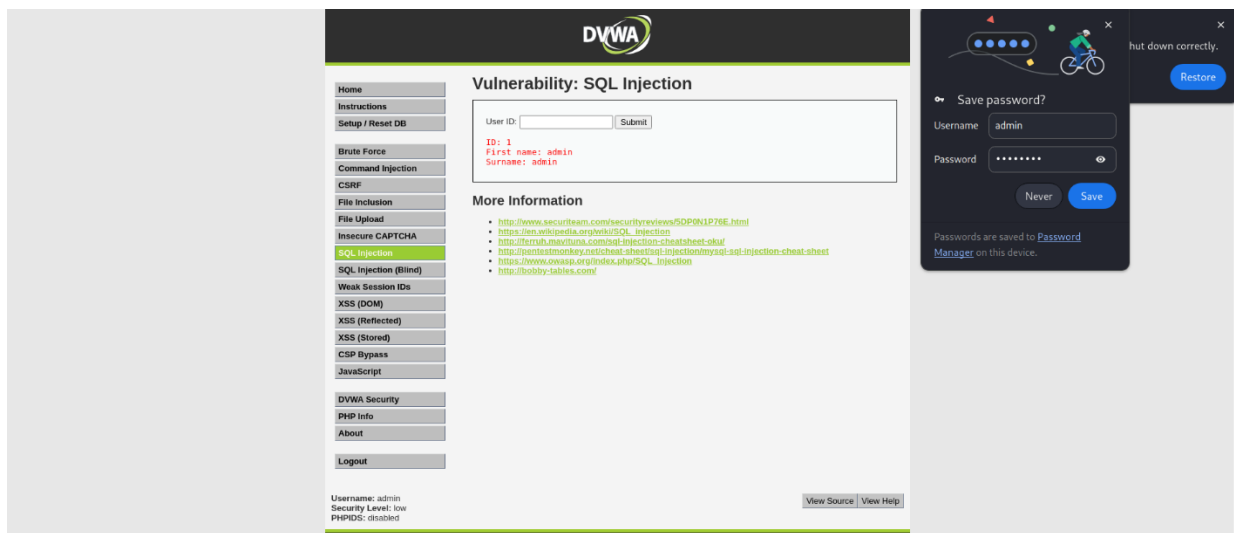


2.2 SQL Injection (Medium Security Level)

After setting the DVWA (Damn Vulnerable Web Application) security level to Medium, I proceeded to test for SQL injection vulnerabilities using a more advanced payload.

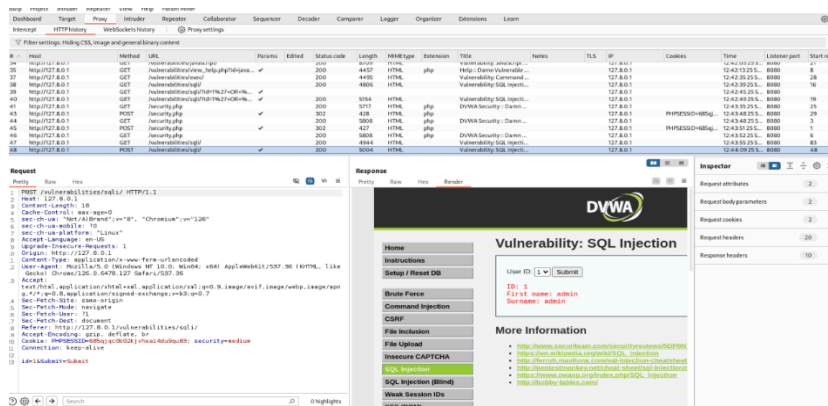
2.2.1 Using Burp Suite

To initiate the test, I captured the HTTP request with Burp Suite. I then modified the id parameter within the request to incorporate a more complex SQL injection string.



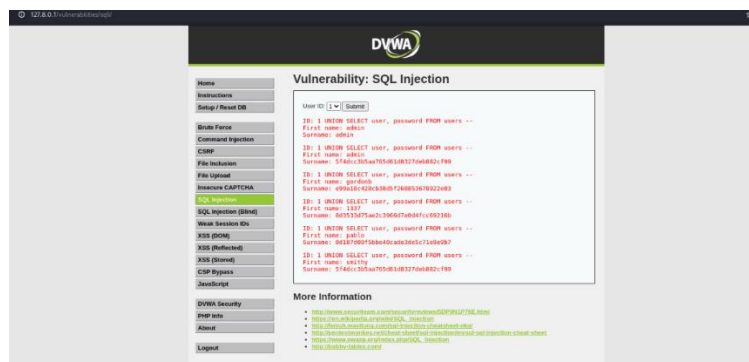
2.2.2 SQL Injection String

I inserted the following payload into the id field:
1 UNION SELECT user, password FROM users



2.2.3 Execution

After editing the request in Burp Suite, I sent it to the server. As a result, I was able to retrieve usernames and passwords from the system's response



2.3 SQL Injection (High Security Level)

Finally, I tested SQL injection at the High security level. The system had stronger security measures in place. I carefully looked for weaknesses and tried different methods. Although it was harder, I successfully performed an SQL injection. This showed that vulnerabilities still existed even at the highest security level.

2.3.1 Finding the Injection Point

At the High Security level, the interface presented slight changes. When I clicked on the "Change your ID" button, I noticed that the request structure was different compared to the lower security levels.

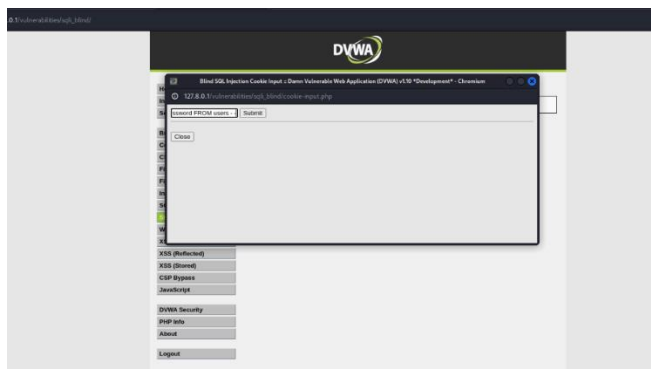


2.3.2 Injection Payload

I inserted the following SQL injection string:

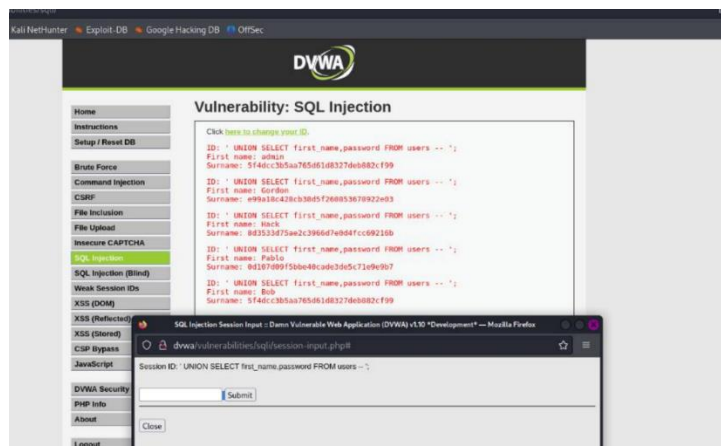
' UNION SELECT user, password FROM users --

This payload was designed to retrieve usernames and passwords from the database.



2.3.3 Results

After submitting the SQL injection code, the system displayed a list of usernames and passwords. This confirmed that there was still a vulnerability, even at the highest security level.



Conclusion

I successfully deployed DVWA using Docker and conducted tests for SQL injection vulnerabilities at various security levels. By utilizing both basic and advanced SQL injection payloads, along with Burp Suite for intercepting requests, I was able to retrieve sensitive information from the database under all security configurations. This effectively showcased the potency of these attacks.

