UNIVERSITY SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

MINOR PROJECT REPORT

DETECTION OF MALICIOUS TRAFFIC USING PYTHON AND WIRESHARK

Submitted by:
Naman Rajput
01616412816
B. Tech. ECE 7th sem

Naman Mittal
41216412816
B. Tech. ECE 7th sem

Submitted to:
Dr. M. Bala Krishna

CERTIFICATE OF ORIGINALITY

This is to certify that the dissertation entitled "DETECTION OF MALICIOUS TRAFFIC USING PYTHON AND WIRESHARK" done by Mr. Naman Rajput, Roll No. 01616412816 and Mr. Naman Mittal, Roll No. 41216412816 is an authentic work carried out by them at the University School of Information and Communication Technology under my guidance. The matter embodied in this Dissertation has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

Dr. M. Bala Krishna

ACKNOWLEDGEMENT

It gives us immense pleasure to take this opportunity to acknowledge our obligation to our Dissertation guide, Dr. M. Bala Krishna, Associate Professor, University School of Information and Communication Technology, GGSIPU, who has not only guided us throughout the Dissertation but also made a great effort in making the Dissertation a success. We are highly thankful to our guide for his keen interest, valuable guidance, technical acumen, round the clock encouragement, moral support & suggestions in the completion of the dissertation.

Naman Rajput
Roll No. 01616412816
B.Tech ECE 7th Sem
University School Of Information
Communication & Technology

Naman Mittal
Roll No. 41216412816
B.Tech ECE 7th Sem
University School Of Information
Communication & Technology

CONTENTS

ABSTRACT

A network at any level needs a level of safety and security to work at an ideal pace. A user in an open network is prone to multiple unwanted traffic, due to which the user experiences a lag in network. To ensure a smooth sailing and uninterrupted safe surfing, our aim is to create a program which could detect potential attacking scenario that we may encounter in our time surfing an open network. By detecting which users warrant a purpose for an action against them, we enable the user to block the malicious user or packets. This would enable the user to work in an unknown, possibly hostile environment, without worrying about invasion to privacy.

PROBLEM STATEMENT

In daily use, we encounter a lot of public network which might not be secure due to which our systems are prone to malware and unwanted traffic which can breach our privacy, steal data or worse, to overcome this problem we are working on a **python script/program which could detect malicious packets in real-time using pyshark**, which is a Python wrapper for tshark, the filtering of packets would be based on the protocols the packets belong to, number of packets the same ip send, port the packet wants to access etc.

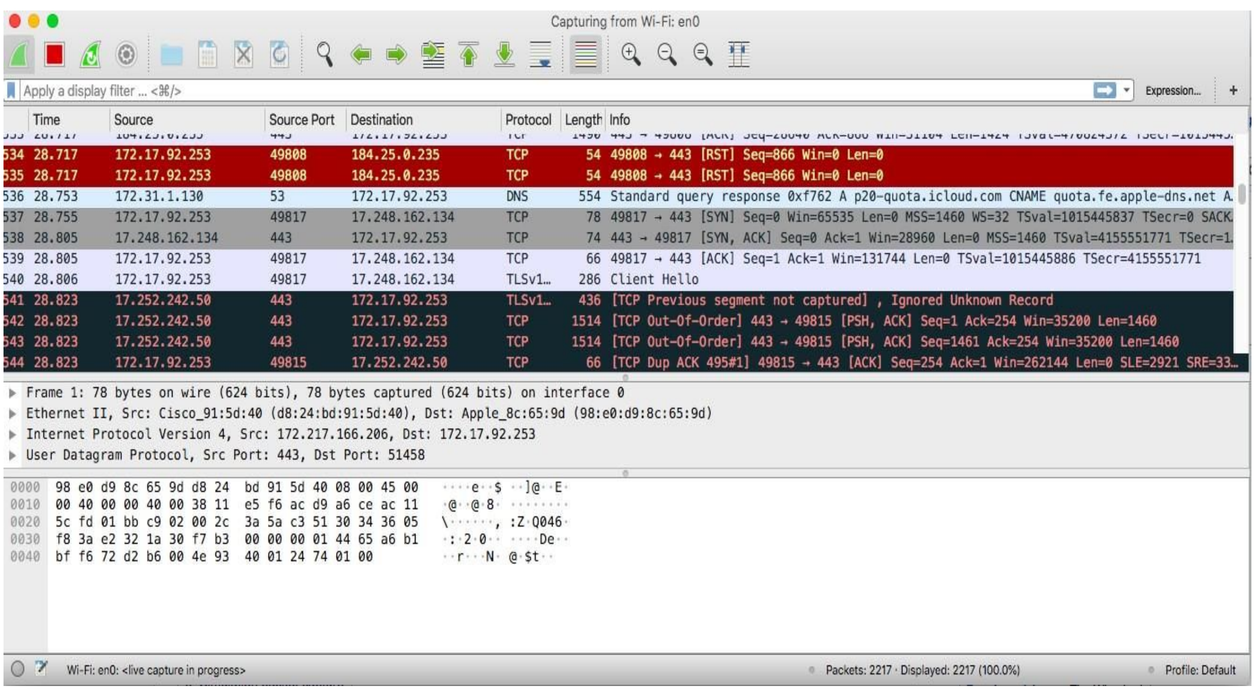
WIRESHARK

The **wireshark** tool lets the user filter out the packets relevant using display filter and the capture filter, using this feature we'll first figure which packet might be suspicious enough to harm the network and then apply them in the filter through which we could easily identify which packets to warn the user about.

The harmful packets could be of any layer possible, and each malicious packet would have its own special features which could make it different from other packets of the same layer. The characteristics could want: access to a port of the destination, send a large number of malicious packets, to change its MAC address.

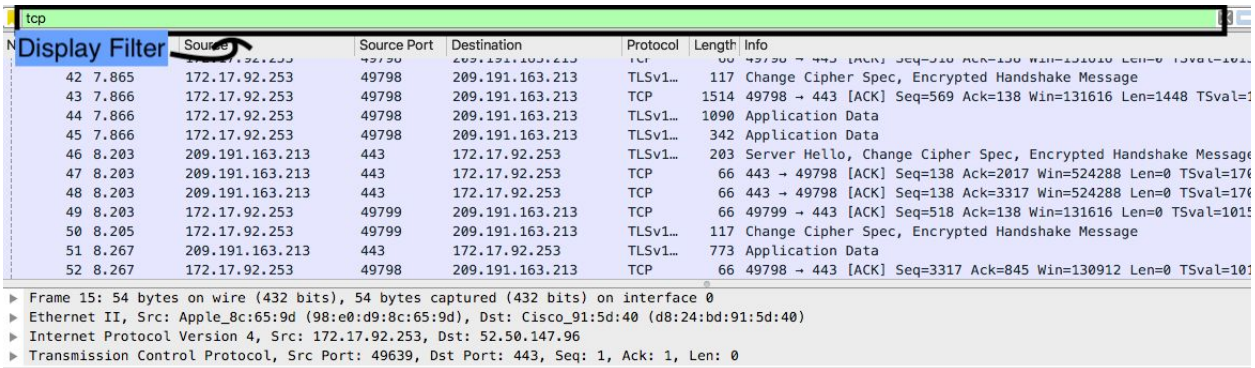
Wireshark is free open-source packet analyzer. It is a cross-platform using the Qt widget toolkit for its user interface, to capture packet it uses pcap, it can run on OS like: MacOS, Linux, Solaris and Microsoft Windows. It is used for troubleshooting of network and also analysing the packets, also for software and communications rules development and more.

Wireshark lets the user put the current network into promiscuous mode, where they can see all the traffic visible on that interface including single link traffic not sent to that MAC address of the network control interface.

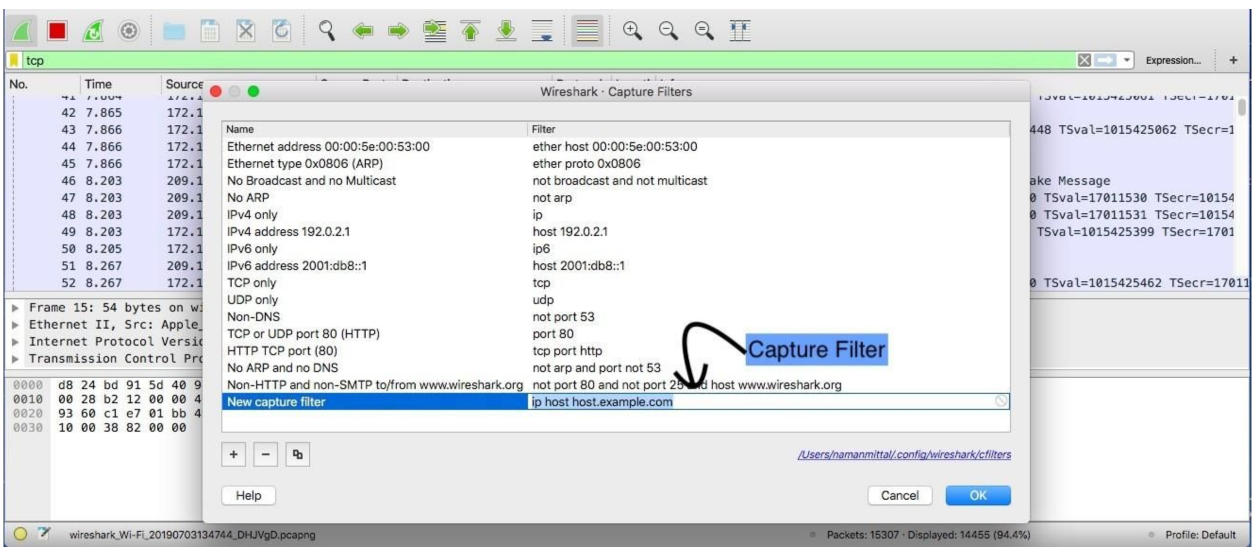


Sample window of Wireshark Tool

Wiresharks lets the user filter out the packets captured for analysing, it has two filters, a display filter, which filters the packets captured and shows the relevant ones and a capture filter, which only catches the packets that are relevant according to the filter. Both the filters work differently and have a different syntax.



Display Filter



Capture Filter

Wireshark colours packets based on rules that correspond to the features in packets, to help the user identify the types of packets easily. A default set of protocols is also provided; users can alter predefined rules of coloring packets and add or remove rules.

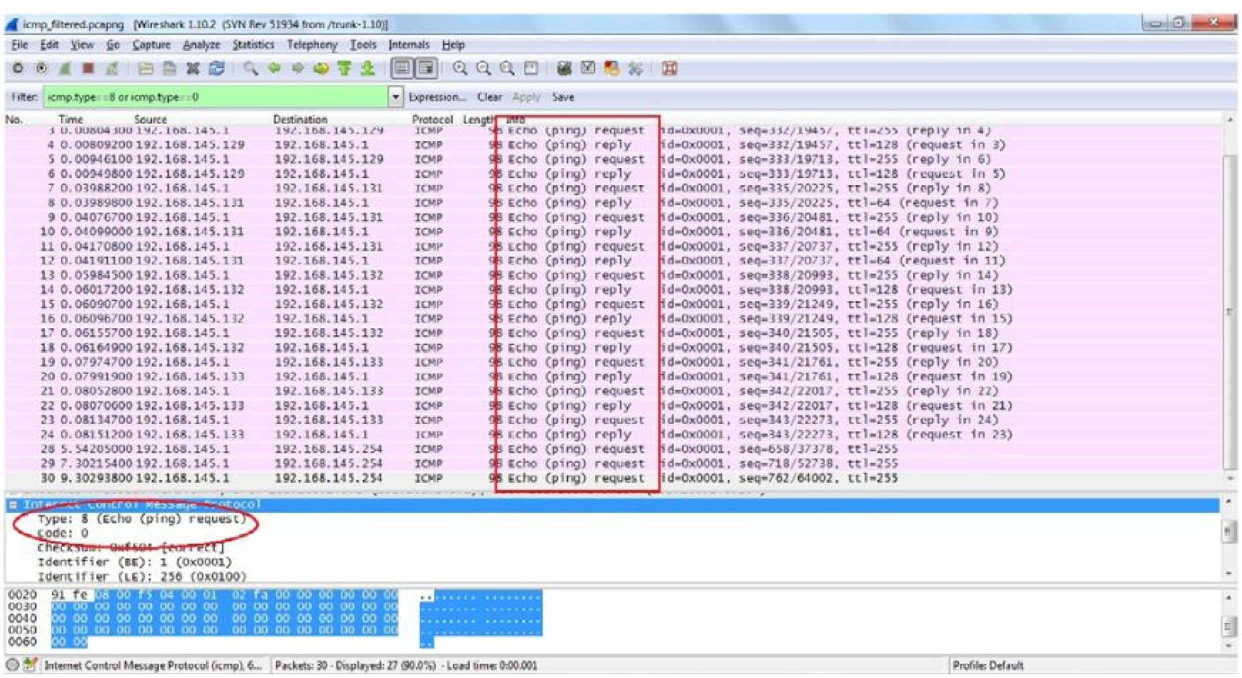
PING SWEEP ATTACK

Ping sweep scan is used in a network at many layers, to find out which IPs are currently active in the network. It can be performed using many layers: TCP or UDP or ICMP, but the most used one is ICMP Ping Sweep. In this, several ICMP type 8, ECHO requests are first sent after which, ICMP type 0, ECHO reply packets are used for extraction of IP.

If the target host doesn’t acknowledge ECHO service then this ping sweep will not function. Which is why ICMP ping sweep is mostly used, but if there is a firewall in between which has the ability to block incoming ICMP packets then ICMP ping scan also is of no use.

To detect ICMP ping sweep in Wireshark a simple filter of icmp.type==8 or icmp.type==0 is applied. After applying this filter if more than expected packets are received, then it’s possible that ping sweep is trying to access and get into the network.

We need to be careful about the volume of traffic of these packets, as it might be normal ping traffic of the network. Ping traffic should be considered as a ping scan only if you are receiving unexpected number of increase in ICMP traffic.



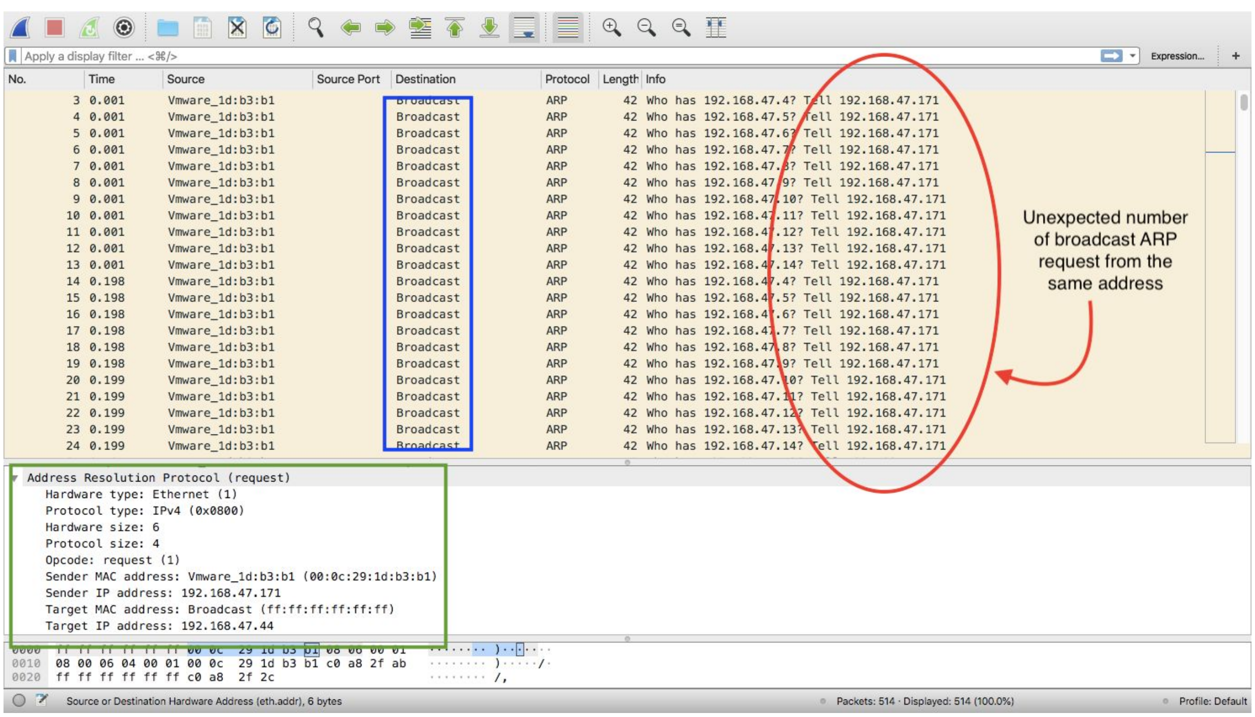
ARP Sweep/ARP Scan Attack

If a firewall is placed in obstruction to ICMP packets, it is blocked and ICMP ping sweep becomes of no use. In such similar situations, ARP sweep is used to find out active IPs in the current network.

While performing ARP scan, attacker broadcasts ARP packets with destination MAC as 0xff:ff:ff:ff:ff:ff, for each and every possible IP destination in the selected subnet and an ARP response means that the host is active or live.

This attack got an advantage over ping scan as ARP communication can’t be filtered or disabled as all TCP/IP network is completely based on it. Blocking or disabling ARP packets will definitely break TCP/IP communication or it will result in static ARP entries. A disadvantage of this attack is that it can’t penetrate layer-3 Devices.

Detection of this attack is quite simple too. Unexpected number of ARP requests are evident in the screenshot, it is a clear indicator for ARP scan or ARP sweep being performed on the network.

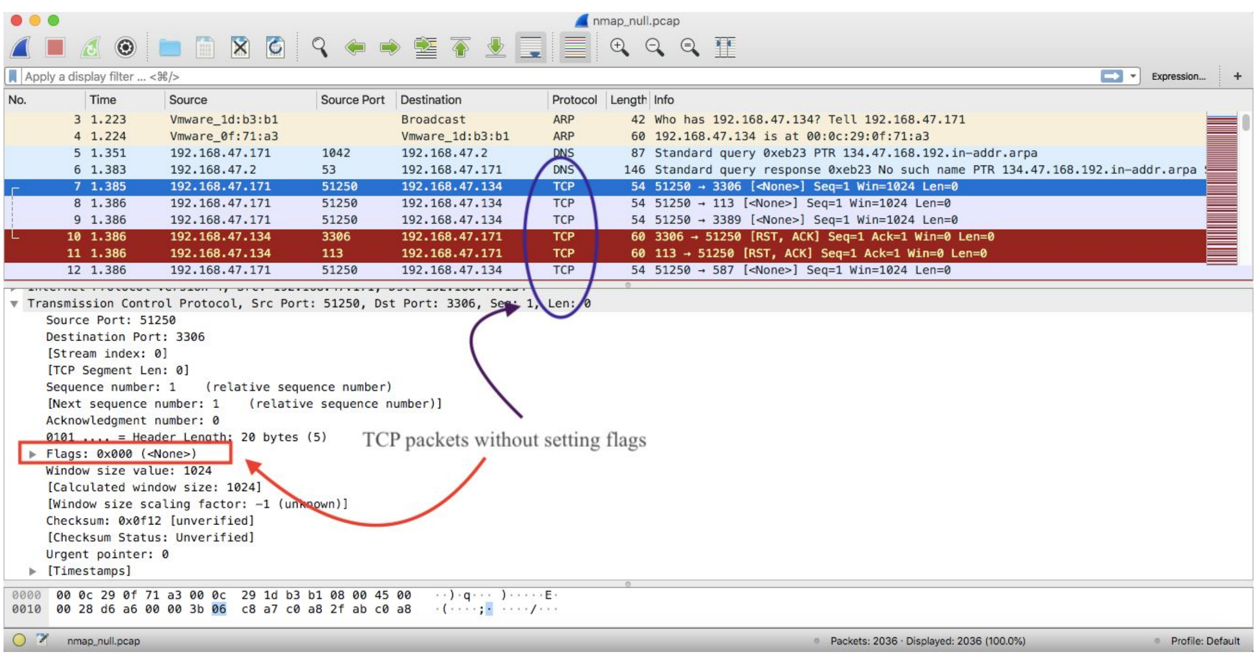


NULL Scan

A Null scan in a network is implemented in the TCP layer of the network. It is used to identify the ports of a network of a user, so that the user can be identified. A Null scan is the easiest scan to be performed to gain information of the user. It can be detected using a simple filter on the TCP layer of the network.

To detect whether a Null scan is being performed on a network or not, the hacker sends a large number of malicious packets of the TCP layer without setting any flag on the packets of this layer. If in response, of the packets sent, the hacker gets a RST reply to the packets sent, they can conclude that the port to which the packet was sent is closed.

The Null scan port is firewalled if no response is received by the hacker, i.e. the port is either open, filtered, or the response received as reply packets are of Type 3, Code 1, 2, 3, 9, 10 or 13 of the ICMP layer. To filter out the packets without any flag, a condition of 0x000 of flags of TCP packet received is set.



NMAP: The Network Mapper

Nmap is a free and open software which is used for Network Management in an open network. Nmap uses multiple new ways to find out the ip address of a system in a network. It can be used to monitor host and destination status in a network. Nmap is available as official binary packages on platforms like Windows, Linux, MacOs to use for network observation. Nmap has a classic command line structure but also has a GUI.

We are using Nmap to find out different systems in an open network so that we can find the IPs of these systems. We attack a system using nmap and the attacks it uses to find out the IPs. By attacking using Nmap, we can identify the attack that we actually conducted from one system to another. By using Nmap we can conclude that our detection software is working as per the attacks done by the Nmap system.

Following is the interface of NMAP network analyser:

```
gh0st@DedSec:~$ nmap --help
Nmap 7.60 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
```

FPING: Ping Tool

Fping is a command line tool for Linux in which, ICMP packets are sent in a network as echo requests at a high performing network range. This means that, a ping echo request of ICMP is sent within a network to the range of all the systems present in the network. First, it sends request to one host, then moves to another host to send the same request. This process of moving from one host to another is done in a round robin manner.

Following is the interface of FPING Ping Tool:

```
gh0st@DedSec:~$ fping --help
Usage: fping [options] [targets...]

Probing options:
   -4, --ipv4         only ping IPv4 addresses
   -6, --ipv6         only ping IPv6 addresses
   -b, --size=BYTES   amount of ping data to send, in bytes (default: 56)
   -B, --backoff=N    set exponential backoff factor to N (default: 1.5)
   -c, --count=N      count mode: send N pings to each target
   -f, --file=FILE    read list of targets from a file ( - means stdin)
   -g, --generate     generate target list (only if no -f specified)
                      (give start and end IP in the target list, or a CIDR address)
                      (ex. fping -g 192.168.1.0 192.168.1.255 or fping -g 192.168.1.0/24)
   -H, --ttl=N        set the IP TTL value (Time To Live hops)
   -I, --iface=IFACE  bind to a particular interface
   -l, --loop         loop mode: send pings forever
   -m, --all          use all IPs of provided hostnames (e.g. IPv4 and IPv6), use with -A
   -M, --dontfrag     set the Don't Fragment flag
   -O, --tos=N        set the type of service (tos) flag on the ICMP packets
   -p, --period=MSEC  interval between ping packets to one target (in ms)
                      (in loop and count modes, default: 1000 ms)
   -r, --retry=N      number of retries (default: 3)
   -R, --random       random packet data (to foil link data compression)
   -S, --src=IP       set source address
   -t, --timeout=MSEC individual target initial timeout (default: 500 ms,
                      except with -l/-c/-C, where it's the -p period up to 2000 ms)

Output options:
   -a, --alive        show targets that are alive
   -A, --addr         show targets by address
   -C, --vcount=N     same as -c, report results in verbose format
   -D, --timestamp    print timestamp before each output line
   -e, --elapsed      show elapsed time on return packets
   -i, --interval=MSEC  interval between sending ping packets (default: 10 ms)
   -n, --name         show targets by name (-d is equivalent)
   -N, --netdata      output compatible for netdata (-l -Q are required)
   -o, --outage       show the accumulated outage time (lost packets * packet interval)
   -q, --quiet        quiet (don't show per-target/per-ping results)
   -Q, --squiet=SECS  same as -q, but show summary every n seconds
   -s, --stats        print final stats
   -u, --unreach      show targets that are unreachable
   -v, --version      show version
```

Methodology Adopted/ALGORITHM

To overcome our problem of invasion of privacy in a network, we will go through the following steps:

1. Observe the networks by exploring the methods of exploitations and bugs for accessing and using the data of a user.
2. By identifying how these exploitations work and how the data is accessed and identity compromised using this working.
3. Work on a possible way to try and identify these attacks which try and access our personal data.
4. Implement a code to detect a few of these attacks based on the exploitations, which would warn the user of the culprit.
5. Finally, try and identify the source of the attack in the network, so that further action could be taken against the user by blocking the user from the network.

SOFTWARE & HARDWARE REQUIREMENTS

- Pyshark 0.4.2.9 (https://pypi.org/project/pyshark)
- Python 2.7.16 (https://www.python.org/downloads/release/python-2716)
- Wireshark 2.6.10 (https://www.wireshark.org/)
- NMAP: Network analyser tool
- FPING: Ping tool
- Packet data of the network on which attacks were performed
- Two laptops with minimum i3 processor, 8 gb RAM & 1 gb of free space

SOURCE CODE/OUTPUT

Ping Sweep Attack

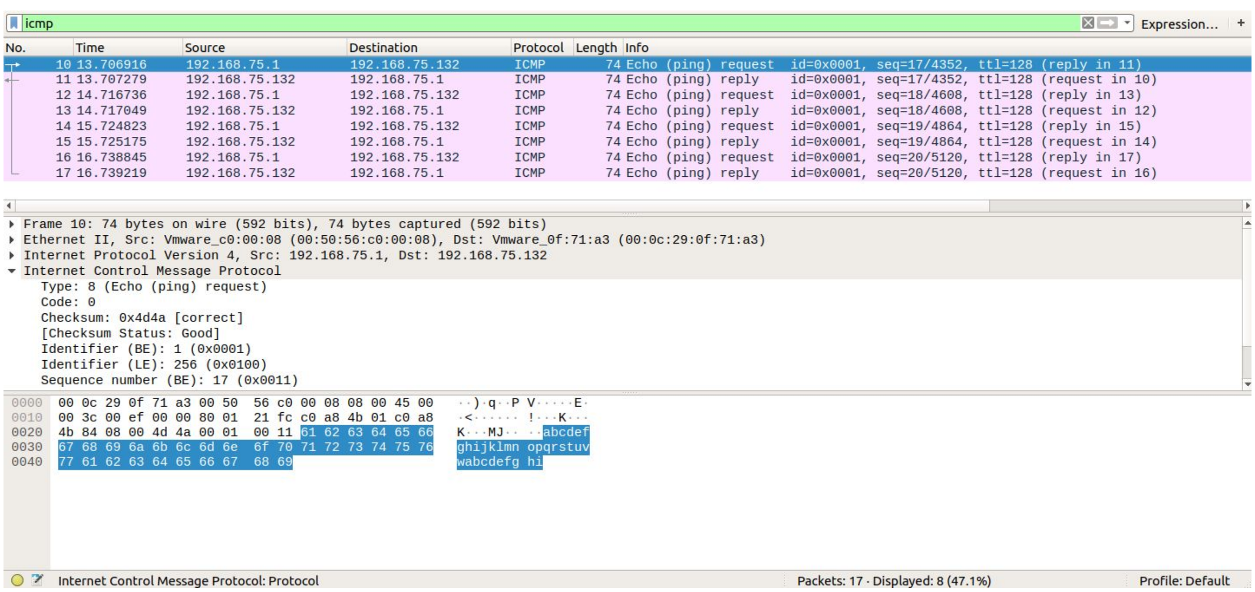
```
import pyshark
import operator

def ping_sweep():
	capture = pyshark.FileCapture('./ping.pcap', display_filter="icmp")
	under_attack=0
	while True:
	ip_address={}
	count=0
	for packet in capture:
	typ=str(packet.icmp.type)
	if typ=='0':
		count+=1
		ip=str(packet.ip.src)
		if ip in ip_address:
		ip_address[ip]+=1
		else:
		ip_address[ip]=1
	elif typ=='8':
		count+=1
		ip=str(packet.ip.dst)
		if ip in ip_address:
		ip_address[ip]+=1
		else:
		ip_address[ip]=1
	if count > 7:
	if under_attack == 0:
		print('EXCESSIVE ICMP TRAFFIC DETECTED, SOURCE IP: ',
max(ip_address.items(), key=operator.itemgetter(1))[0])
	under_attack=1
	elif count < 7:
	under_attack=0

ping_sweep()
```

Live Attack Simulation of Ping Sweep using FPING:

```
gh0st@DedSec:~$ fping -g 10.102.164.1 10.102.164.50
10.102.164.1 is alive
10.102.164.5 is alive
10.102.164.7 is alive
10.102.164.6 is alive
10.102.164.8 is alive
10.102.164.9 is alive
10.102.164.12 is alive
10.102.164.22 is alive
10.102.164.23 is alive
10.102.164.25 is alive
10.102.164.26 is alive
10.102.164.30 is alive
10.102.164.31 is alive
10.102.164.27 is alive
10.102.164.35 is alive
10.102.164.38 is alive
10.102.164.33 is alive
10.102.164.44 is alive
10.102.164.48 is alive
```

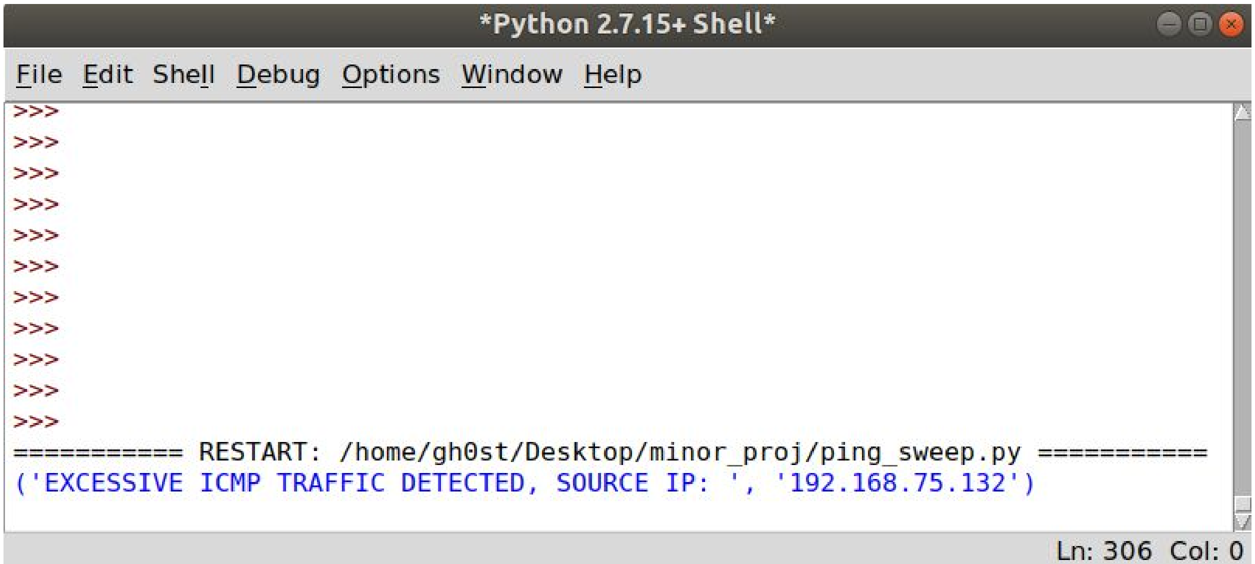
Detection of IP addresses in a network using Ping Sweep attack.

Packet data of the network(Pcap file):



Representation of how packets are received, using wireshark, when a hacker is trying to perform a Ping Sweep Attack. Number of packets for ICMP protocol are unexpectedly high from one single source IP address. This could mean the source is trying to access data and can be harmful to the destination IP.

Output:



After searching the input packets, a message is displayed for the user to be vary of the source IP address as a suspicious number of packets were received of the same protocol of ICMP. The user is required to take necessary action to prevent loss of data and privacy.

ARP Sweep Attack

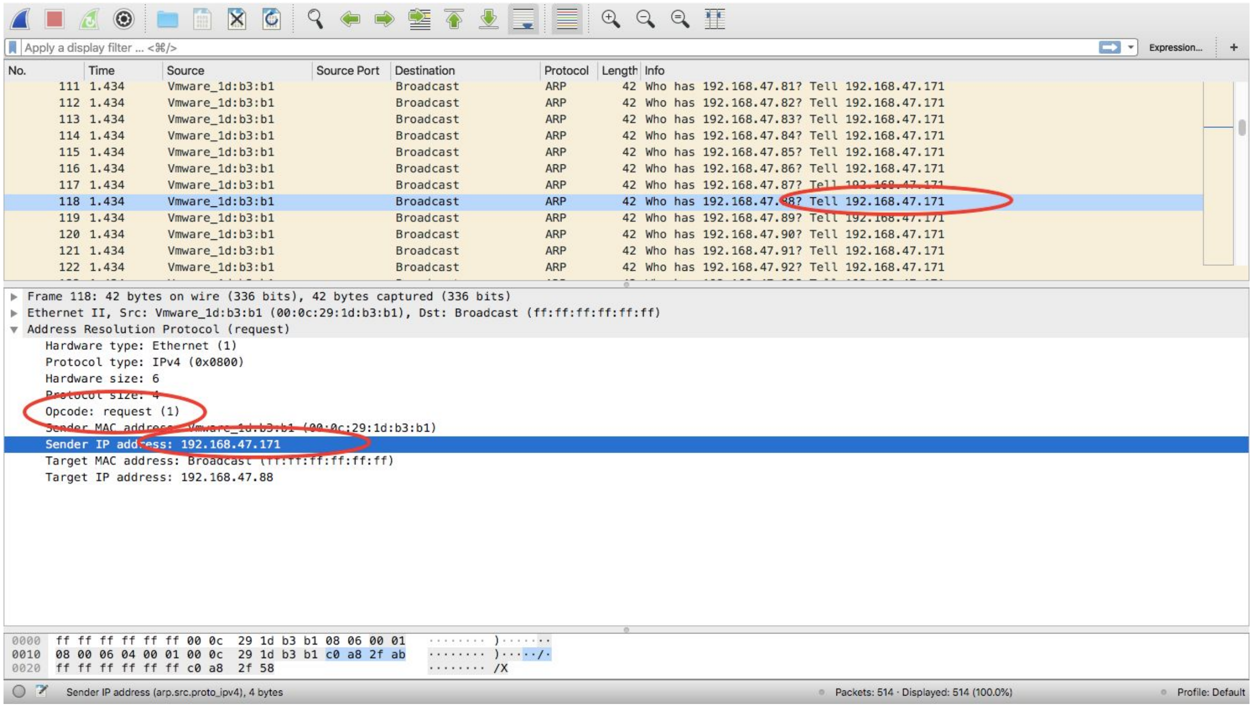
```
import pyshark
import operator

def arp_sweep():
	capture = pyshark.FileCapture('./arp_sweep.py', display_filter = "arp")
	print('CAPTURING')
	while True:
	mac_addresses={}
	arp_count=0
	under_attack = 0
	for packet in capture:
	dest_mac = packet.eth.dst
	packet_type =  str(packet.arp.opcode)
	if packet_type == '1':
		arp_count+=1
		src_mac = packet.eth.src
		if src_mac not in mac_addresses:
		mac_addresses[src_mac]=1;
		elif src_mac in mac_addresses:
		mac_addresses[src_mac]+=1
	if arp_count > 100:
	if under_attack == 0:
		print('EXCESSIVE ARP TRAFFIC DETECTED, SOURCE MAC: ',
max(mac_addresses.items(), key=operator.itemgetter(1))[0])
		return
	elif arp_count<100:
	under_attack=0

arp_sweep()
```

Live Attack Simulation of ARP Scan using NMAP:

```
gh0st@DedSec:~$ nmap -sP 10.102.164.1-50

Starting Nmap 7.60 ( https://nmap.org ) at 2019-11-06 23:25 IST
Nmap scan report for _gateway (10.102.164.1)
Host is up (0.015s latency).
Nmap scan report for 10.102.164.5
Host is up (0.016s latency).
Nmap scan report for 10.102.164.6
Host is up (0.016s latency).
Nmap scan report for 10.102.164.7
Host is up (0.016s latency).
Nmap scan report for 10.102.164.8
Host is up (0.016s latency).
Nmap scan report for 10.102.164.9
Host is up (0.016s latency).
Nmap scan report for 10.102.164.12
Host is up (0.054s latency).
Nmap scan report for 10.102.164.22
Host is up (0.0037s latency).
Nmap scan report for 10.102.164.23
Host is up (0.0030s latency).
Nmap scan report for 10.102.164.25
Host is up (0.058s latency).
Nmap scan report for 10.102.164.26
Host is up (0.014s latency).
Nmap scan report for 10.102.164.27
Host is up (0.015s latency).
Nmap scan report for 10.102.164.30
Host is up (0.074s latency).
Nmap scan report for 10.102.164.31
Host is up (0.074s latency).
Nmap scan report for 10.102.164.33
Host is up (0.090s latency).
Nmap scan report for 10.102.164.35
Host is up (0.12s latency).
Nmap scan report for 10.102.164.44
Host is up (0.25s latency).
Nmap scan report for 10.102.164.48
Host is up (0.014s latency).
Nmap done: 50 IP addresses (18 hosts up) scanned in 2.41 seconds
```

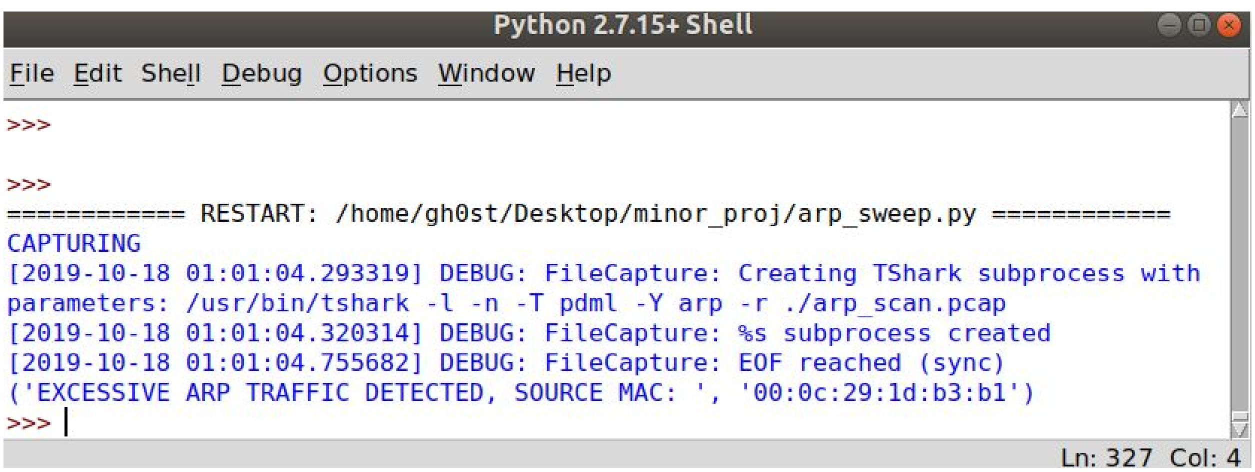
Search and detection of IP addresses of systems in a network using ARP scan in the network.

Packet data of the network:



Representation of how packets are received, using wireshark, when a hacker is trying to perform an ARP Sweep Attack. Number of packets for ARP protocol of broadcast type with opcode='1' are unexpectedly high from one single source IP address. This could mean the source is trying to access data and can be harmful to the destination IP.

Output:



After searching the input packets, a message is displayed for the user to be vary of the source IP address as a suspicious number of packets were received of the same protocol of ARP. The user is required to take necessary action to prevent loss of data and privacy.

NULL Scan

```
import pyshark
import operator

def null_scan():
	cap= pyshark.LiveCapture(interface='wlp2s0', bpf_filter='tcp')
	cap.set_debug()
	ip_add={}
	count = 0
	under_attack = 0
	while True:
	for pkt in cap.sniff_continuously(packet_count=200):
	if(str(pkt.tcp.flags)=="0x00000000"):
		count+=1
		ip=pkt.ip.src
		if ip not in ip_add:
		ip_add[ip]=1
		else:
		ip_add[ip]+=1

	if count > 100:
	print("ok")
	if under_attack == 0:
		print('EXCESSIVE NULL SCAN DETECTED, SOURCE IP: ',
max(ip_add.items(), key=operator.itemgetter(1))[0])
		return
	elif count < 100:
	under_attack=0

null_scan()
```

Live Simulation of Null Scan Attack using NMAP:

```
gh0st@DedSec:~$ sudo nmap -sN 10.102.165.34

Starting Nmap 7.60 ( https://nmap.org ) at 2019-11-30 06:08 IST
Nmap scan report for 10.102.165.34
Host is up (0.14s latency).
Not shown: 999 closed ports
PORT      STATE         SERVICE
55555/tcp open|filtered unknown
MAC Address: C0:EE:FB:54:5A:0B (OnePlus Tech (Shenzhen))

Nmap done: 1 IP address (1 host up) scanned in 217.89 seconds
```

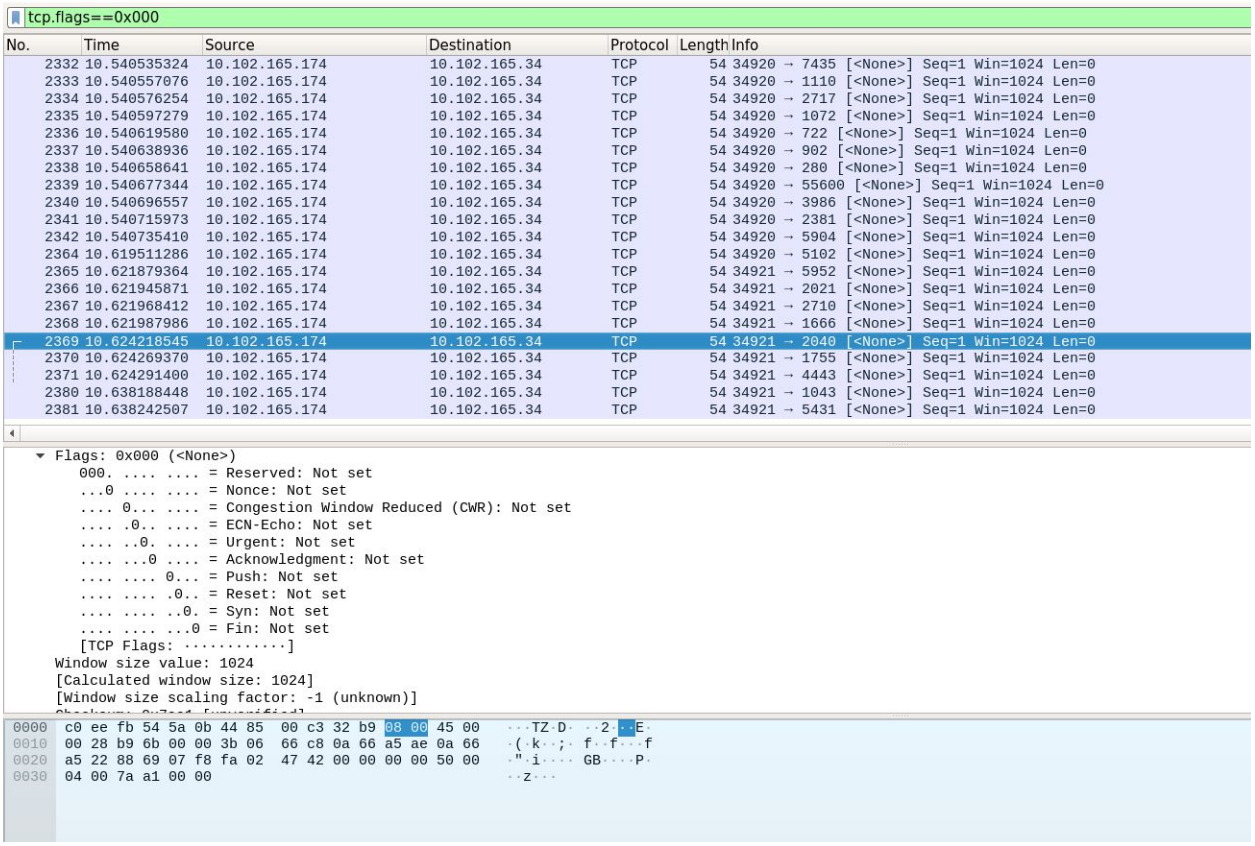
Performing the NULL Scan Attack to identify listening TCP ports. Null Scan can help identify potential holes for server hardening.

Output:

```
Python 2.7.15+ Shell
File  Edit  Shell  Debug  Options  Window  Help
========= RESTART: /home/gh0st/Desktop/minor_proj/null_scan_live.py ======
===
[2019-11-30 00:26:12.086312] DEBUG: LiveCapture: %s subprocess created
[2019-11-30 00:26:12.116801] DEBUG: LiveCapture: Creating TShark subproces
s with parameters: /usr/bin/tshark -l -n -T pdml -r -
[2019-11-30 00:26:12.120491] DEBUG: LiveCapture: %s subprocess created
ok
('EXCESSIVE NULL TCP PACKETS DETECTED, SOURCE IP: ', '10.102.165.174')
>>> 
>>>
```

After searching the input packets, a message is displayed for the user to be vary of the source IP address as a suspicious number of packets were received of the same protocol of TCP. The user is required to take necessary action to prevent loss of data and privacy.

Packets captured by Wireshark during a NULL Scan Attack:



Representation of how packets are received, using wireshark, when a hacker is trying to perform a NULL Scan Attack. Number of packets for TCP protocol with no set flag are unexpectedly high from one single source IP address. This could mean the source is trying to access data and can be harmful to the destination IP.

CONCLUSIONS

After observing numerous open networks and traffic in those networks, we came to many conclusions:

First, the network is open to a number of loopholes which can be and are exploited by few notorious users trying to invade our privacy.

Second, the IP address of any system in an open network is obtained by any user in the network by running the simplests of scripts, which can be found anywhere nowadays.

Third, without proper security to overcome these unfriendly users, we open ourselves and our data to be used by these users and even our identity through our IP and MAC addresses.

FUTURE WORK

By creating a detection software, we opened up the work for future development in cyber security and difficulties we might face in a network. We now see our future endeavour in creating a real time program to not only detect but block the malicious packets and eventually block the user who is actually using these cheap tricks to exploit the bugs in the network. A live blocking software would make logging into any network worry-free experience and would enable the user to identify the unwanted users.

REFERENCES/RESOURCES

- thePacketGeek - https://thepacketgeek.com/pyshark-using-the-capture-object/
- Network's packet data - https://asecuritysite.com/forensics/pcap?infile=ping.pcap
- Analyzing network reconnaissance attempts (by Packt)
- Detect/Analyze Scanning Traffic Using Wireshark - PenTest
- Python 2.7 Documentation - https://docs.python.org/2.7/
- Pyshark Documentation - https://github.com/KimiNewt/pyshark/
- CAPEC-304: TCP Null Scan - https://capec.mitre.org/data/definitions/304.html
- Nmap Documentation - https://nmap.org/book/scan-methods-null-fin-xmas-scan.html