

Cristian Cortez

ID: if2482

CS 471: Security & Info Assurance

Assignment 5

Abstract

In this assignment, we experiment with network scanning of an Ubuntu system conducted with a Kali VM. The purpose of this assignment is to discover open ports and the services that use them through packet analysis of the traffic generated by a network scan. This analysis is then used to understand the effectiveness of a network scanner as a Cyber Security and Network Security tool both as part of attack as well as a defense. This results in the capability of network scanners at discovery of hosts, OSs, and port status.

Introduction

A Kali VM will be used to conduct the network scan on the targeted Ubuntu host. An Ubuntu system will serve as the vulnerable victim. Ubuntu will have a disabled firewall as well as various services including: ssh, netcat listener, python server hosting a webpage. The various ports that will be exposed by the scan include 22, 8000, and 31337. These two systems are setup on bridged network that allows them to get an IP address from the router. Network scanner nmap and its companion GUI zenmap will be used to generate traffic. Wireshark will be used to capture the traffic for packet analysis.

Commands used:

NIX GENERAL

```
// get the IP address, MAC address
ip addr

// update and upgrade packages
sudo apt update
sudo apt upgrade

// search apt cache for package
sudo apt search [package]

// install a package
sudo apt install [package]
```

SSH

```
// install the openssh server
sudo apt install openssh-server

// check ssh status
sudo systemctl status ssh

// start the ssh service
sudo systemctl start ssh

// ssh to a host
ssh [user]@[ip addr]
```

NETWORK GENERAL

```
// check Ubuntu firewall status
sudo ufw status

//disable Ubuntu firewall
sudo ufw disable

// get the IP address
ip addr

// start a netcat listener on port 31337
nc -l -p 31337

// start a netcat connection
nc [ip addr] 31337
```

PYTHON SERVER

```
// host a python web server
python2 -m SimpleHTTPServer 8080
python3 -m http.server 8080
```

NMAP

```
// nmap stealthy scan and file output
sudo nmap -v -sS -A -T4 [victim ip] >> ~/dir/to/nmap.output.txt

// 1. scan with TCP connect
sudo nmap -v -sT -T4 [victim ip] >> ~/dir/to/nmap.output1.txt
```

```
// 2. scan with IP protocol
sudo nmap -v -sO -T4 [victim ip] >> ~/dir/to/nmap.output2.txt

// 3. xmas scan
sudo nmap -v -sX -T4 -p 22,31337 [victim ip] >>
~/dir/to/nmap.output3.txt
```

WIRESHARK : FILTERS

```
// search for a particular port (22, 8000, 31337) with tcp traffic
tcp.port == 31337

// search for all the port used in the assignment (22, 8000, 31337)
tcp.port in {22, 8000, 31337}

// search for all http GET methods
http.request.method == GET

// search for a tcp flag (ack, syn, fin, psh, etc.) in the packet
tcp.flags.ack
```

Summary of Results

A: Ubuntu Firewall, SSH server, Netcat Listener

1. In an Ubuntu VM, disable the Ubuntu Firewall.

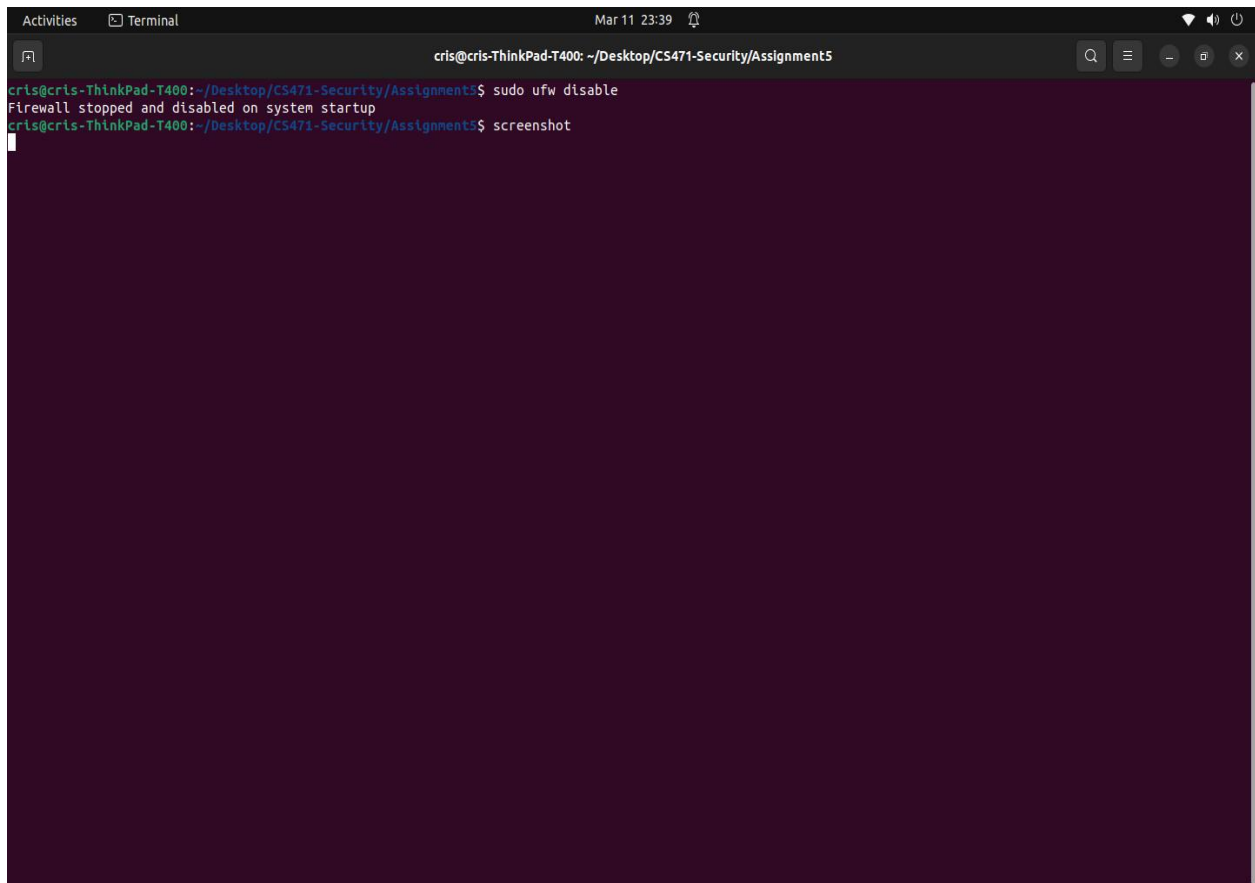
Use Commands:

```
// check firewall status
sudo ufw status

// disable firewall
sudo ufw disable
```

Disable the Ubuntu Firewall. Firewalls may have a default rule that prevents a particular port from receiving traffic. In order to scan for all potentially open ports, regardless of a firewall, we will first need to disable it.

In this way, Ubuntu will pose as a completely unprotected "victim" host.

A terminal window titled 'cris@cris-ThinkPad-T400: ~/Desktop/CS471-Security/Assignment5'. The terminal shows the command 'sudo ufw disable' being executed, with the output 'Firewall stopped and disabled on system startup'. Below this, the command 'screenshot' is entered at the prompt. The terminal has a dark purple background and white text. The window's title bar shows 'Activities', 'Terminal', and the date 'Mar 11 23:39'.

2. Install SSH, start SSH, check SSH status.

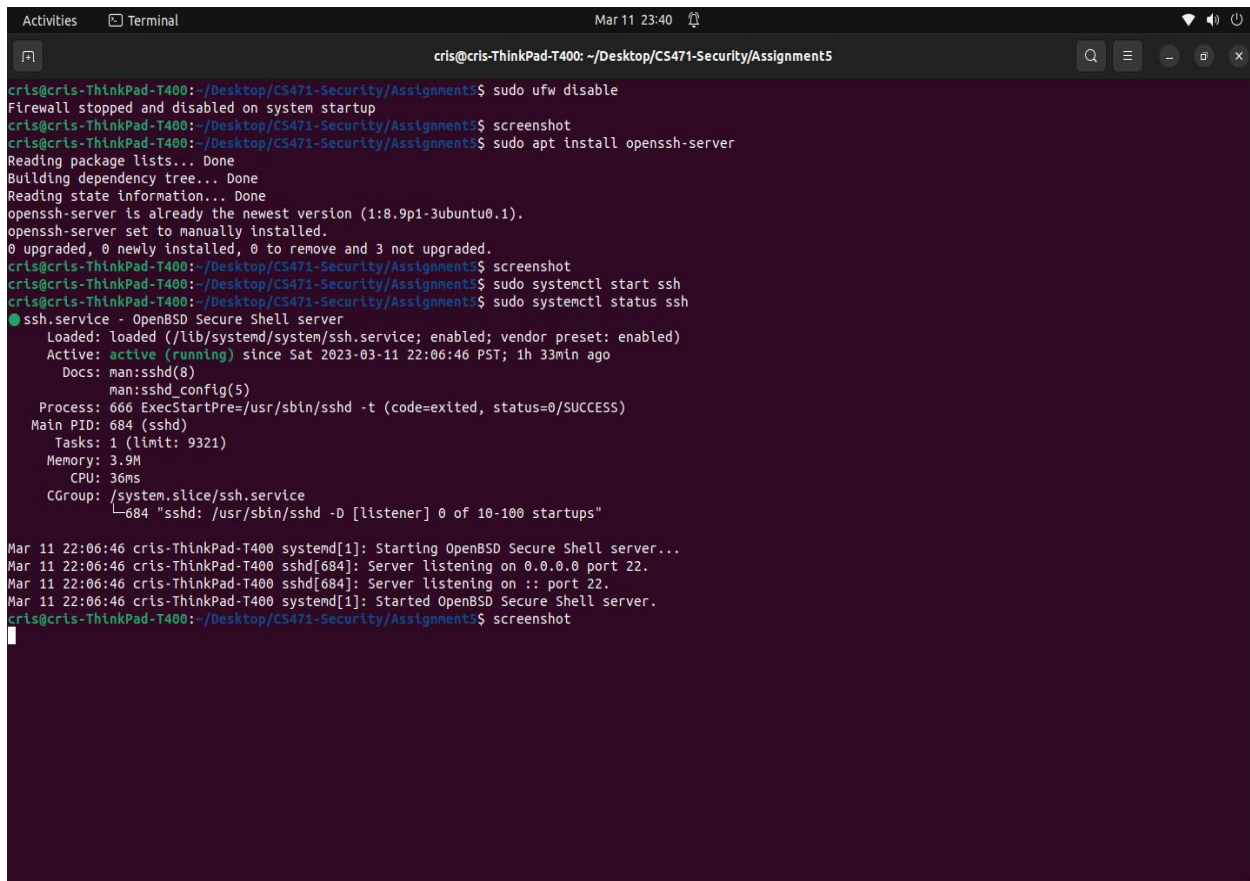
Use Commands:

```
// install the openssh server
sudo apt install openssh-server

// check ssh status
sudo systemctl status ssh

// start the ssh service
sudo systemctl start ssh
```

This will be useful in setting up a ssh server on the Ubuntu system. One reason to do this would be to allow for ssh server to start everytime the system starts. For this assignment, we will attempt to access the host through an ssh connection. SSH service will also be scanned using a port scanner.



```
cris@cris-ThinkPad-T400: ~/Desktop/CS471-Security/Assignment5
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5$ sudo ufw disable
Firewall stopped and disabled on system startup
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5$ screenshot
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openssh-server is already the newest version (1:8.9p1-3ubuntu0.1).
openssh-server set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5$ screenshot
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5$ sudo systemctl start ssh
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2023-03-11 22:06:46 PST; 1h 33min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Process: 666 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Main PID: 684 (sshd)
    Tasks: 1 (limit: 9321)
   Memory: 3.9M
      CPU: 36ms
   CGroup: /system.slice/ssh.service
           └─684 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Mar 11 22:06:46 cris-ThinkPad-T400 systemd[1]: Starting OpenBSD Secure Shell server...
Mar 11 22:06:46 cris-ThinkPad-T400 sshd[684]: Server listening on 0.0.0.0 port 22.
Mar 11 22:06:46 cris-ThinkPad-T400 sshd[684]: Server listening on :: port 22.
Mar 11 22:06:46 cris-ThinkPad-T400 systemd[1]: Started OpenBSD Secure Shell server.
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5$ screenshot
```

3. Start a Netcat Listener with IP Address using port 31337.

Use Commands:

```
// get the Ubuntu IP Address
ip addr

// In Ubuntu, use the IP address and start a netcat listener on port 31337
nc -l -p 31337

// In Kali, connect to the netcat listener
nc 192.168.0.43 31337

// Start a python webserver on port 8000
python3 -m http.server 8000
```

First, find the ip address of the Ubuntu system.

Next, start a netcat listener to listen on that address and port 31337.

```
Activities Terminal Mar 13 23:32
cris@cris-ThinkPad-T400: ~/Desktop/CS471-Security/Assignment5/webpage

cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5/webpage$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s25: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 00:27:13:b2:19:32 brd ff:ff:ff:ff:ff:ff
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:26:c6:be:3d:c0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.43/24 brd 192.168.0.255 scope global dynamic noprefixroute wlp3s0
        valid_lft 75011sec preferred_lft 75011sec
    inet6 2601:642:4a80:8540:7b99:713e:d765:eb48/64 scope global temporary dynamic
        valid_lft 342909sec preferred_lft 74500sec
    inet6 2601:642:4a80:8540:395:946a:3afe:299c/64 scope global dynamic mngtnpaddr noprefixroute
        valid_lft 342909sec preferred_lft 342909sec
    inet6 fe80::3842:222:1d21:2142/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5/webpage$ nc -l -p 31337
```

Now, in a directory with a given index.html file, start a python server to server the webpage.

NOTE: If you do not have a spare index.html file, download from online in an appropriate directory to host a python3 server.

```
Activities Terminal Mar 13 23:36
cris@cris-ThinkPad-T400: ~/Desktop/CS471-Security/Assignment5/webpage

cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5/webpage$ ls -l
total 620
drwxr-xr-x 2 cris cris 12288 Mar 13 23:21 index_files
-rw-rw-r-- 1 cris cris 619922 Mar 13 23:21 index.html
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5/webpage$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

- vulnerable with no firewall
- running several services such as:
 - netcat listener
 - python webserver

B: Setup Kali to do network scanning.

Use Commands:

Kali housekeeping before installing another network scanner (nmap and zenmap). Depending on the amount of packages needed to upgrade, this process could take 5+ mins, such as my case...

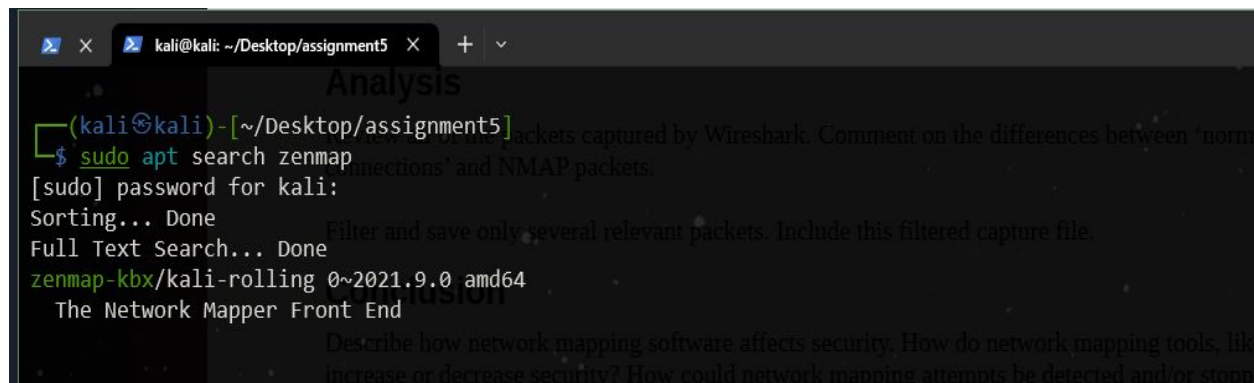
2. Search Cache for zenmap, and install it if its not already installed.

Use Commands:


```
// search for the zenmap program in cache
sudo apt-cache search zenmap

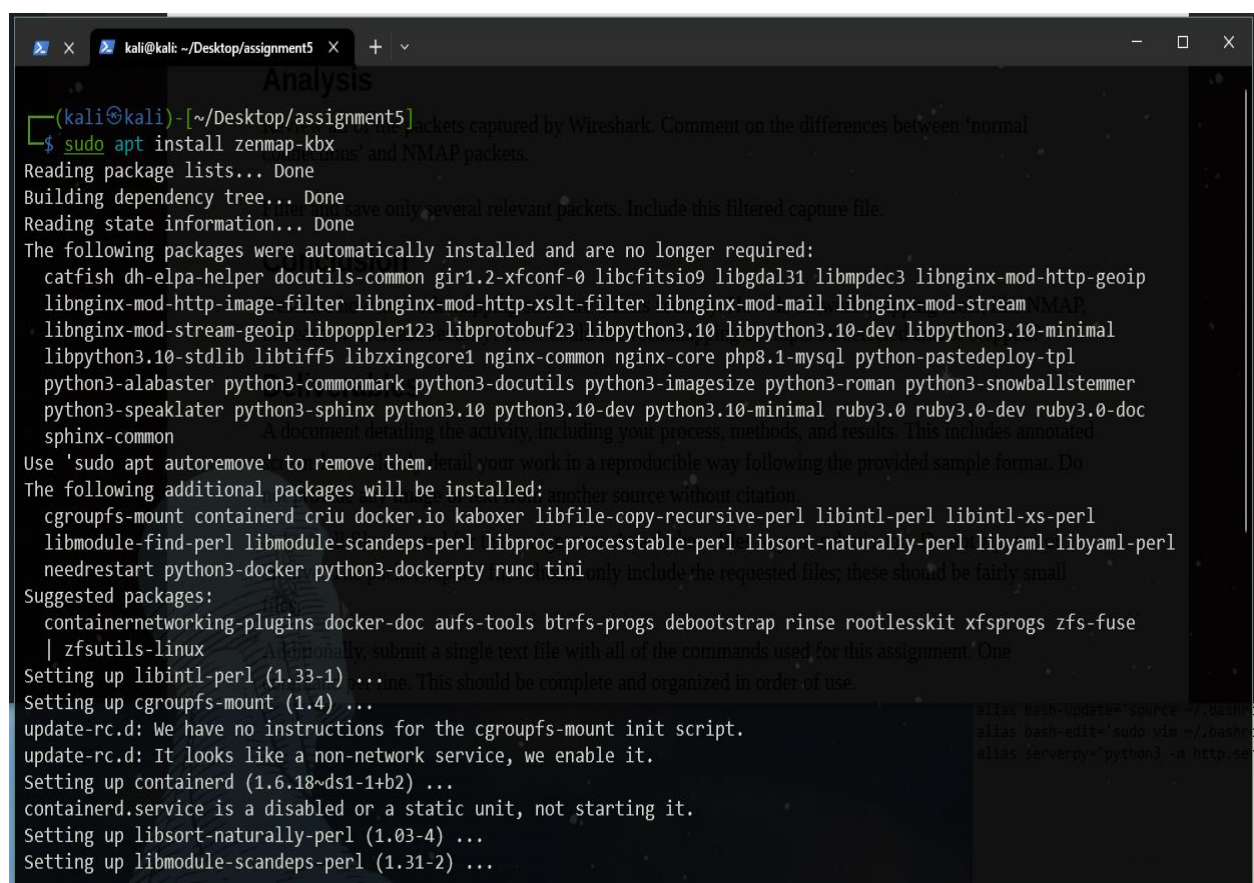
// install the package
sudo apt install zenmap-kbx
```

Simply look for the zenmap package. If you do not have a recent entry for zenmap, then install it.



```
(kali㉿kali)-[~/Desktop/assignment5]
└─$ sudo apt search zenmap
[sudo] password for kali:
Sorting... Done
Full Text Search... Done
zenmap-kbx/kali-rolling 0~2021.9.0 amd64
  The Network Mapper Front End
```

I had to install it.



```
(kali㉿kali)-[~/Desktop/assignment5]
└─$ sudo apt install zenmap-kbx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

The following packages were automatically installed and are no longer required:
catfish dh-elpa-helper docutils-common gir1.2-xfconf-0 libcfitsio9 libgdal31 libmpdec3 libnginx-mod-http-geoip
libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip libpoppler123 libprotobuf23 libpython3.10 libpython3.10-dev libpython3.10-minimal
libpython3.10-stdlib libtiff5 libzxcingcore1 nginx-common nginx-core php8.1-mysql python-pastedeploy-tpl
python3-alabaster python3-commonmark python3-docutils python3-imagesize python3-roman python3-snowballstemmer
python3-speaklater python3-sphinx python3.10 python3.10-dev python3.10-minimal ruby3.0 ruby3.0-dev ruby3.0-doc
sphinx-common
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
cgroupfs-mount containerd criu docker.io kaboxer libfile-copy-recursive-perl libint1-perl libint1-xs-perl
libmodule-find-perl libmodule-scandeps-perl libproc-processtable-perl libsort-naturally-perl libyaml-libyaml-perl
needrestart python3-docker python3-dockerpty runc tini
Suggested packages:
containernetworking-plugins docker-doc aufs-tools btrfs-progs debootstrap rinse rootlesskit xfsprogs zfs-fuse
| zfsutils-linux
Setting up libint1-perl (1.33-1) ...
Setting up cgroupfs-mount (1.4) ...
update-rc.d: We have no instructions for the cgroupfs-mount init script.
update-rc.d: It looks like a non-network service, we enable it.
Setting up containerd (1.6.18~ds1-1+b2) ...
containerd.service is a disabled or a static unit, not starting it.
Setting up libsort-naturally-perl (1.03-4) ...
Setting up libmodule-scandeps-perl (1.31-2) ...
```

Proof that its installed.

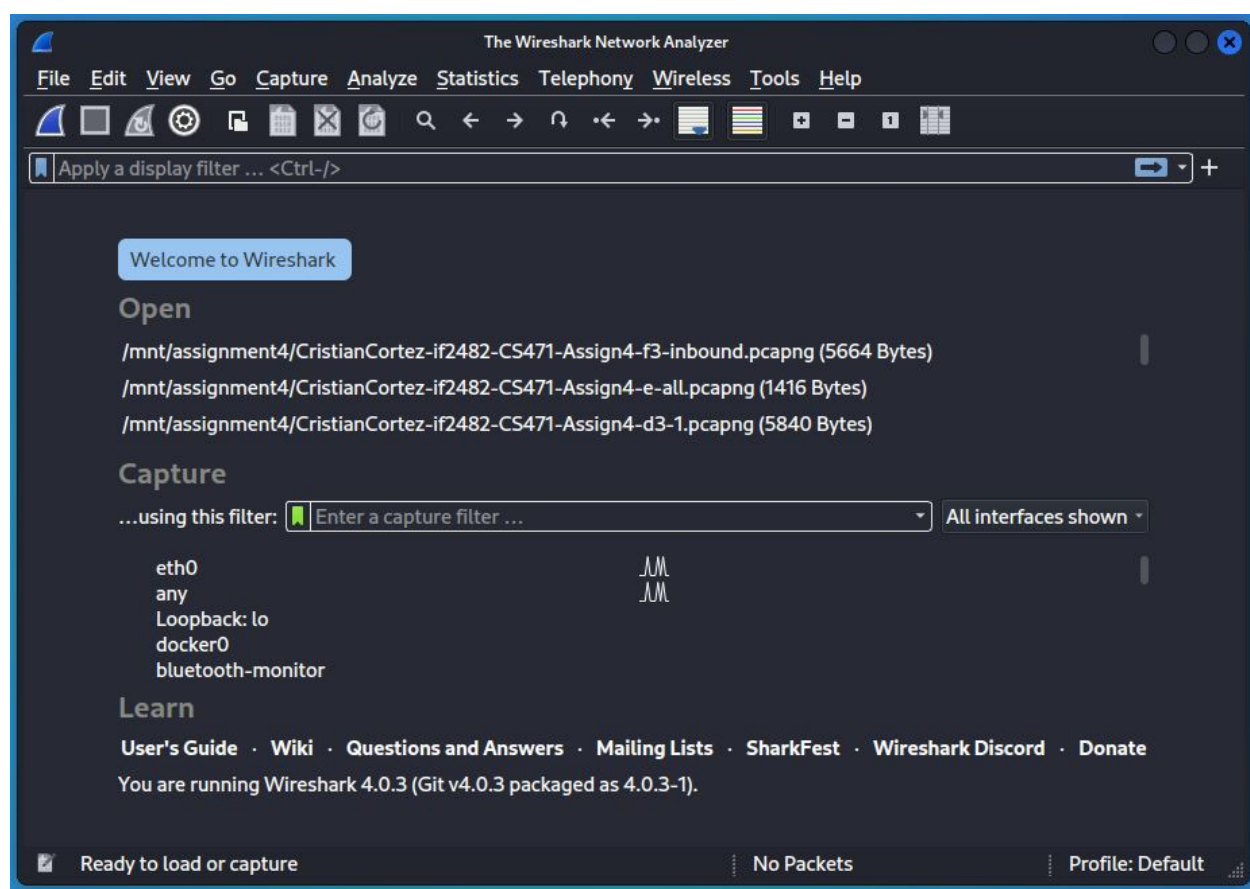

```
(kali@kali) - [~/Desktop/assignment5]
$ sudo apt search zenmap
Sorting... Done
Full Text Search... Done
zenmap-kbx/kali-rolling,now 0~2021.9.0 amd64 [installed]
  The Network Mapper Front End

(kali@kali) - [~/Desktop/assignment5]
$
```

C: Pre-scan Packet Capture

1. Start Wireshark

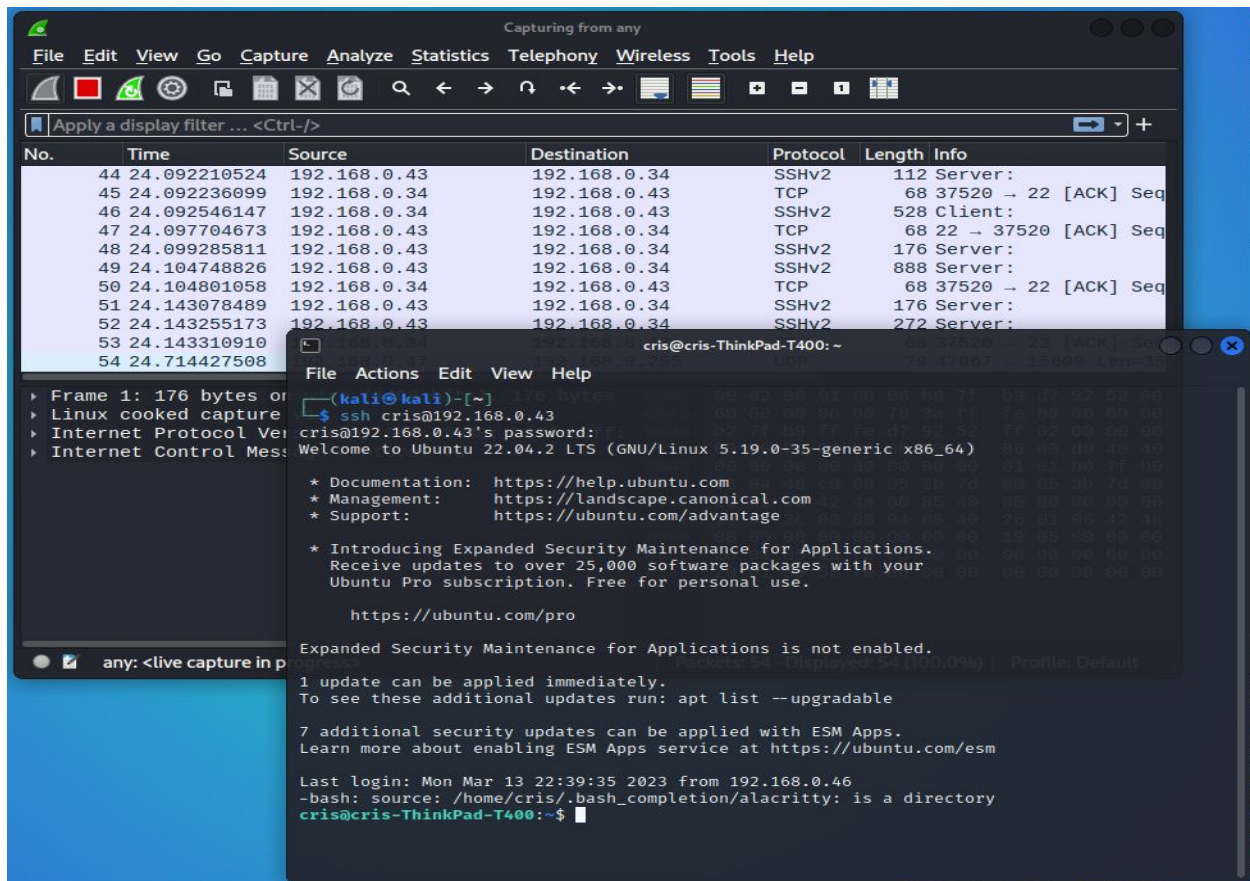
Select any adapter.



2. Connect with SSH from Kali to Ubuntu

Use Command:

```
// ssh kali to ubuntu
ssh cris@192.168.0.43
```

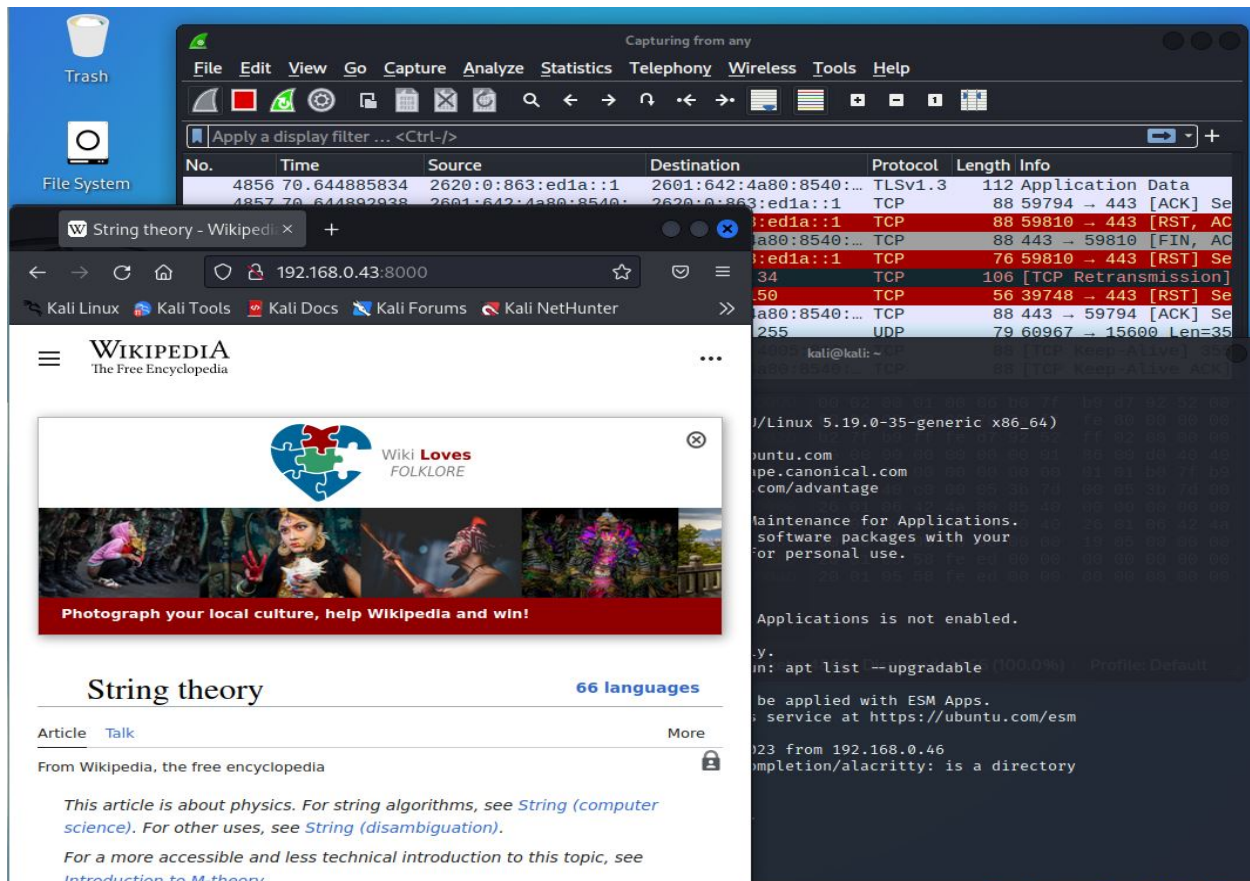


Close the session.

3. View the Ubuntu hosted webpage from Kali.

In a web browser, try view the hosted webpage at the IP address and port number specified above.

General: [IP ADDRESS]:[PORT]
 Specific: 192.168.0.43:8000



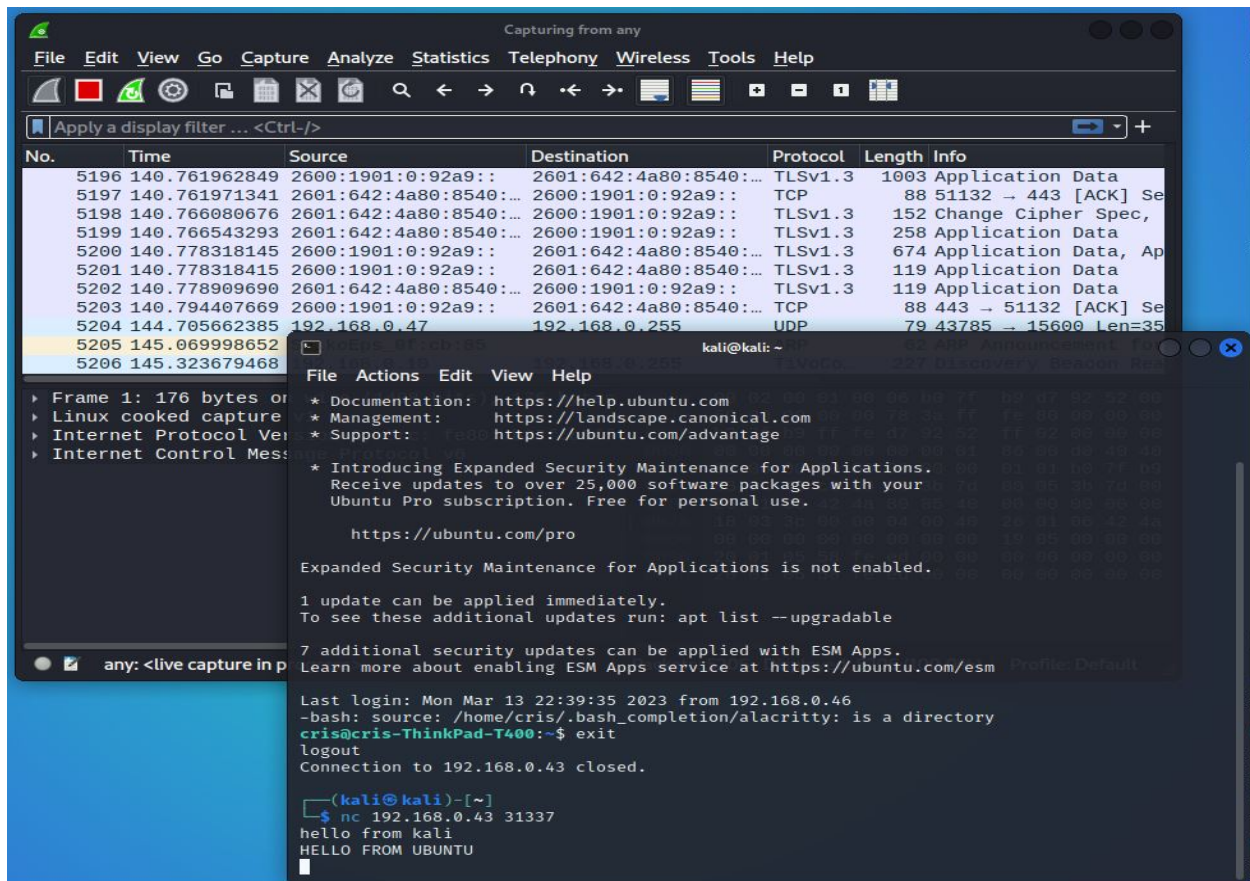
4. Attempt to connect to Ubuntu netcat listener from Kali

Use Commands:

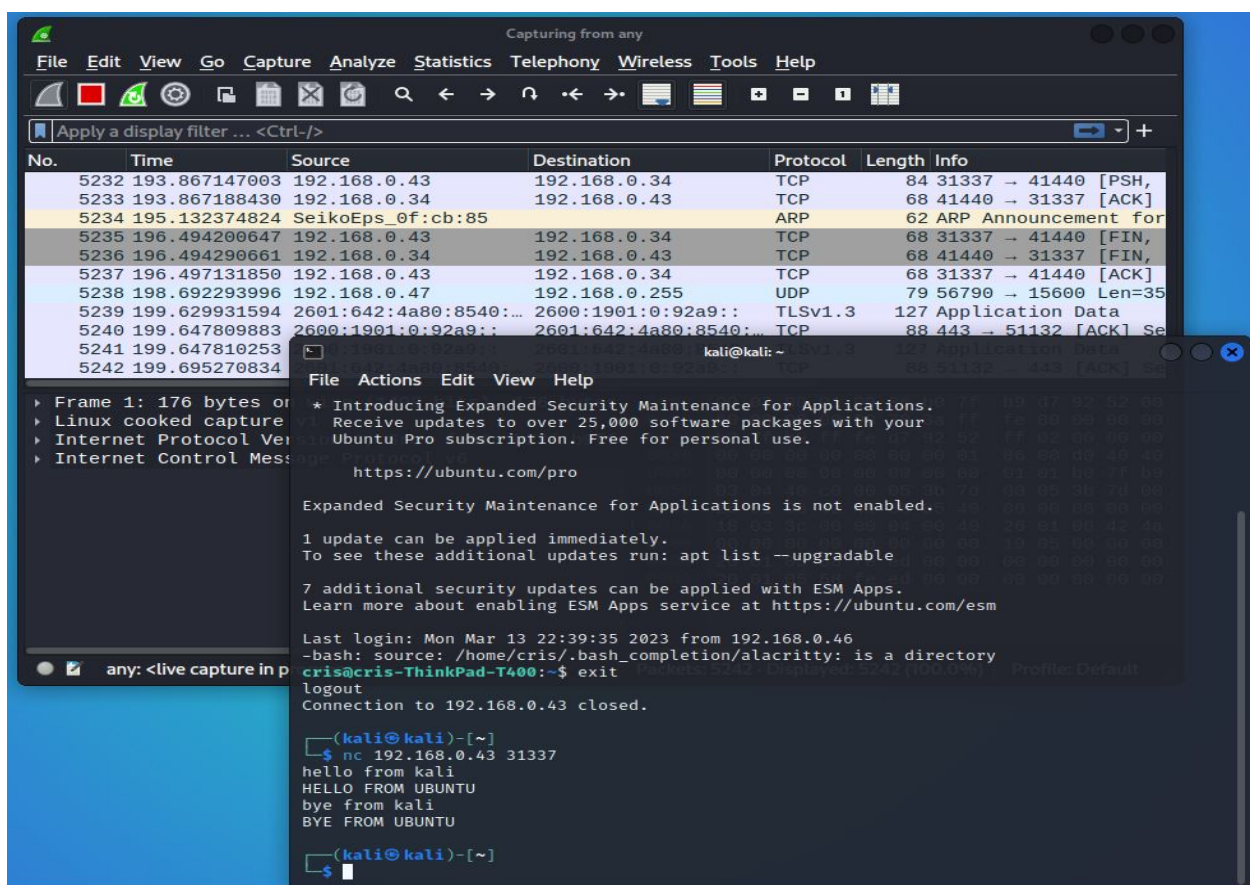
```
// kali to ubuntu listner
nc 192.168.0.43 31337
```

From kali, execute the above command to connect to the ubuntu system via a netcat connection. Once connected, send a few messages back and forth to generate some traffic.

Here is a Hello exchange:



Here is the Goodbyes...:



And here is the complete conversation from the Ubuntu point of view:

```

cris@cris-ThinkPad-T400: ~/Desktop/CS471-Security/Assignment5/webpage
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5/webpage$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s25: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 00:27:13:b2:19:32 brd ff:ff:ff:ff:ff:ff
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:26:c6:be:3d:c0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.43/24 brd 192.168.0.255 scope global dynamic noprefixroute wlp3s0
        valid_lft 75011sec preferred_lft 75011sec
    inet6 2601:642:4a80:8540:7b99:713e:d765:eb48/64 scope global temporary dynamic
        valid_lft 342909sec preferred_lft 74500sec
    inet6 2601:642:4a80:8540:395:946a:3afe:299c/64 scope global dynamic mngtnpaddr noprefixroute
        valid_lft 342909sec preferred_lft 342909sec
    inet6 fe80::3842:222:1d21:2142/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5/webpage$ nc -l -p 31337
hello from kali
HELLO FROM UBUNTU
bye from kali
BYE FROM UBUNTU
^C
cris@cris-ThinkPad-T400:~/Desktop/CS471-Security/Assignment5/webpage$

```

Close the connection once enough traffic is captured.

5. Stop Packet Capture and save pcap file.

Stop the wireshark packet capture and save the file. My file contained 5000+ packets. This will need to be trimmed later to just a few of the important ones.

D: NMAP Scan

Pre-Scan Checklist:

Ensure the following services are running on Ubuntu:

1. Ubuntu Firewall is disabled

Use Command:

```

sudo ufw status
sudo ufw disable

```

2. SSH server is running

Use Command:

```

sudo systemctl status ssh
sudo systemctl enable ssh

```

3. Netcat listener is enabled

Use Command:

```
nc -l -p 31337
```

4. Python server is running

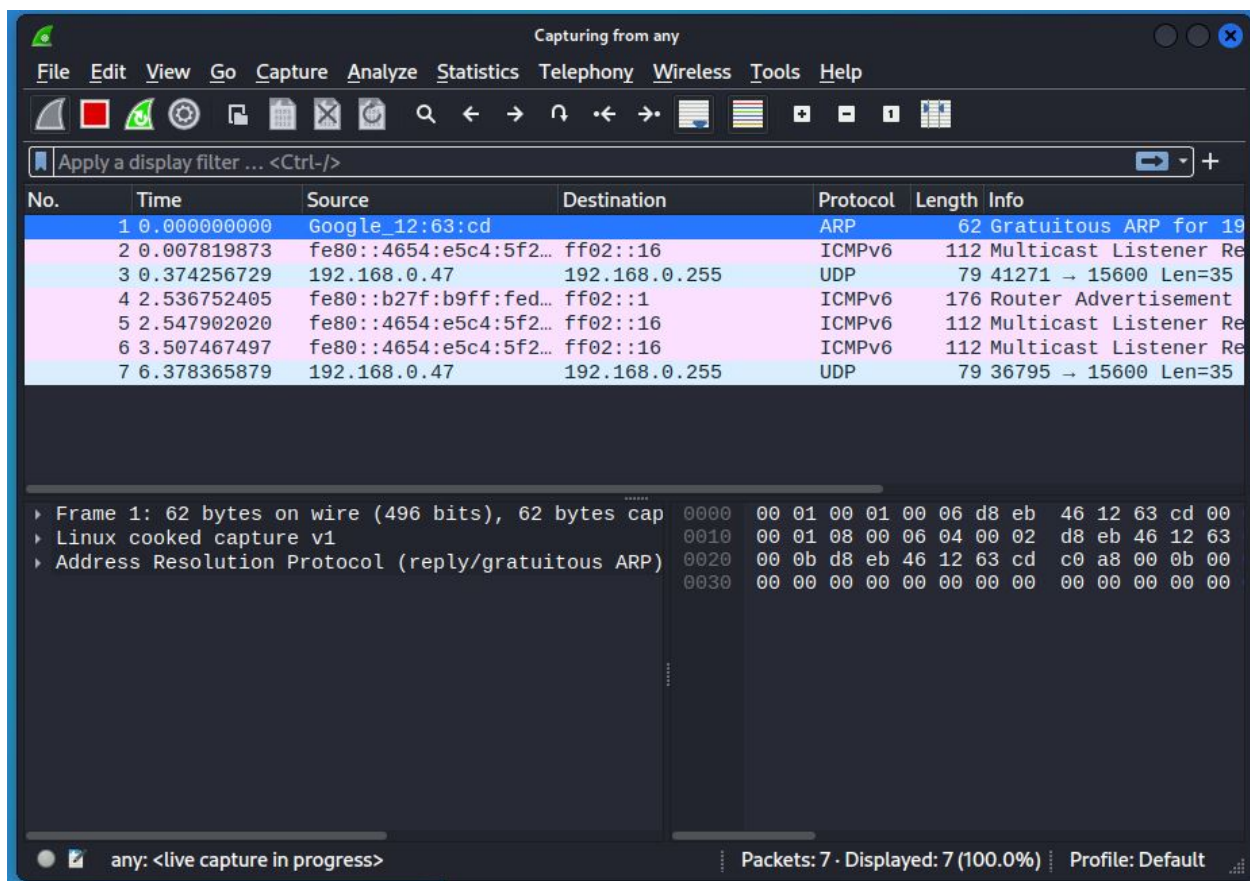
Within a directory with an *index.html*:

Use Command:

```
python3 -m http.server 8000
```

1. Start Wireshark

On Kali, Start a wireshark client and select the any adapter.

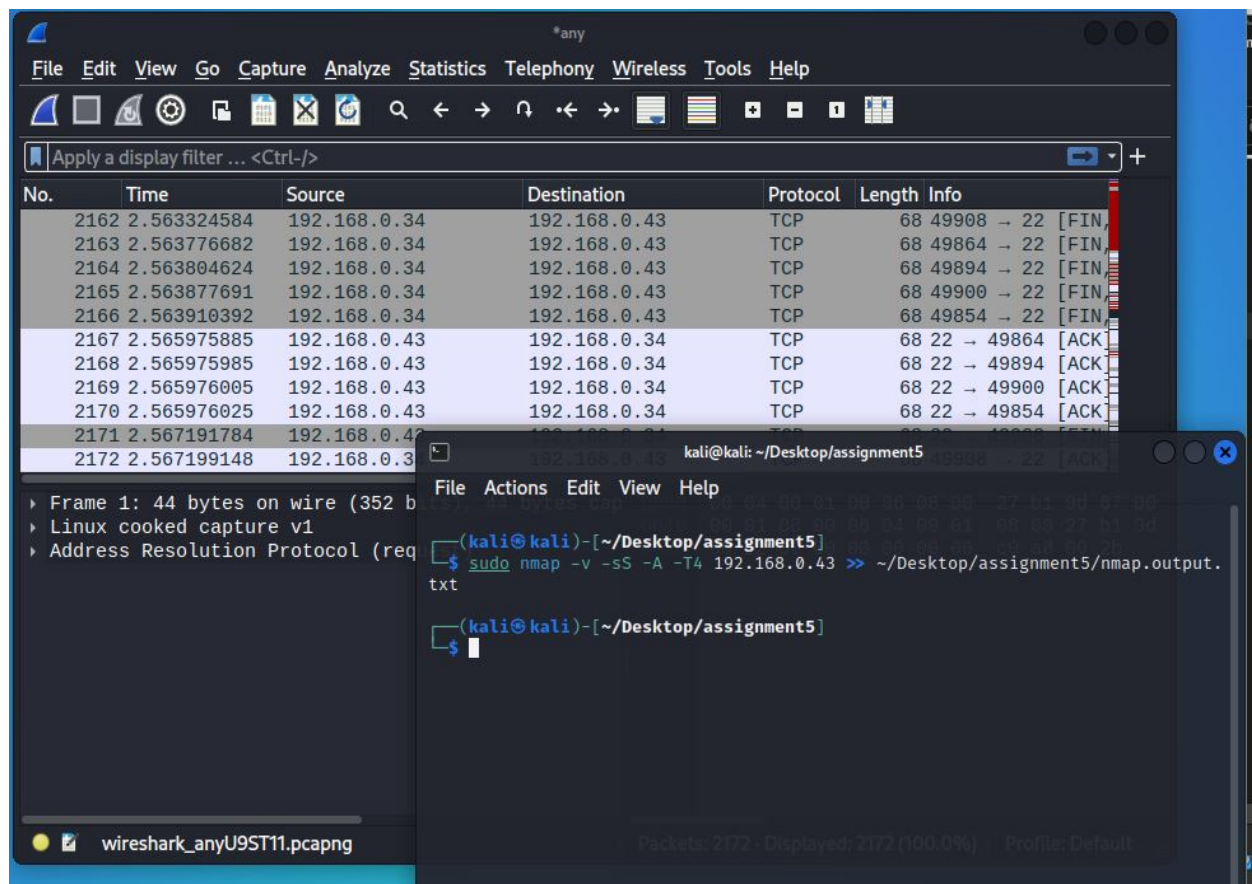


2. Start nmap scan of Ubuntu

Use Commands:


```
// nmap scan and output save
// Remember the Ubuntu IP for here
sudo nmap -v -sS -A -T4 192.168.0.43 >> ~/Desktop/assignment5/nmap.output.txt
```

Redirect all nmap output to a file for later analysis.



When the scan is finished, stop the packet capture of wireshark, save this pcap file for later analysis.

3. Run Three additional nmap scans of the Ubuntu system.

Start a wireshark client and select the any adapter.

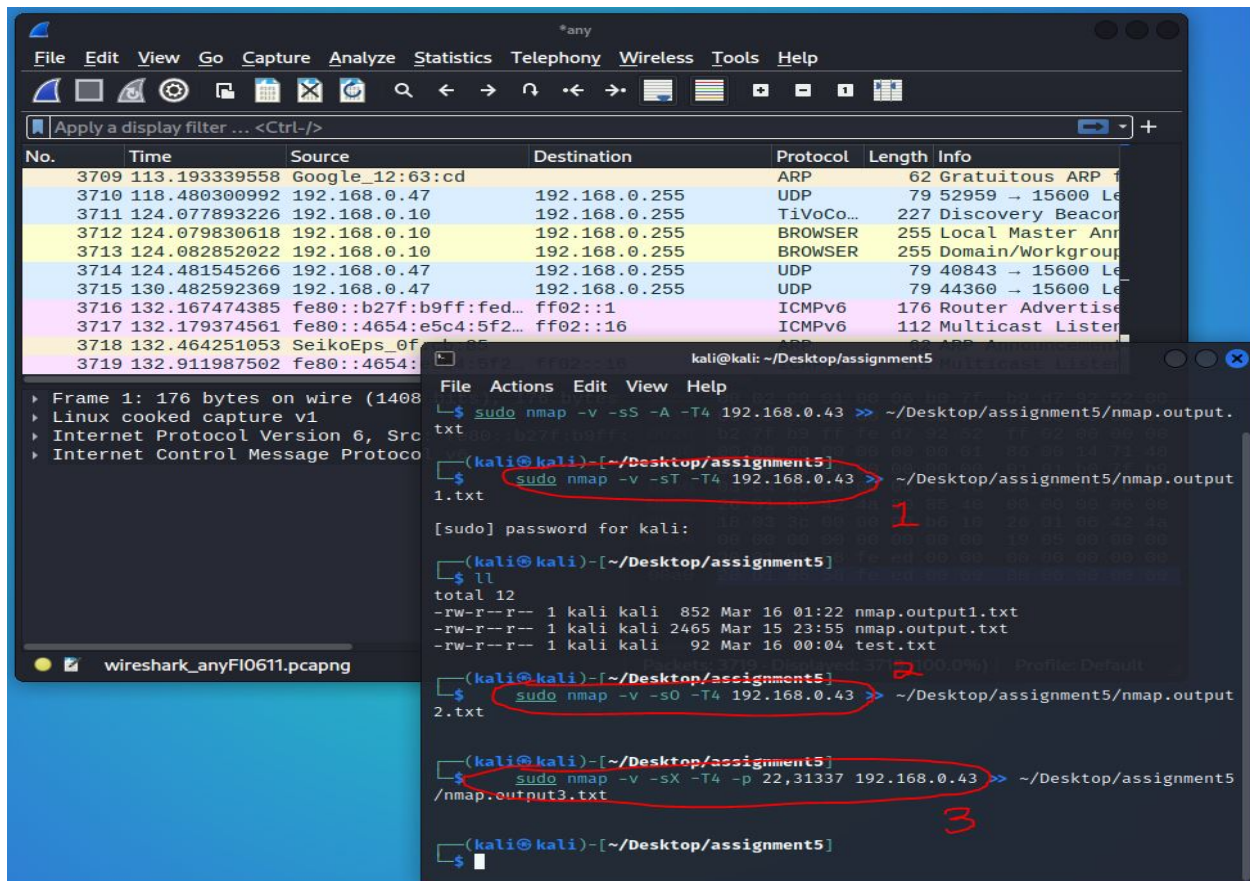
Use Commands:

```
// 1. scan with TCP connect
sudo nmap -v -sT -T4 192.168.0.43 >> ~/Desktop/assignment5/nmap.output1.txt

// 2. scan with IP protocol
sudo nmap -v -sO -T4 192.168.0.43 >> ~/Desktop/assignment5/nmap.output2.txt

// 3. xmas scan
sudo nmap -v -sX -T4 -p 22,31337 192.168.0.43 >>
~/Desktop/assignment5/nmap.output3.txt
```

When the scan is finished, stop the packet capture of wireshark, save this pcap file for later analysis.



NOTE:

Some of these scans may have broken the SSH session, the netcat instance open on port 31337, or the python server on port 8000.

E: ZENMAP Scan

Pre-Scan Checklist:

Ensure the following services are running on Ubuntu:

1. Ubuntu Firewall is disabled

Use Command:

```
sudo ufw status
sudo ufw disable
```

2. SSH server is running

Use Command:

```
sudo systemctl status ssh
sudo systemctl enable ssh
```

3. Netcat listener is enabled

Use Command:

```
nc -l -p 31337
```

4. Python server is running

Within a directory with an *index.html*:

Use Command:

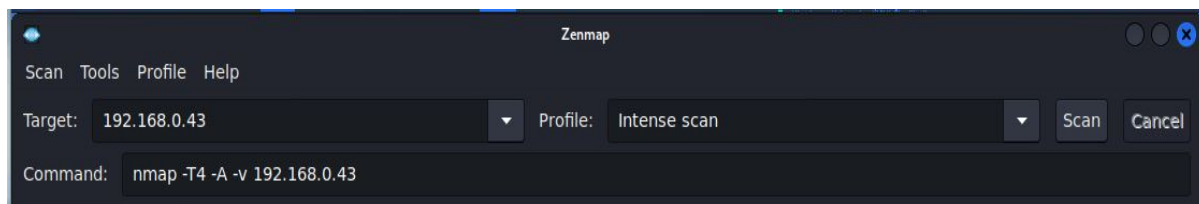
```
python3 -m http.server 8000
```

1. Start Wireshark

On Kali, Start a wireshark client and select the any adapter.

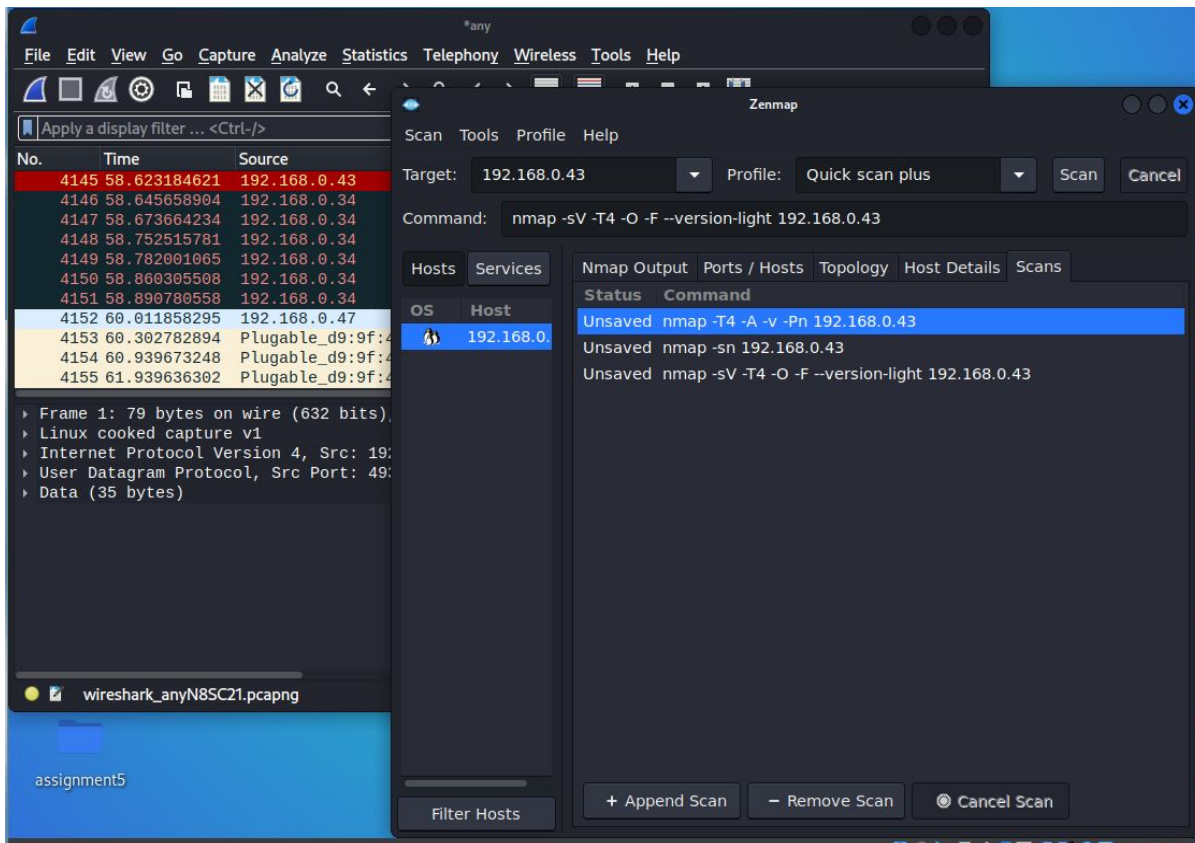
2. Run three scans using Zenmap

Open zenmap (installed earlier in section B-2). Within the "Target" box, enter the IP address for Ubuntu.



Run the following scans using zenmap:

1. Intense scan, no ping
2. Ping scan
3. Quick scan plus



Analysis

Analysis of the nmap traffic generated by this assignment is broken up into three separate analysis points:

1. normal traffic (Section C: Pre-Scan Packet Capture)
2. required nmap traffic (Section D2)
3. additional nmap traffic (Sections D3 and E)

This analysis was done using the following filters in a wireshark pcap file:

```
# search for a particular port (22, 8000, 31337) with tcp traffic
tcp.port == 31337

# search for all the port used in the assignment (22, 8000, 31337)
tcp.port in {22, 8000, 31337}

# search for all http GET methods
http.request.method == GET

# search for a tcp flag (ack, syn, fin, psh, etc.) in the packet
tcp.flags.ack
```

Normal Traffic

Most of the traffic generated by our nmap scans used TCP packets. Using the normal traffic generate without nmap, we can establish a base line for the various connection types we saw in the homework.

SSH

For SSH, use filter

```
tcp.port == 22
```

This filter allows us to see the various tcp packets sent for the ssh connection. Specifically, we are able to see the various back and forth packets that establish the agreement on what ssh version to use, the key exchange, and the generation of new keys.

Python Server

For the python server hosting the *index.html*, use filter

```
tcp.port == 8000 and http.server.method == GET
```

This filter allows for us to view the initial *GET* http method that is used to serve the webpage. Following the *GET* method, we can see various tcp packets for the index files that are used to build the webpage.

Netcat

For the netcat connection, use filter

```
tcp.port == 31337
```

This filter displays all the tcp traffic for the tcp packets sent from the netcat sender on Kali towards the netcat listener on Ubuntu. These packets are not the many. Some packets contain the unencrypted plaintext message.

Required nmap traffic

First, filter packets for the specific ports used in the assignment

```
tcp.port in {22, 8000, 31337}
```

The nmap command used the *-sS*, a "stealthy" flag that provides one half of the tcp handshake and never completes connections. From this, we can infer a large number of *[SYN]*, *[SYN,ACK]* without the final *[ACK]*.

This is what we see in the packet analysis. The *[RST]* is sent to break the connection at various times across all three of these ports. Specifically in ssh, we can see how the nmap traffic tries to establish an ssh connection, but ultimately breaks this with a *[RST]*. Nmap will repeatedly try and create this connection on port 22. After many failed attempts, there are a few packets that show nmap attempting to use an *Nmap Host Key* during the ssh establishment agreement to discover the specific ssh version being used. That is, if nmap discovers a

ssh connection, nmap will try and guess the version type. We can see this happen across various packets. Eventually, nmap find it and send various connection termination packets towards the ssh 22 port.

The majority of the packets can be seen in the traffic port 22. In fact, on ports 8000 and 31337 nmap only produced 2 packets for each ([SYN], [SYN,ACK]). This is because these port numbers belong to a general use group for serving various protocols (such as HTTP) and other general webserver hosts.

Additional nmap traffic

6 additional network scans were conducted; 3 from the nmap command line program and 3 from the nmap GUI zenmap. The scan used include:

1. TCP Connect Scan
2. IP Protocol scan
3. Xmas scan
4. Intense No ping scan
5. Ping scan
6. Quick Scan Plus

We can see the traffic for ach of these scans in the packet capture file. For example, the IP Protocol Scan "iterates over the IP protocol field" to determine what machines support which protocol (TCP, ICMP, etc.) Its not entirely a port scanner, but can be used as one. These factors produce alot of traffic in that each protocol header must be sent to the target machine. We can see in the pcap file various protocols being sent including UDP, ARP, ICMP, DNS, and TCP.

Xmas scan is sets various TCP flags in order to "light the packet up like a Christmas tree". From this, packets are able to sneak behind filtered systems such as firewalls and routers. There were not very many packets with these flags sent, however they do provide a simialr way of avoiding detection that the stealthy scan used aswell.

Conclusion

In this assignment, we used a network scanner to probe a victim Ubuntu system, with various services running, to determine what kinds of information can be analysed from the generated packet traffic. A Kali VM was used to run the network scanner nmap as well as its GUI client zenmap. Both Ubuntu and Kali were setup with Bridged network adaptors that allowed them to gain an IP Address from the host network. Wireshark was used to capture the traffic from each network scan. The conducted network scans found various ports open and the services that used them as well as other system info such as OS version and ssh connection information.

Network scanners like nmap have potential in both cyber security defense and offense. As a defense mechanism, nmap would be used to expose any vulnerabilities within a network, such as unnessary open ports and unauthorized host connections. As an offese tool, nmap could be used to learn these vulnerabilities as well as gain identifiable host information such as the various services a host is running or even its OS type.

Since the potential for nmap to be used for good is equal to its potential to do harm, I would say that it has a neutral affect on cybersecurity. Specifically, when it is used to discover network vulnerbailites, nmap increase the secruity of a network. When nmap is used to make unauthroized discovery of host information on a network, then it decreases the security of the network.I suppose then that the timing of nmap determines

whether it fully increases or decreases security. That is, if a defender can discover network vulnerabilities before an attacker can discover them, then nmap increases security.

Networking mapping attempts can be detected through network analysis of traffic. This can be done manually, as is in the use of Wireshark, or automatically such as a Firewall or Intrusion Detection/Prevention Systems (IDS/IPS). Firewalls could easily be configured to determine if network traffic is apart of normal connections or apart of bad scans.

Before describing how each tool used in this assignment provides or does not provide the X.800 Security Services, let's take a brief moment to define them.

1. *Authentication*: ensures that all parties involved in a data access or connection are who they say they are.
2. *Access Control*: the ability to limit and control access to system resources through security policies and mechanisms.
3. *Data Confidentiality*: prevents unauthorized data access.
4. *Data-Integrity*: provides assurance that total data streams remain unchanged by unauthorized entities.
5. *Non-repudiation*: protects against denial of involvement within a connection.

NMAP : Authentication, Data Confidentiality, Data Integrity

Nmap is a tool used to discover networks and the various connections that comprise the network. It does this by establishing connections with the various end-points within a network and learning identifiable information from them, then closing the connections.

First, it is important to distinguish normal connections from abnormal ones. Within a network, connections are made to provide a way for communicating information. Essentially, connections are established to transfer data from one device to another. When a connection is made, a certain amount of ID information is shared to, among various other reasons, establish trust between endpoints. Network scanners like nmap abuse this trust by only creating connections with the sole purpose of learning this ID info and not to actually transfer useful data.

Authentication is at risk by network scanners such as nmap. When a normal connection is made, identifiable information is shared between endpoints to establish a trust or authentication. Connections made with nmap serve only to learn this information, then end. Some connections, such as the stealthy `-sS` flag, only creates partial connections then ends.

Data Integrity is also at risk by these abnormal connections made with nmap. Connection integrity is a tenet within data integrity that defines a way for connections to be valid within a network. I believe that the intent to establish connection (to provide an avenue for the transfer of data and not just for learning about shared identities) is an important part of a connections integrity. Nmap breaks this integrity.

Lastly, **Data Confidentiality** is broken by nmap. That is, again, about the intent of creating connections for the sole purpose of learning ID information of the hosts that are connected on the network. While this is not "illegal", it is an abuse of the networks and the rules used to create connections. The sharing of this information is supposed to be for establishing trust, a who-knows-who of endpoints so that communications can happen. Going around a network and learning who these endpoints are and not using them for data transfers can be a malicious action.