Cristian Cortez

ID: if2482

CS 471: Security & Info Assurance

Assignment 6

Abstract

In this assignment, we create a honeypot using various netcat listeners on ports of real-world network applications such as http, https, ssh, Remote Desktop Protocol, and Python web server. The purpose of this assignment is to demonstrated the effectiveness of honeypots to gather information on the various services that they are "faking" in order to determine attackers techniques and tactics. Packet analysis determines that many information can be gathered from attackers connections with honeypot including ip address and attacking vectors (nmap). This proves that honeypots are an invaluable tool for strengthening defensive mechanisms.

Introduction

An Ubuntu system will be used to implement various netcat listeners for the purpose of creating a honeypot. The services include http, https, ssh, Remote Desktop Protocol, and Python web server running on ports 80, 443, 22, 3389, 2482 respectively. A Kali VM will be used to connect to these services in various ways including web browser, netcat, nmap and ssh. These two systems are setup on bridged network that allows them to get an IP address from the router. Wireshark will be used to capture the traffic generated by the connecting attacking system Kali so that payload anaylsis may be conducted.

Commands used:

NIX GENERAL

```
// get root shell
sudo -i

// get the IP address, MAC address
ip addr

// disable ubuntu firewall
sudo ufw disable

// diable ssh service
sudo systemctl diable ssh
```

NETCAT

```
// create a listener for ports [22,80,443, 3389, 2482]
// echo some realistic information about the service

# 1. HTTP : port 80
while true; do echo -e "HTTP/1.1 200 OK\n\n $(date)" | nc -l -p 80 -q 1 >>
$ass6/nc-http-output.txt; done
# 2. HTTPS : port 443
while true; do echo -e "HTTPS/1.1 200 OK\n\n $(date)" | nc -l -p 443 -q 1 >>
$ass6/nc-https-output.txt; done
# 3. SSH : port 22
while true; do echo -e "OpenSSH_8.9p1Ubuntu-3ubuntu0.1, OpenSSL 3.0.2 15 Mar 2022"
| nc -l -p 22 -q 1 >> $ass6/nc-ssh-output.txt; done
# 4. FTP : port 20
while true; do echo -e "HTTPS/1.1 200 OK\n\n $(date)" | nc -l -p 443 -q 1 >>
$ass6/nc-FTP-output.txt; done
# 5. SMTP : port 25
while true; do echo -e "HTTPS/1.1 200 OK\n\n $(date)" | nc -l -p 443 -q 1 >>
$ass6/nc-SMTP-output.txt; done

// create connections for ports [22,80,443, 3389, 2482]
// echo some helpful messages for logging

# 1. HTTP: Port 80
(echo "Hello HTTP, from Kali at : $(date)" | nc 192.168.0.49 80) &

# 2. HTTPS : Port 443
(echo "Hello HTTPS, from Kali at : $(date)" | nc 192.168.0.49 443) &

# 3. SSH : Port 22
(echo "Hello SSH, from Kali at : $(date)" | nc 192.168.0.49 22) &

# 4. RDP : Port 3389
(echo "Hello RDP, from Kali at : $(date)" | nc 192.168.0.49 3389) &

# 5. Python Server : Port 2482
(echo "Hello Python Server, from Kali at : $(date)" | nc 192.168.0.49 2482)
```

NMAP

```
// scan the Ubuntu host for open ports on range 1 to 65535 and push ouput
sudo nmap -p 1-65535 -T4 -A -v 192.168.0.49 >> nmap-output.txt
```

WIRESHARK : FILTERS

```
// filter the packets for only those that contain the ports and ip addr of Ubuntu
tcp.port in {80, 443, 22, 3389, 2482} && ip.addr== 192.168.0.49

// filter for ssh, http, https, etc.
ssh

// filter for ip address source
ip.source == 192.168.0.34

// filter for ip address destination
ip.dst == 192.168.0.49
```

Summary of Results

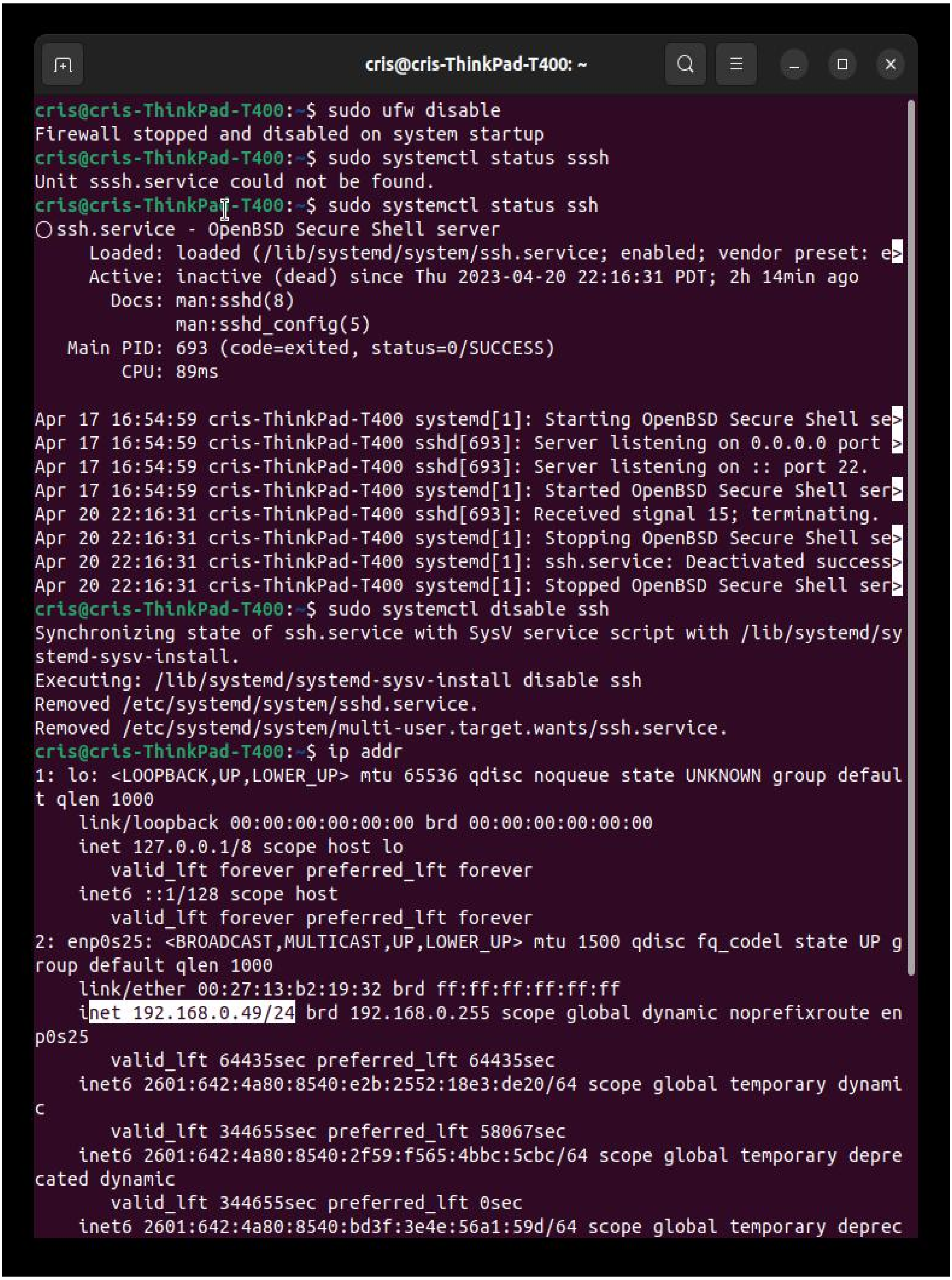
A: Ubuntu Precheck and Kali Wireshark

1. Disable firewall and ssh

Use Commands:

```
// disable ubuntu firewall
sudo ufw disable

// check status of ssh service
sudo systemctl status ssh

// disable sudo ssh service
sudo systemctl disable ssh

// get ubuntu ip
ip addr
```

This ensures that the Ubuntu system is primed for the various netcat listeners and services we will be running later on.



```
cris@cris-ThinkPad-T400:~$ sudo ufw disable
Firewall stopped and disabled on system startup
cris@cris-ThinkPad-T400:~$ sudo systemctl status sssh
Unit sssh.service could not be found.
cris@cris-ThinkPad-T400:~$ sudo systemctl status ssh
○ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: e>
     Active: inactive (dead) since Thu 2023-04-20 22:16:31 PDT; 2h 14min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 693 (code=exited, status=0/SUCCESS)
        CPU: 89ms

Apr 17 16:54:59 cris-ThinkPad-T400 systemd[1]: Starting OpenBSD Secure Shell se>
Apr 17 16:54:59 cris-ThinkPad-T400 sshd[693]: Server listening on 0.0.0.0 port >
Apr 17 16:54:59 cris-ThinkPad-T400 sshd[693]: Server listening on :: port 22.
Apr 17 16:54:59 cris-ThinkPad-T400 systemd[1]: Started OpenBSD Secure Shell ser>
Apr 20 22:16:31 cris-ThinkPad-T400 sshd[693]: Received signal 15; terminating.
Apr 20 22:16:31 cris-ThinkPad-T400 systemd[1]: Stopping OpenBSD Secure Shell se>
Apr 20 22:16:31 cris-ThinkPad-T400 systemd[1]: ssh.service: Deactivated success>
Apr 20 22:16:31 cris-ThinkPad-T400 systemd[1]: Stopped OpenBSD Secure Shell ser>
cris@cris-ThinkPad-T400:~$ sudo systemctl disable ssh
Synchronizing state of ssh.service with SysV service script with /lib/systemd/sy
stemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable ssh
Removed /etc/systemd/system/sshd.service.
Removed /etc/systemd/system/multi-user.target.wants/ssh.service.
cris@cris-ThinkPad-T400:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 00:27:13:b2:19:32 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.49/24 brd 192.168.0.255 scope global dynamic noprefixroute en
p0s25
       valid_lft 64435sec preferred_lft 64435sec
    inet6 2601:642:4a80:8540:e2b:2552:18e3:de20/64 scope global temporary dynami
c
       valid_lft 344655sec preferred_lft 58067sec
    inet6 2601:642:4a80:8540:2f59:f565:4bbc:5cbc/64 scope global temporary depre
cated dynamic
       valid_lft 344655sec preferred_lft 0sec
    inet6 2601:642:4a80:8540:bd3f:3e4e:56a1:59d/64 scope global temporary deprec
```

***2. Kali: Start wireshark ***

In Kali, start wireshark and select any adapter.

B: Ubuntu Services

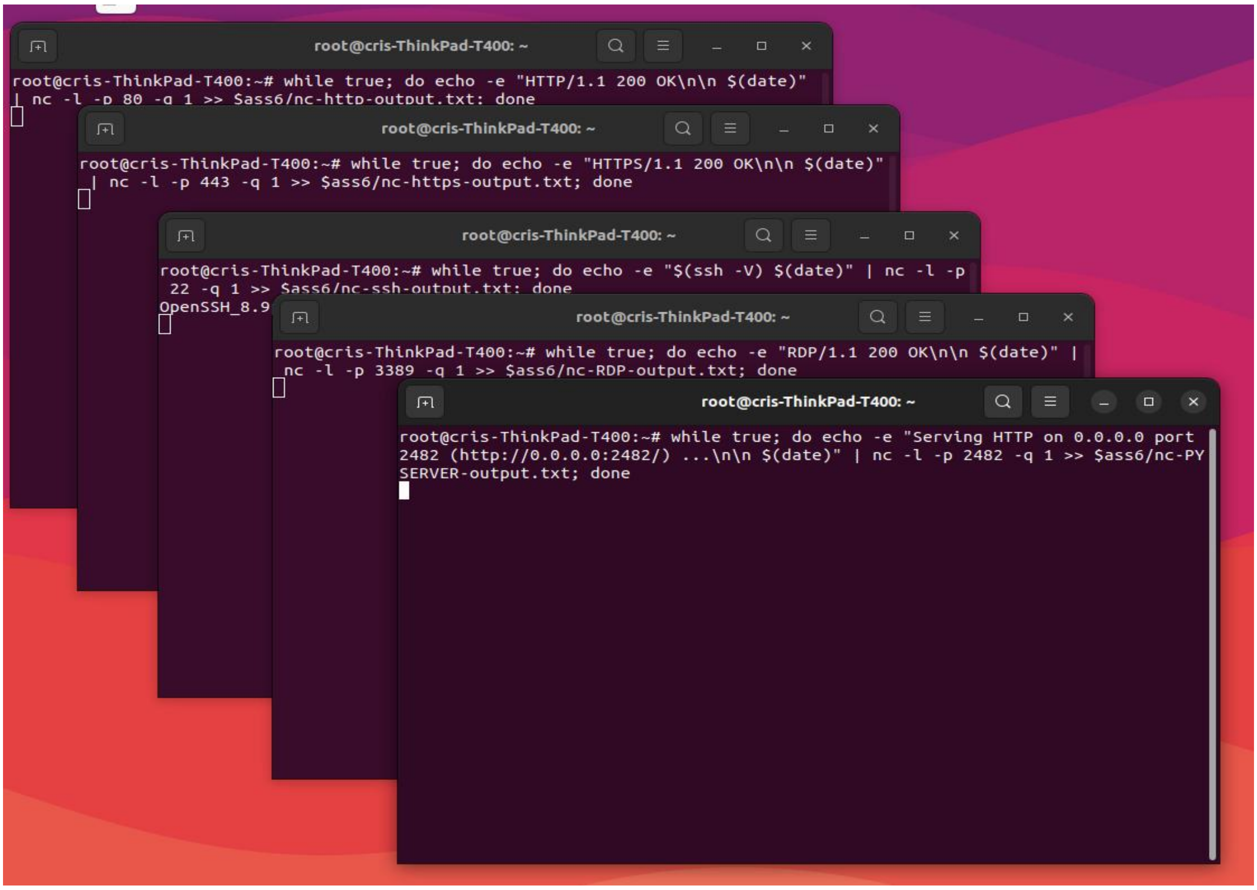
Use Commands:

```
# get a root terminal
sudo -i

# Folder for the assignment output files
# Change this to the correct folder.
ass6='/home/cris/Desktop/CS471-Security/Assignment6/outputs'

# 1. HTTP : port 80
while true; do echo -e "HTTP/1.1 200 OK\n\n $(date)" | nc -l -p 80 -q 1 >> $ass6/nc-http-output.txt; done
# 2. HTTPS : port 443
while true; do echo -e "HTTPS/1.1 200 OK\n\n $(date)" | nc -l -p 443 -q 1 >> $ass6/nc-https-output.txt; done
# 3. SSH : port 22
while true; do echo -e "$(ssh -V) $(date)" | nc -l -p 22 -q 1 >> $ass6/nc-ssh-output.txt; done
# 4. Remote Desktop Protocol : port 3389
while true; do echo -e "RDP/1.1 200 OK\n\n $(date)" | nc -l -p 3389 -q 1 >> $ass6/nc-FTP-output.txt; done
# 5. Python Server : port 2482
while true; do echo -e "Serving HTTP on 0.0.0.0 port 2482 (http://0.0.0.0:2482 ...\n\n$(date)" | nc -l -p 2482 -q 1 >> $ass6/nc-SMTP-output.txt; done
```

On the Ubuntu system, open 5 separate terminals. In each terminal, run `sudo -i` to create a root shell.

Next, in each terminal, include the variable pointing toward the directory for the output files.

In each terminal, run one of the 5 commands listed above. Each one is a netcat listener on a port that would simulate a real service. For example, the listener on port 22, used by an ssh service, simulates ssh by displaying the ssh version info.



```
root@cris-ThinkPad-T400:~# while true; do echo -e "HTTP/1.1 200 OK\n\n $(date)"
| nc -l -p 80 -q 1 >> $ass6/nc-http-output.txt; done
```

```
root@cris-ThinkPad-T400:~# while true; do echo -e "HTTPS/1.1 200 OK\n\n $(date)"
 | nc -l -p 443 -q 1 >> $ass6/nc-https-output.txt; done
```

```
root@cris-ThinkPad-T400:~# while true; do echo -e "$(ssh -V) $(date)" | nc -l -p
 22 -q 1 >> $ass6/nc-ssh-output.txt; done
OpenSSH_8.9
```

```
root@cris-ThinkPad-T400:~# while true; do echo -e "RDP/1.1 200 OK\n\n $(date)" |
 nc -l -p 3389 -q 1 >> $ass6/nc-RDP-output.txt; done
```

```
root@cris-ThinkPad-T400:~# while true; do echo -e "Serving HTTP on 0.0.0.0 port
2482 (http://0.0.0.0:2482/) ...\n\n $(date)" | nc -l -p 2482 -q 1 >> $ass6/nc-PY
SERVER-output.txt; done
```

C: Kali Connections

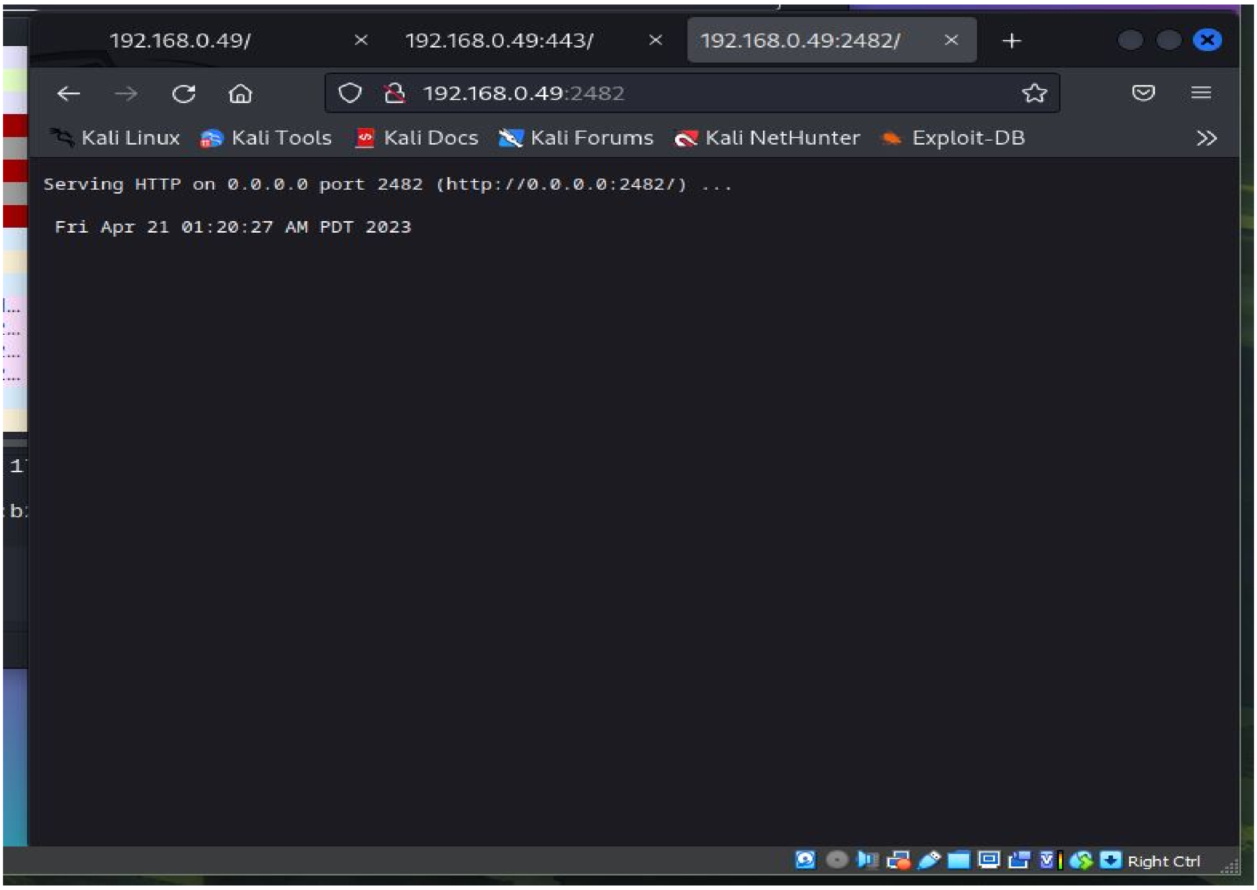
1. Web Browser Connection

within Kali, start a web browser. Navigate to the following:

```
192.168.0.49:80
192.168.0.49:443
192.168.0.49:2482
```

Refresh each a couple of times. This will populate the output files with some content.

This screenshot shows three separate tabs open wth the ports 80, 443, 2482.



```
Serving HTTP on 0.0.0.0 port 2482 (http://0.0.0.0:2482/) ...

 Fri Apr 21 01:20:27 AM PDT 2023
```

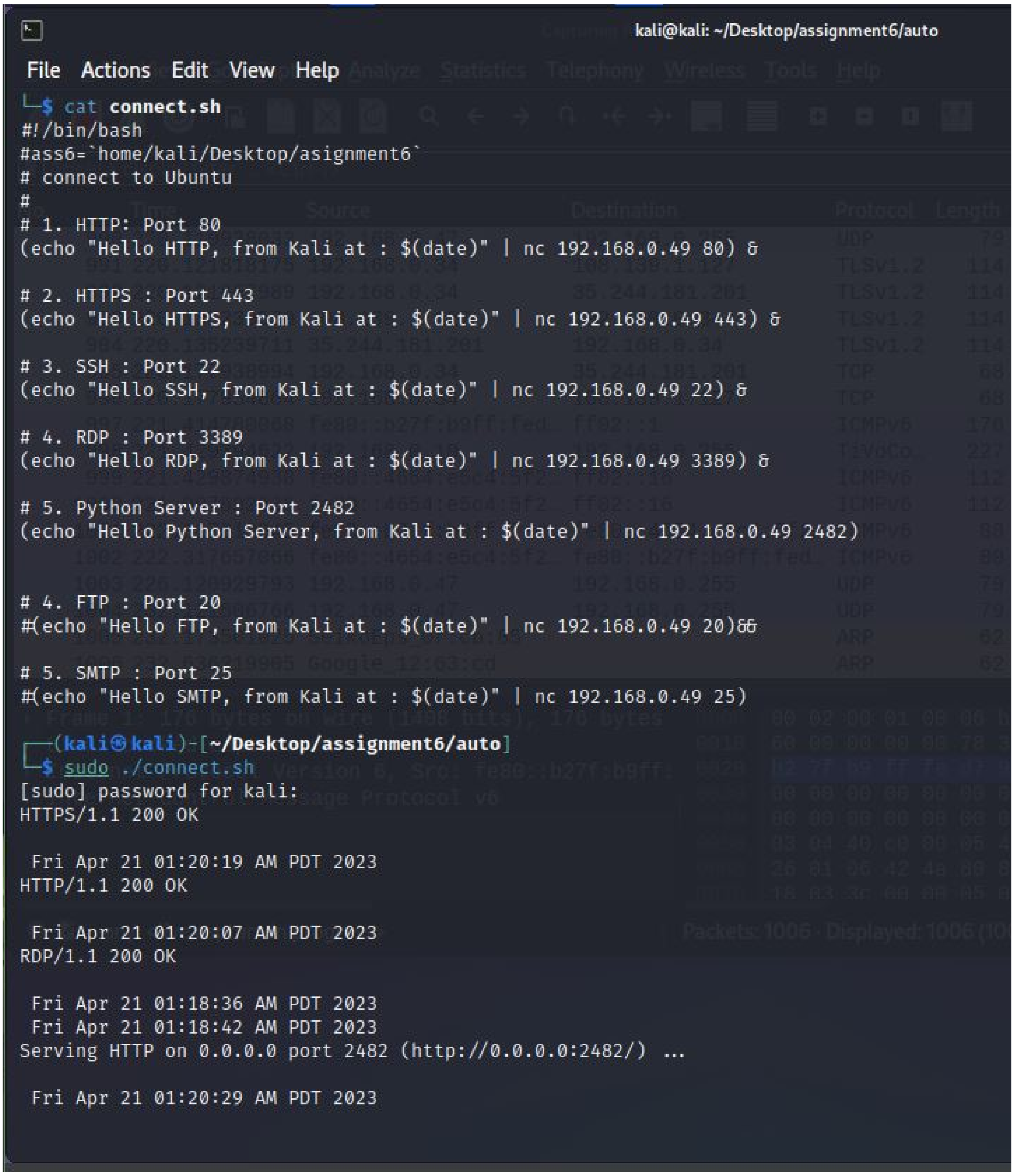
2. Netcat Connection

Use Script file `connect.sh`:

```
# 1. HTTP: Port 80
(echo "Hello HTTP, from Kali at : $(date)" | nc 192.168.0.49 80) &

# 2. HTTPS : Port 443
(echo "Hello HTTPS, from Kali at : $(date)" | nc 192.168.0.49 443) &

# 3. SSH : Port 22
(echo "Hello SSH, from Kali at : $(date)" | nc 192.168.0.49 22) &

# 4. RDP : Port 3389
(echo "Hello RDP, from Kali at : $(date)" | nc 192.168.0.49 3389) &

# 5. Python Server : Port 2482
(echo "Hello Python Server, from Kali at : $(date)" | nc 192.168.0.49 2482)
```

```
# Make script exutable
sudo chmod +x connect.sh

# Run the script
sudo ./connect.sh
```

This script will run all the netcat connections and send a personalized message to the listeners. Within the Ubuntu logs, we will be able to see these messages, confiriming that a connection was made.


kali@kali: ~/Desktop/assignment6/auto
File Actions Edit View Help

```
└─$ cat connect.sh
#!/bin/bash
#ass6=`home/kali/Desktop/asignment6`
# connect to Ubuntu
#
# 1. HTTP: Port 80
(echo "Hello HTTP, from Kali at : $(date)" | nc 192.168.0.49 80) &

# 2. HTTPS : Port 443
(echo "Hello HTTPS, from Kali at : $(date)" | nc 192.168.0.49 443) &

# 3. SSH : Port 22
(echo "Hello SSH, from Kali at : $(date)" | nc 192.168.0.49 22) &

# 4. RDP : Port 3389
(echo "Hello RDP, from Kali at : $(date)" | nc 192.168.0.49 3389) &

# 5. Python Server : Port 2482
(echo "Hello Python Server, from Kali at : $(date)" | nc 192.168.0.49 2482)


# 4. FTP : Port 20
#(echo "Hello FTP, from Kali at : $(date)" | nc 192.168.0.49 20)&&

# 5. SMTP : Port 25
#(echo "Hello SMTP, from Kali at : $(date)" | nc 192.168.0.49 25)

┌──(kali㉿kali)-[~/Desktop/assignment6/auto]
└─$ sudo ./connect.sh
[sudo] password for kali:
HTTPS/1.1 200 OK

 Fri Apr 21 01:20:19 AM PDT 2023
HTTP/1.1 200 OK

 Fri Apr 21 01:20:07 AM PDT 2023
RDP/1.1 200 OK

 Fri Apr 21 01:18:36 AM PDT 2023
 Fri Apr 21 01:18:42 AM PDT 2023
Serving HTTP on 0.0.0.0 port 2482 (http://0.0.0.0:2482/) ...

 Fri Apr 21 01:20:29 AM PDT 2023
```

3. Nmap Connection

Use Commands:

```
sudo nmap -p 1-65535 -T4 -A -v 192.168.0.49 >> nmap-output.txt
```

Nmap will scan the Ubuntu host and attempt to find open ports between the range provided. It will then save this output to the a text file.

4. Other Connections

RDP and SSH both have services that require a client setup.

RDP require the sue of a remote desktop client. Not only, but also requires the infrstructure for a remote desktop server side, i.e. Ubuntu. I did not set this up.

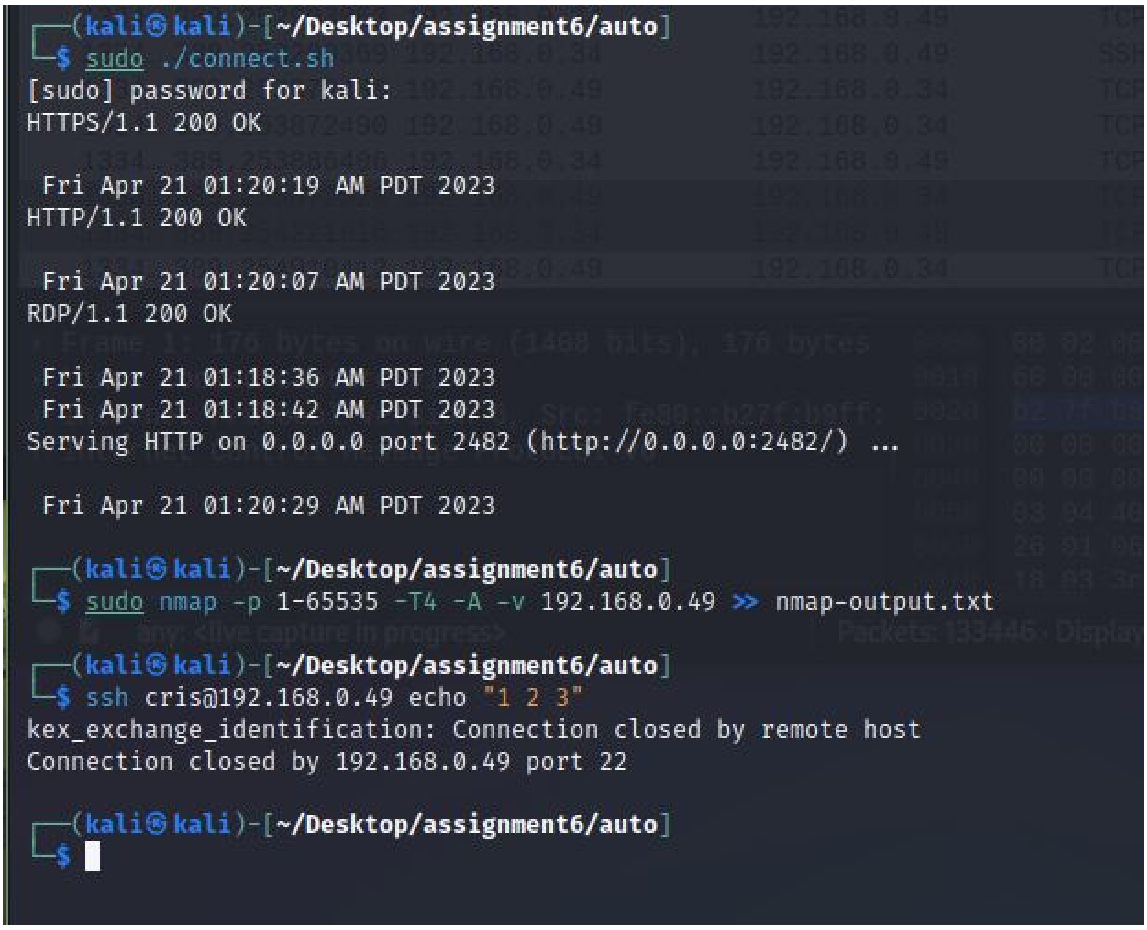
SSH can be easily demonstrated by attempting to create a ssh session from Kali.

Use Commands:

```
ssh cris@192.168.0.49
```

This screen shot deonstrates

- netcat connect script
- nmap scan
- ssh



```
┌──(kali㉿kali)-[~/Desktop/assignment6/auto]
└─$ sudo ./connect.sh
[sudo] password for kali:
HTTPS/1.1 200 OK

 Fri Apr 21 01:20:19 AM PDT 2023
HTTP/1.1 200 OK

 Fri Apr 21 01:20:07 AM PDT 2023
RDP/1.1 200 OK

 Fri Apr 21 01:18:36 AM PDT 2023
 Fri Apr 21 01:18:42 AM PDT 2023
Serving HTTP on 0.0.0.0 port 2482 (http://0.0.0.0:2482/) ...

 Fri Apr 21 01:20:29 AM PDT 2023

┌──(kali㉿kali)-[~/Desktop/assignment6/auto]
└─$ sudo nmap -p 1-65535 -T4 -A -v 192.168.0.49 >> nmap-output.txt

┌──(kali㉿kali)-[~/Desktop/assignment6/auto]
└─$ ssh cris@192.168.0.49 echo "1 2 3"
kex_exchange_identification: Connection closed by remote host
Connection closed by 192.168.0.49 port 22

┌──(kali㉿kali)-[~/Desktop/assignment6/auto]
└─$ 
```

Analysis

The main purpose of this assignment was to simulate attacks on a honeypot using basic netcat listeners and various connection styles. These styles include:

- web browser
- netcat connectors
- nmap port scanner
- ssh session

From packet analysis we learn various information regarding port numbers and source/destination IP addresses. However, we also learn about packet payloads.

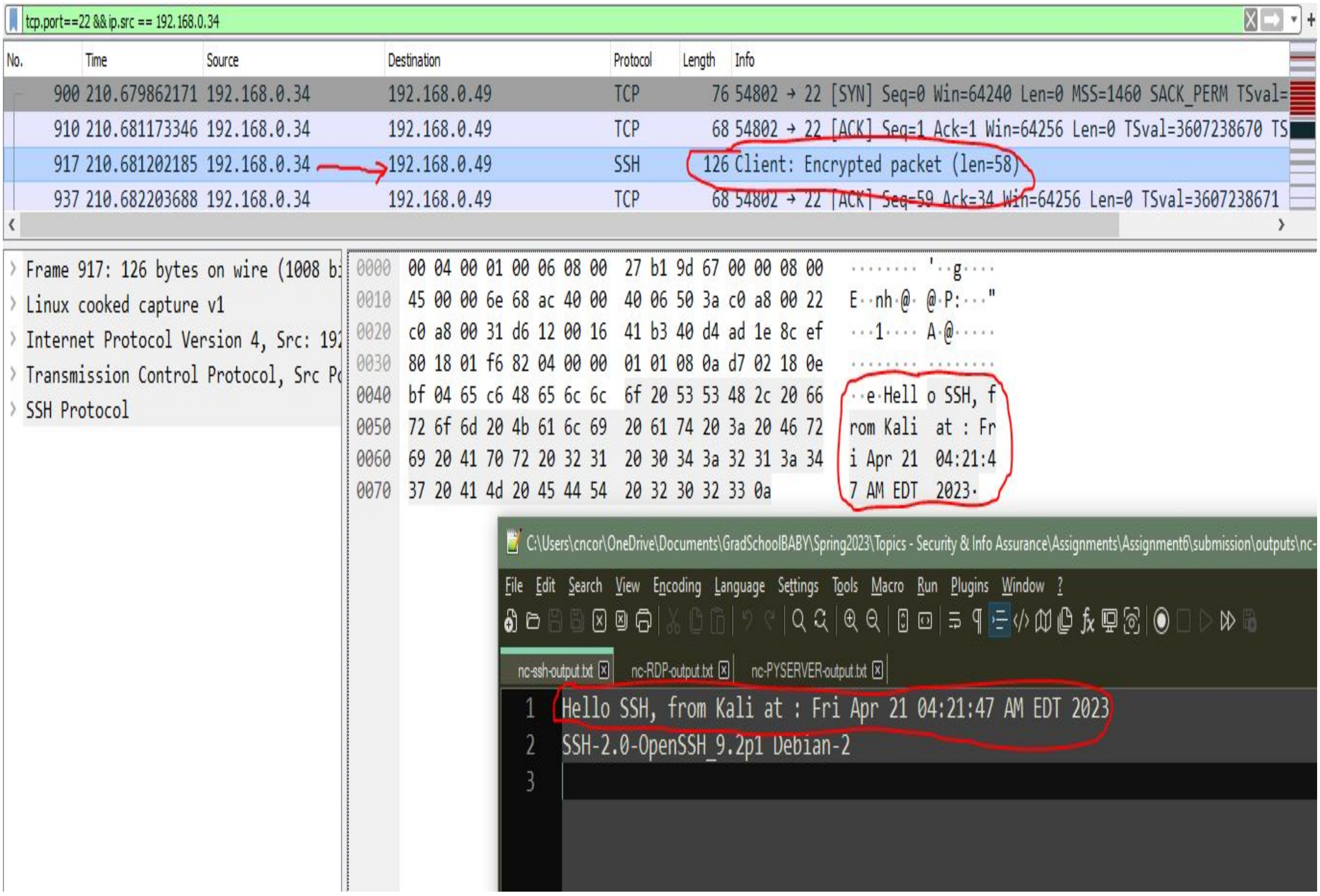
Packet Payloads

The importance of the packet analysis, and the greator honeypot, is in the discovery of the packet payloads. This means that we are able to view the contents of packets, which can then be inferred to the kinds of services the contribute to and from there determine an attackers motives.

For example, packet analysis on an attack on ssh could determine, among otherthings:

- attackers IP which would help determine to the geolocation of the attacker
- data exchanged during ssh connections could show executed commands

As we can see, packet info sends some commands in clear text, despite the attempted encryption. The screenshot shows this confusion; an encrypted packet with a plaintext message. The reason for this is because this is not a real ssh session, therefore there was never any agreement on encryption and keys never exchanged.

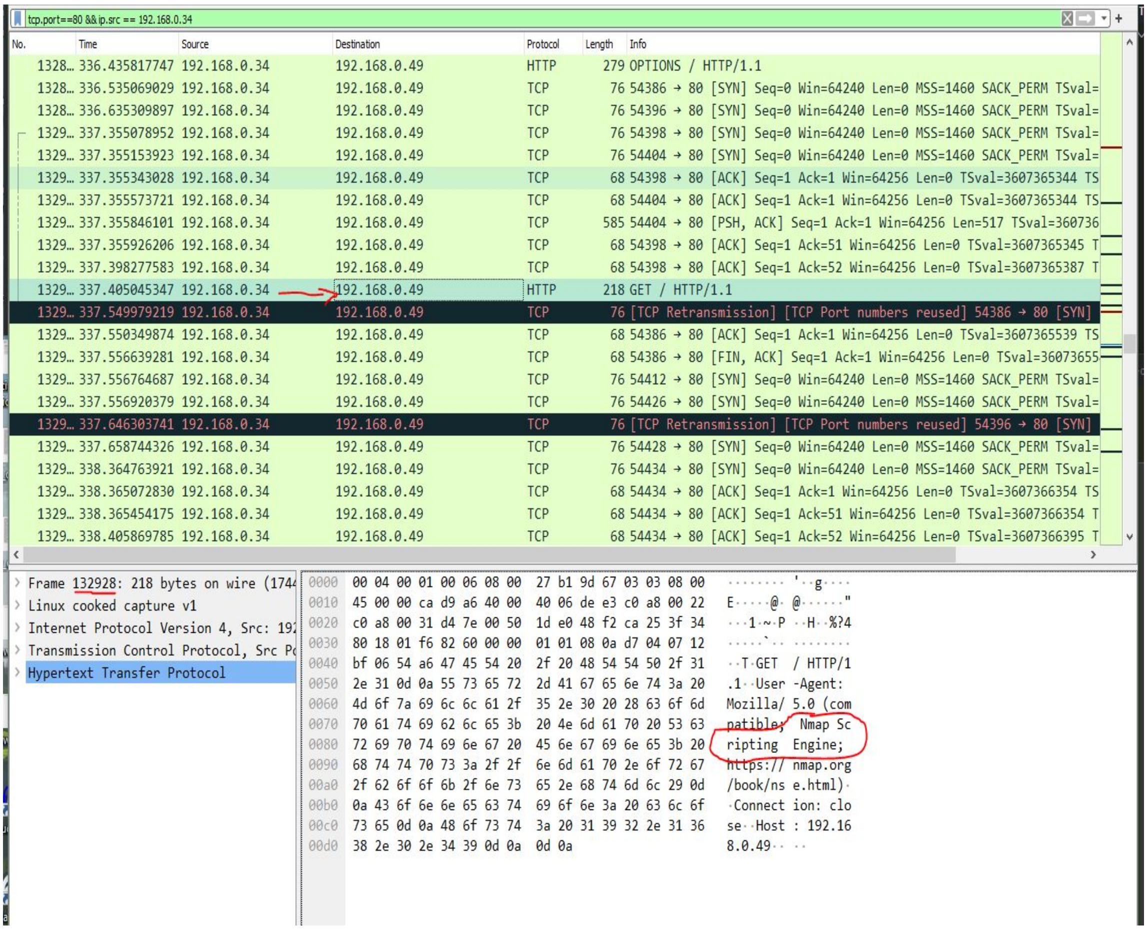


```
tcp.port==22 && ip.src == 192.168.0.34

No.  Time            Source        Destination   Protocol Length Info
 900 210.679862171  192.168.0.34  192.168.0.49  TCP      76 54802 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
 910 210.681173346  192.168.0.34  192.168.0.49  TCP      68 54802 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3607238670 TS
 917 210.681202185  192.168.0.34  192.168.0.49  SSH     126 Client: Encrypted packet (len=58)
 937 210.682203688  192.168.0.34  192.168.0.49  TCP      68 54802 → 22 [ACK] Seq=59 Ack=34 Win=64256 Len=0 TSval=3607238671

> Frame 917: 126 bytes on wire (1008 bi
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 192
> Transmission Control Protocol, Src Po
> SSH Protocol

0000  00 04 00 01 00 06 08 00  27 b1 9d 67 00 00 08 00   ········ '··g····
0010  45 00 00 6e 68 ac 40 00  40 06 50 3a c0 a8 00 22   E··nh·@· @·P:···"
0020  c0 a8 00 31 d6 12 00 16  41 b3 40 d4 ad 1e 8c ef   ···1···· A·@·····
0030  80 18 01 f6 82 04 00 00  01 01 08 0a d7 02 18 0e   ········ ········
0040  bf 04 65 c6 48 65 6c 6c  6f 20 53 53 48 2c 20 66   ··e·Hell o SSH, f
0050  72 6f 6d 20 4b 61 6c 69  20 61 74 20 3a 20 46 72   rom Kali  at : Fr
0060  69 20 41 70 72 20 32 31  20 30 34 3a 32 31 3a 34   i Apr 21  04:21:4
0070  37 20 41 4d 20 45 44 54  20 32 30 32 33 0a         7 AM EDT  2023·
```

```
C:\Users\cncor\OneDrive\Documents\GradSchoolBABY\Spring2023\Topics - Security & Info Assurance\Assignments\Assignment6\submission\outputs\nc-

Hello SSH, from Kali at : Fri Apr 21 04:21:47 AM EDT 2023
SSH-2.0-OpenSSH_9.2p1 Debian-2
```

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
nc-ssh-output.txt
nc-RDP-output.txt
nc-PYSERVER-output.txt

Nmap output shows very good detail about the potential payloads these connections may have been servicing. In this case, nmap was used by an attacker to discover these vulnerabilities, yet various packets captured by wireshark show the nmap characterization.



```
tcp.port==80 && ip.src == 192.168.0.34
```

```
No.     Time          Source          Destination     Protocol  Length Info
1328… 336.435817747 192.168.0.34    192.168.0.49    HTTP      279 OPTIONS / HTTP/1.1
1328… 336.535069029 192.168.0.34    192.168.0.49    TCP        76 54386 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
1328… 336.635309897 192.168.0.34    192.168.0.49    TCP        76 54396 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
1329… 337.355078952 192.168.0.34    192.168.0.49    TCP        76 54398 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
1329… 337.355153923 192.168.0.34    192.168.0.49    TCP        76 54404 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
1329… 337.355343028 192.168.0.34    192.168.0.49    TCP        68 54398 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3607365344 TS
1329… 337.355573721 192.168.0.34    192.168.0.49    TCP        68 54404 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3607365344 TS
1329… 337.355846101 192.168.0.34    192.168.0.49    TCP       585 54404 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=517 TSval=360736
1329… 337.355926206 192.168.0.34    192.168.0.49    TCP        68 54398 → 80 [ACK] Seq=1 Ack=51 Win=64256 Len=0 TSval=3607365345 T
1329… 337.398277583 192.168.0.34    192.168.0.49    TCP        68 54398 → 80 [ACK] Seq=1 Ack=52 Win=64256 Len=0 TSval=3607365387 T
1329… 337.405045347 192.168.0.34    192.168.0.49    HTTP      218 GET / HTTP/1.1
1329… 337.549979219 192.168.0.34    192.168.0.49    TCP        76 [TCP Retransmission] [TCP Port numbers reused] 54386 → 80 [SYN]
1329… 337.550349874 192.168.0.34    192.168.0.49    TCP        68 54386 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3607365539 TS
1329… 337.556639281 192.168.0.34    192.168.0.49    TCP        68 54386 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=36073655
1329… 337.556764687 192.168.0.34    192.168.0.49    TCP        76 54412 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
1329… 337.556920379 192.168.0.34    192.168.0.49    TCP        76 54426 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
1329… 337.646303741 192.168.0.34    192.168.0.49    TCP        76 [TCP Retransmission] [TCP Port numbers reused] 54396 → 80 [SYN]
1329… 337.658744326 192.168.0.34    192.168.0.49    TCP        76 54428 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
1329… 338.364763921 192.168.0.34    192.168.0.49    TCP        76 54434 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
1329… 338.365072830 192.168.0.34    192.168.0.49    TCP        68 54434 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3607366354 TS
1329… 338.365454175 192.168.0.34    192.168.0.49    TCP        68 54434 → 80 [ACK] Seq=1 Ack=51 Win=64256 Len=0 TSval=3607366354 T
1329… 338.405869785 192.168.0.34    192.168.0.49    TCP        68 54434 → 80 [ACK] Seq=1 Ack=52 Win=64256 Len=0 TSval=3607366395 T
```

```
> Frame 132928: 218 bytes on wire (1744
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 19
> Transmission Control Protocol, Src P
> Hypertext Transfer Protocol
```

```
0000  00 04 00 01 00 06 08 00  27 b1 9d 67 03 03 08 00   ········ '··g····
0010  45 00 00 ca d9 a6 40 00  40 06 de e3 c0 a8 00 22   E·····@· @······"
0020  c0 a8 00 31 d4 7e 00 50  1d e0 48 f2 ca 25 3f 34   ···1·~·P ··H··%?4
0030  80 18 01 f6 82 60 00 00  01 01 08 0a d7 04 07 12   ·····`·· ········
0040  bf 06 54 a6 47 45 54 20  2f 20 48 54 54 50 2f 31   ··T·GET  / HTTP/1
0050  2e 31 0d 0a 55 73 65 72  2d 41 67 65 6e 74 3a 20   .1··User -Agent: 
0060  4d 6f 7a 69 6c 6c 61 2f  35 2e 30 20 28 63 6f 6d   Mozilla/ 5.0 (com
0070  70 61 74 69 62 6c 65 3b  20 4e 6d 61 70 20 53 63   patible;  Nmap Sc
0080  72 69 70 74 69 6e 67 20  45 6e 67 69 6e 65 3b 20   ripting  Engine; 
0090  68 74 74 70 73 3a 2f 2f  6e 6d 61 70 2e 6f 72 67   https:// nmap.org
00a0  2f 62 6f 6f 6b 2f 6e 73  65 2e 68 74 6d 6c 29 0d   /book/ns e.html)·
00b0  0a 43 6f 6e 6e 65 63 74  69 6f 6e 3a 20 63 6c 6f   ·Connect ion: clo
00c0  73 65 0d 0a 48 6f 73 74  3a 20 31 39 32 2e 31 36   se··Host : 192.16
00d0  38 2e 30 2e 34 39 0d 0a  0d 0a                     8.0.49·· ··
```

We can see nmap requesting a GET http method. This would suggest that the attacker is attempting to scan for open ports that could potentially serve as an exploitation.

Conclusion

In this assignemnt, we used various netcat listeners to simulate a honeypot using an Ubuntu system A kali linux system was used to provide various forms of communications with theses listeners. Wireshark packet capture was used to intercept the attacking traffic. Packet analysis showed various information about the attacker including ip address, plaintext commands, and packet payloads.

The purpose of this honeypot was to discover the various techniques and tatctics used by an attacker. This was a demonstration of the importance of gathering defensive intelligence. Intel gathred by honeypots can strength defensive capabilities. Understadning how attackers exploit networks and systems allows for defenders to implement proper techniques to prevent them.

Honeypots offer a sense of relief that the main system is safe from attacks while the honeypot serves as a "trap". However, "trap" implies that an attack was caught and somehow diverted. Instead, honeypots offer the ability for defenders to learn about the various tactics and techniques used by attackers without risk of exposing the main system to attacks. The greatest purpose of a honeypot lies is in gathering intelligence about potential attacks and vulnerabilities that a network might incur without risk of exposing the network to real attacks.

It is tempting to call honeypots "security through obscurity". In fact, a honeypot is not an active defensive tool, yet more of a tool for defense. This means that the intelligence learned from the attacks the honeypot "traps" will embolden the true defensive tools. In that way, defensive tools are boradened to protect more vulnerabilties with the intel gathered by honeypots.

Before describing how each tool used in this assignment provides or does not provide the X.800 Secuirty Services, lets 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 resouces through secuirty policies and mechanisms.
3. *Data Confideniality*: prevents unauthorized data acess.
4. *Data-Integrity*: provides assurance that total data streams remain unchanged by unauthorized entities.
5. *Non-repuditation*: protects against denial of involvement within a connection.

HONEYPOT : Authentication, Access Control, Data Confidentialty, Data-integrity, Non-repudiation

Honeypots offer intel on the techniques used by attackers to exploit systems. In this way, honeypots effect each of the X.800 Security Services. Generally, honeypots offer intel on how to identify exploits within these services.

Authentication: A honeypot might be used as an authentication server as seen apart of the EAPOL scheme. An attacker is able to bypass authentication mechnisms. Yet a honey would help determine how this was done.

Access Control: An attacker might try to issue priviledged commands on a fake ssh server setupt by a honeypot. This would be trapped by the honeypot and an analysis of the attack could provide important insight on the techniques used.

Data Confidentialty: Similarly, an attacker might attempt to access data that they are not allowed to see. Honeypots would in the discovery of the various mechanisms that an attacker would use to break this confidentiality.

Data-integrity: Attackers might attempt to modify data stored on a honeypot. Analysis of this attack provided by the honeypot could determine how these attempts might be made.

Non-repudiation: Honeypots log all various outputs from connections made by that attacker. This provides non-repudiation against any denial of responsibility by the attacker.