Assignment 1

Issiah Deleon

Abstract

The intent of this lab is to demonstrate the vulnerability of data sent over a network. That data is free to be intercepted by anyone “listening” to the network traffic using software such as WireShark. Sample data is generated to demonstrate the interception of network traffic using two terminals in one virtual machine. The results demonstrate that all data is vulnerable to unintended interceptions by a third party, particularly unencrypted traffic.

Introduction

Kali Linux is used inside a virtual machine with VirtualBox Manager software. Within the Linux machine, Netcat is used as a computer networking utility for sending and receiving packets of unencrypted data over a network. To “listen” in on the exchange of data, WireShark will be used as a packet analyzer.

The following Netcat commands are used to send (sender) information from one terminal, and to receive the information from another one (listener).

Netcat listener commands:

```
nc -l -p 3000
```

Netcat sender commands:

```
nc 10.0.0.231 3000
```

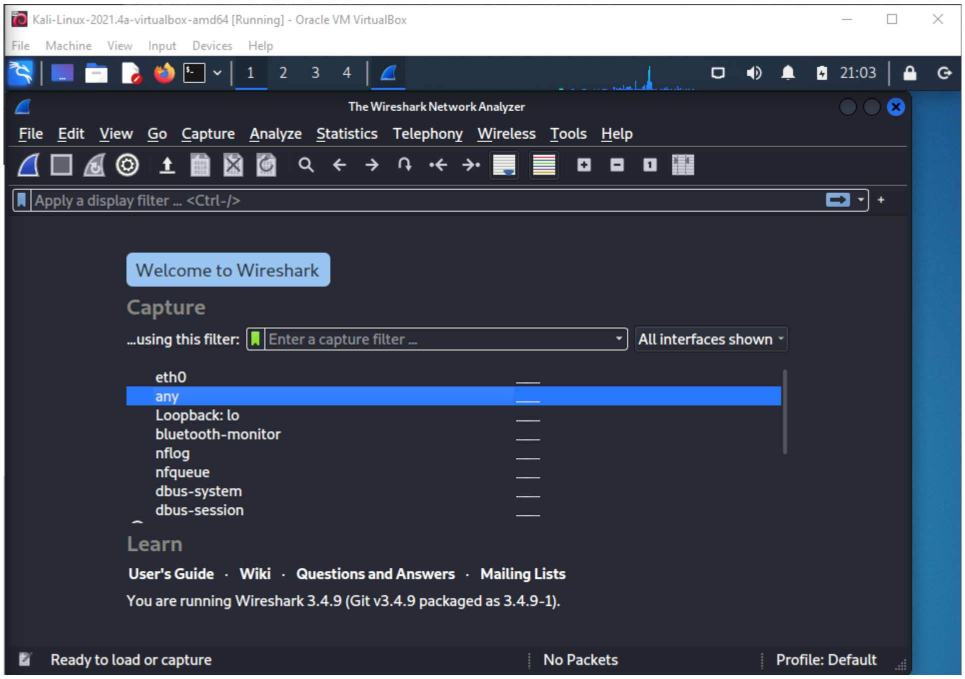
IP Address of system: 10.0.0.231

Port # used: 3000

WireShark will run as the root user and will use the "any" adapter to listen to all devices and their traffic.

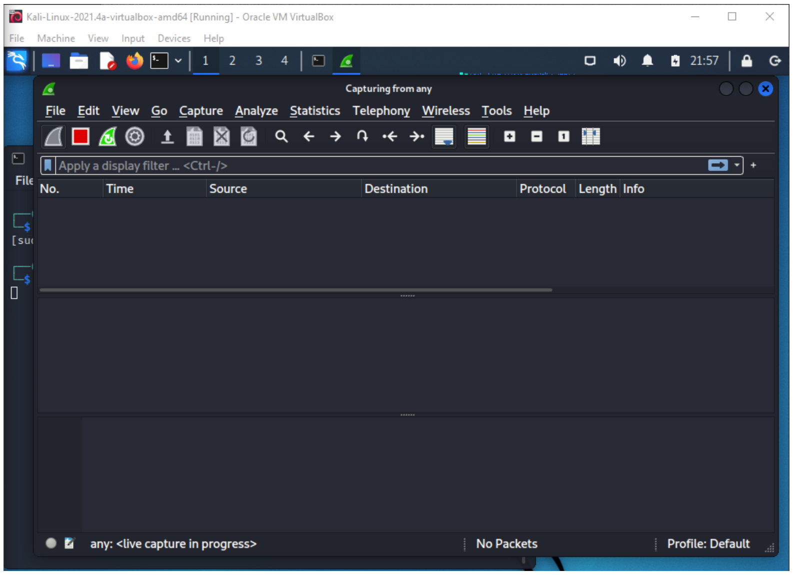
Summary

At the top left of your virtual desktop, click on the "Applications" icon. Search for WireShark and open the application. After opening WireShark, select the "any" option highlighted in the screenshot below. This allows you to capture the traffic of every device.


Kali-Linux-2021.4a-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
21:03
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Welcome to Wireshark
Capture
…using this filter:
Enter a capture filter …
All interfaces shown
eth0
any
Loopback: lo
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists

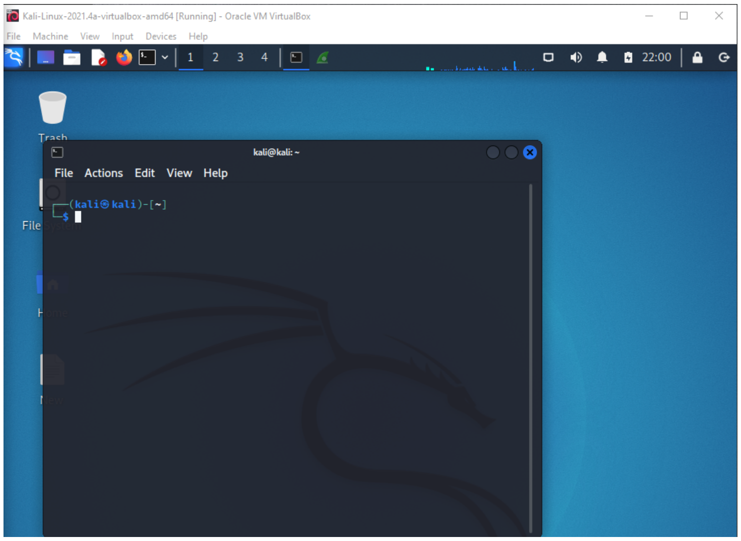
You are running Wireshark 3.4.9 (Git v3.4.9 packaged as 3.4.9-1).

Ready to load or capture No Packets Profile: Default


Kali-Linux-2021.4a-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
21:57
Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
No. Time Source Destination Protocol Length Info

any: <live capture in progress> No Packets Profile: Default

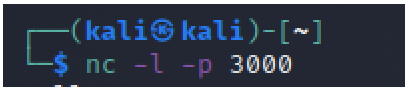
Listening will begin automatically and look like this before collecting any packets.


Kali-Linux-2021.4a-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
22:00
Trash
File
kali@kali: ~
File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ █
```

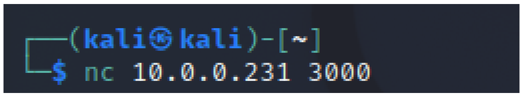
Now, open your first terminal window located at the top left of the screen. It looks like a black box with a $ symbol inside. This will be your "listener" terminal. Once opened, type in the following "listener" command: *nc -l -p 3000*. This instructs the terminal to connect to port 3000.

Your terminal should look like this, and is ready to receive text from the other terminal.



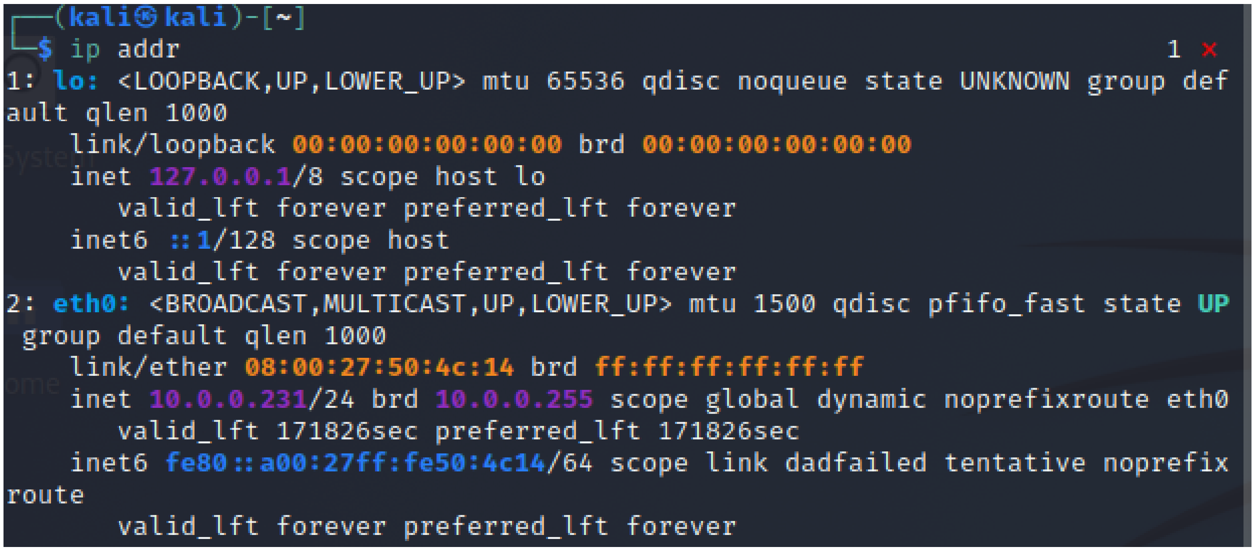
```
┌──(kali㉿kali)-[~]
└─$ nc -l -p 3000
```

Next, open another terminal to serve as your "sender" terminal by simply clicking on the terminal icon once more. You will see a new blank terminal window, into which you should type the following command to connect to the same port your listener terminal is connected to: *nc 10.0.0.231 3000*. You will notice that after entering this command, the command line moves to a new blank line and allows text to be typed. Go ahead and type anything onto a new line in your sender terminal, such as the word "Testing" or "Hello" and press enter.

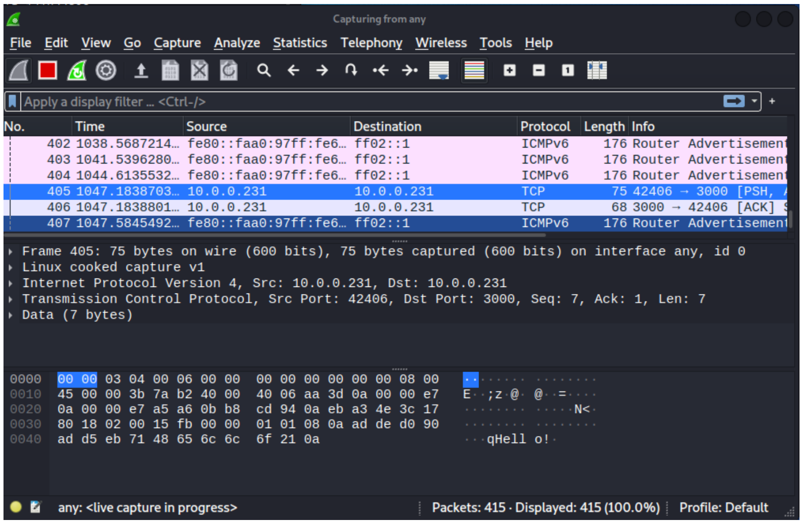


```
┌──(kali㉿kali)-[~]
└─$ nc 10.0.0.231 3000
```

The IP address used above was gathered using the command: ip addr. Shown below, we will be using the address under eth0: inet - 10.0.0.231



```
┌──(kali㉿kali)-[~]
└─$ ip addr                                                                1 ✕
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def
ault 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: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
 group default qlen 1000
    link/ether 08:00:27:50:4c:14 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.231/24 brd 10.0.0.255 scope global dynamic noprefixroute eth0
       valid_lft 171826sec preferred_lft 171826sec
    inet6 fe80::a00:27ff:fe50:4c14/64 scope link dadfailed tentative noprefix
route
       valid_lft forever preferred_lft forever
```



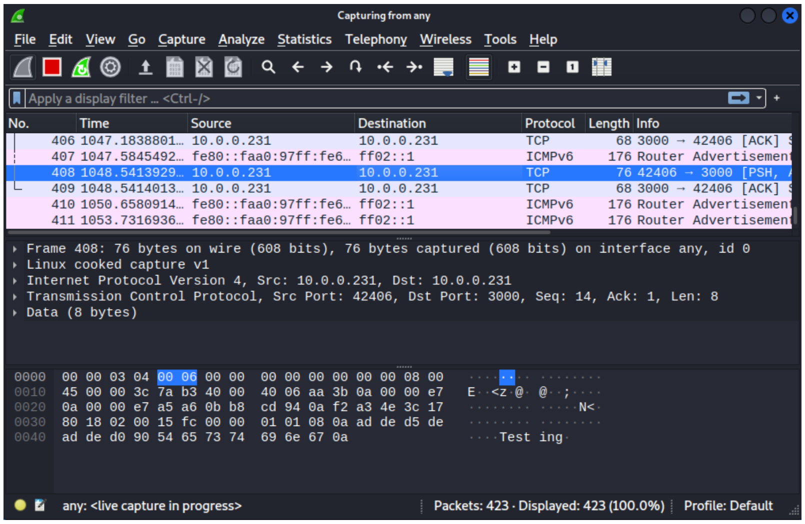
```
No.   Time            Source                 Destination     Protocol  Length Info
  402 1038.5687214…  fe80::faa0:97ff:fe6…  ff02::1         ICMPv6      176 Router Advertisement
  403 1041.5396280…  fe80::faa0:97ff:fe6…  ff02::1         ICMPv6      176 Router Advertisement
  404 1044.6135532…  fe80::faa0:97ff:fe6…  ff02::1         ICMPv6      176 Router Advertisement
  405 1047.1838703…  10.0.0.231            10.0.0.231      TCP          75 42406 → 3000 [PSH, A
  406 1047.1838801…  10.0.0.231            10.0.0.231      TCP          68 3000 → 42406 [ACK] S
  407 1047.5845492…  fe80::faa0:97ff:fe6…  ff02::1         ICMPv6      176 Router Advertisement
```

```
▸ Frame 405: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface any, id 0
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 10.0.0.231, Dst: 10.0.0.231
▸ Transmission Control Protocol, Src Port: 42406, Dst Port: 3000, Seq: 7, Ack: 1, Len: 7
▸ Data (7 bytes)
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 00 3b 7a b2 40 00  40 06 aa 3d 0a 00 00 e7   E··;z·@· @··=····
0020  0a 00 00 e7 a5 a6 0b b8  cd 94 0a eb a3 4e 3c 17   ········ ·····N<·
0030  80 18 02 00 15 fb 00 00  01 01 08 0a ad de d0 90   ········ ········
0040  ad d5 eb 71 48 65 6c 6c  6f 21 0a                  ···qHell o!·
```

any: <live capture in progress> | Packets: 415 · Displayed: 415 (100.0%) | Profile: Default

Now open WireShark and locate the packet containing “TCP” under the “Protocol” column, and “[PSH, ACK]” in the “Info” column of the packets. This packet contains the text sent from the sender terminal and received from the listener terminal using Netcat.



Capturing from any

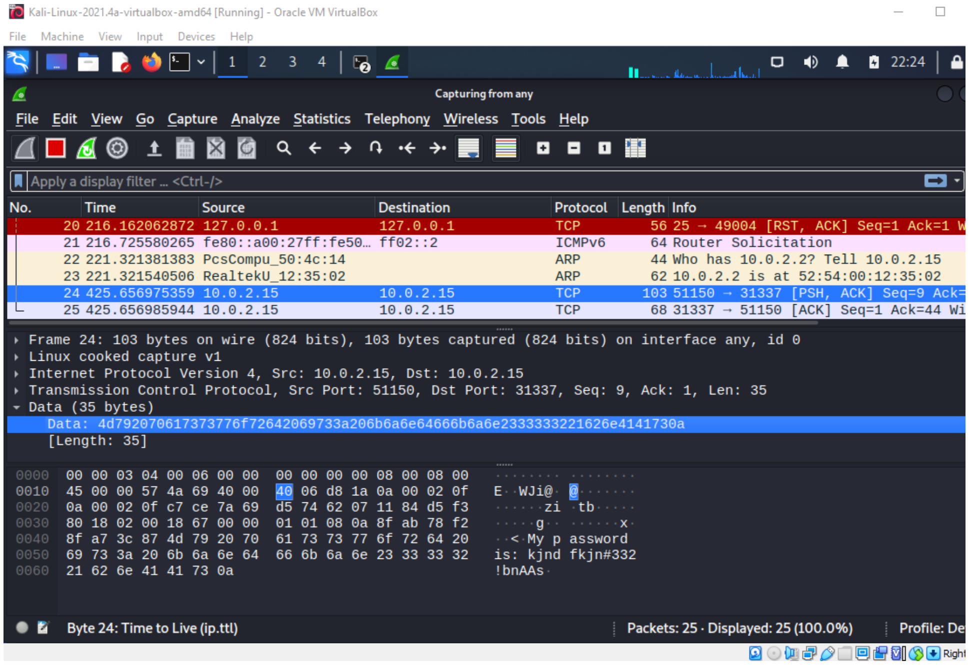
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time            Source                 Destination   Protocol  Length Info
 406 1047.1838801…  10.0.0.231             10.0.0.231    TCP           68 3000 → 42406 [ACK]
 407 1047.5845492…  fe80::faa0:97ff:fe6…   ff02::1       ICMPv6       176 Router Advertisement
 408 1048.5413929…  10.0.0.231             10.0.0.231    TCP           76 42406 → 3000 [PSH,
 409 1048.5414013…  10.0.0.231             10.0.0.231    TCP           68 3000 → 42406 [ACK]
 410 1050.6580914…  fe80::faa0:97ff:fe6…   ff02::1       ICMPv6       176 Router Advertisement
 411 1053.7316936…  fe80::faa0:97ff:fe6…   ff02::1       ICMPv6       176 Router Advertisement
```

```
› Frame 408: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 10.0.0.231, Dst: 10.0.0.231
› Transmission Control Protocol, Src Port: 42406, Dst Port: 3000, Seq: 14, Ack: 1, Len: 8
› Data (8 bytes)
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 00 3c 7a b3 40 00  40 06 aa 3b 0a 00 00 e7   E··<z·@· @··;····
0020  0a 00 00 e7 a5 a6 0b b8  cd 94 0a f2 a3 4e 3c 17   ········ ·····N<·
0030  80 18 02 00 15 fc 00 00  01 01 08 0a ad de d5 de   ········ ········
0040  ad de d0 90 54 65 73 74  69 6e 67 0a               ····Test ing·
```

any: <live capture in progress> | Packets: 423 · Displayed: 423 (100.0%) | Profile: Default



Kali-Linux-2021.4a-virtualbox-amd64 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help
1 2 3 4
22:24
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Right

Capturing from any

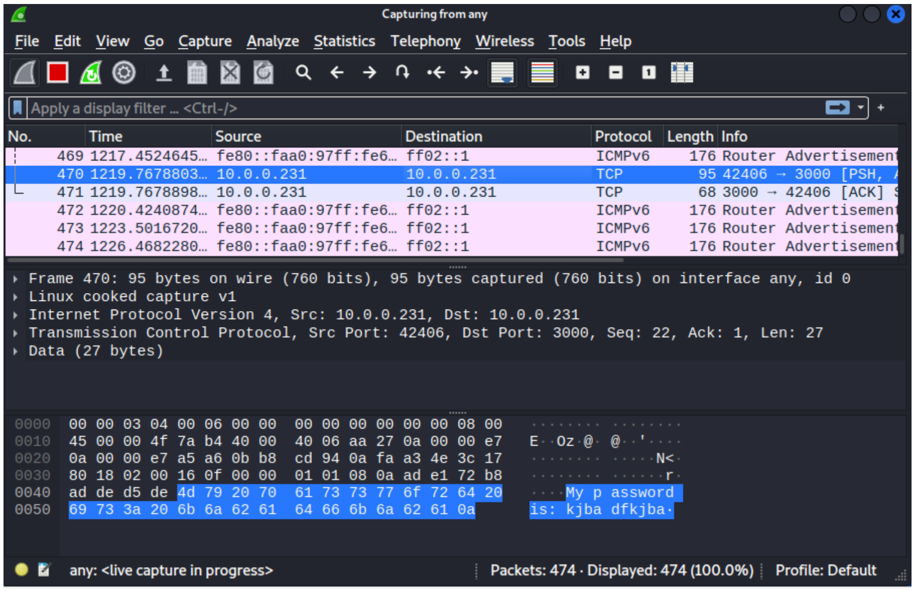
```
No.  Time           Source                 Destination  Protocol  Length Info
  20 216.162062872  127.0.0.1              127.0.0.1    TCP           56 25 → 49004 [RST, ACK] Seq=1 Ack=1 W
  21 216.725580265  fe80::a00:27ff:fe50…   ff02::2      ICMPv6        64 Router Solicitation
  22 221.321381383  PcsCompu_50:4c:14                   ARP           44 Who has 10.0.2.2? Tell 10.0.2.15
  23 221.321540506  RealtekU_12:35:02                   ARP           62 10.0.2.2 is at 52:54:00:12:35:02
  24 425.656975359  10.0.2.15              10.0.2.15    TCP          103 51150 → 31337 [PSH, ACK] Seq=9 Ack=
  25 425.656985944  10.0.2.15              10.0.2.15    TCP           68 31337 → 51150 [ACK] Seq=1 Ack=44 Wi
```

```
› Frame 24: 103 bytes on wire (824 bits), 103 bytes captured (824 bits) on interface any, id 0
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.15
› Transmission Control Protocol, Src Port: 51150, Dst Port: 31337, Seq: 9, Ack: 1, Len: 35
⌄ Data (35 bytes)
    Data: 4d792070617373776f72642069733a206b6a6e64666b6a6e2333333221626e4141730a
    [Length: 35]
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 08 00 08 00   ········ ········
0010  45 00 00 57 4a 69 40 00  40 06 d8 1a 0a 00 02 0f   E··WJi@· @·······
0020  0a 00 02 0f c7 ce 7a 69  d5 74 62 07 11 84 d5 f3   ······zi ·tb·····
0030  80 18 02 00 18 67 00 00  01 01 08 0a 8f ab 78 f2   ·····g·· ······x·
0040  8f a7 3c 87 4d 79 20 70  61 73 73 77 6f 72 64 20   ··<·My p assword
0050  69 73 3a 20 6b 6a 6e 64  66 6b 6a 6e 23 33 33 32   is: kjnd fkjn#332
0060  21 62 6e 41 41 73 0a                               !bnAAs·
```

Byte 24: Time to Live (ip.ttl) | Packets: 25 · Displayed: 25 (100.0%) | Profile: De

Notice how the information sent from the sender terminal is intercepted and gathered by WireShark for anyone listening to view. Both the "Testing" text and the "My password is:" texts were intercepted and the packets were clear to view and analyze. If the information sent had been an actual password or other piece of private data, the eavesdropper would have full access to it.


Capturing from any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.  Time             Source                Destination   Protocol Length Info
 469 1217.4524645…   fe80::faa0:97ff:fe6…  ff02::1       ICMPv6     176 Router Advertisement
 470 1219.7678803…   10.0.0.231            10.0.0.231    TCP         95 42406 → 3000 [PSH, A
 471 1219.7678898…   10.0.0.231            10.0.0.231    TCP         68 3000 → 42406 [ACK] S
 472 1220.4240874…   fe80::faa0:97ff:fe6…  ff02::1       ICMPv6     176 Router Advertisement
 473 1223.5016720…   fe80::faa0:97ff:fe6…  ff02::1       ICMPv6     176 Router Advertisement
 474 1226.4682280…   fe80::faa0:97ff:fe6…  ff02::1       ICMPv6     176 Router Advertisement

Frame 470: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 10.0.0.231, Dst: 10.0.0.231
Transmission Control Protocol, Src Port: 42406, Dst Port: 3000, Seq: 22, Ack: 1, Len: 27
Data (27 bytes)

0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 00 4f 7a b4 40 00  40 06 aa 27 0a 00 00 e7   E··Oz·@· @··'····
0020  0a 00 00 e7 a5 a6 0b b8  cd 94 0a fa a3 4e 3c 17   ········ ·····N<·
0030  80 18 02 00 16 0f 00 00  01 01 08 0a ad e1 72 b8   ········ ······r·
0040  ad de d5 de 4d 79 20 70  61 73 73 77 6f 72 64 20   ····My p assword 
0050  69 73 3a 20 6b 6a 62 61  64 66 6b 6a 62 61 0a      is: kjba dfkjba·

any: <live capture in progress>   Packets: 474 · Displayed: 474 (100.0%)   Profile: Default
```

Conclusion

This assignment has demonstrated how data or traffic can be intercepted by packet analyzing software such as WireShark. Although the information sent from a sender terminal and received by the listener terminal was not encrypted, this is still a solid demonstration of the inherent vulnerability of any type of data sent over a network. Further measures are necessary to secure private and personal information over a network.

WireShark is a piece of software that essentially opens a window between two entities - the sender and the receiver. Through this window, WireShark allows the user to intercept the network traffic between these two entities. This provides the user with access to network traffic that would otherwise be impossible to analyze. This is a useful tool to help diagnose any potential problems within a network, such as security vulnerabilities or active security threats. It allows you to hone in on the root cause of these problems, providing insight and valuable information to the user. Using this tool allowed me to discover the inherent vulnerabilities of any and all data being sent across a network, from one entity to another. I discovered that with these inherent vulnerabilities comes various measures to secure your information with methods such as data encryption, which essentially scrambles the information contained within packets so that anyone snooping in on a piece of software like WireShark is unable to do anything with the information collected.