COMP3052.SEC LAB: PACKET SNIFFING AND ANALYSIS

(**Thank you to Mike Pound and Liming Xu**)

LAB DESCRIPTION

In this lab we'll be looking at packet sniffing and analysis techniques. As with the last lab (Firewalls and Port Scanning), we'll use communication between two virtual machines as a demonstration of various Internet protocols, and what information can be gleaned by passively observing them.

INTRODUCTION

At a very low level, everything we send over the internet is split into small datagrams called packets. Data, which may be any part of any protocol currently being used, can be wrapped up in TCP frames, which includes ordering and session information for sustained communication between parties. TCP packets also send acknowledgements to confirm receipt. Above this, an IP frame holds the addresses necessary for transmission across the network, but contains little by way of delivery confirmation or session information.

What might surprise some about network communication is that packets are often seen by many computers, not just the recipient. Routers and switches will make an effort to forward packets in the right direction, but the final hop will usually just broadcast packets to all nearby machines assuming they'll ignore the message if it isn't for them.

This makes packet sniffing possible: we can record every packet that appears, not just those that are addressed to us. Packet analysis takes this a step further and uses knowledge of protocols to examine the incoming packets for patterns. This is fundamental to security analysis, network protocol testing, and of course, man-in-the middle attacks.

CLONING VIRTUAL MACHINES

As with the previous lab session (Firewalls and Port Scanning), we need two virtual machines to examine the communication protocols. Please refer to the separate "Cloning VMs" and "Networking VMs" lab manuals, where you will find instructions on setting up a network of machines. Return here once you have two Kali installations running, and communicating. These machines will be set up to communicate only with each other. **For obvious reasons, sniffing wider network (e.g., an entire lab with full of people) packets isn't allowed on the School machines!**

Login Information

As with previous labs, a description of the Kali operating system and the general setup of the virtual machines is given in the first lab document and the additional files. Both machines have the same login credentials, which are listed here for reference. As usual, you should know to avoid logging in as root!

Normal User	**Root**
Username: sec	Username: root
Password: security	Password: toor

SETTING UP THE SERVER

Select the running VM that will act as your server. We won't worry about a firewall this time, but we want to make sure SSH and Telnet are running. As with the last lab, these commands should set you up:

SSH: This is already installed, simply use the command: `sudo service ssh start`

Telnet: (if already installed): `sudo service xinetd start`

If not installed, then: `sudo apt-get install xinetd`

You can perform `nmap localhost` on yourself to confirm both services are listening on their respective ports. Once these are installed and running, you can minimize this machine and start work on the testing VM.

PACKET CAPTURE

Before we can examine the contents of packets and see what's going on in a network, we need to capture them. There are numerous tools we can use to do this, they switch the network adapter into promiscuous mode, and grab everything they see. Let's look at a simple terminal example, `tcpdump`.

Open up two terminals on the client machine. On the first, initiate a tcpdump session using the following command:

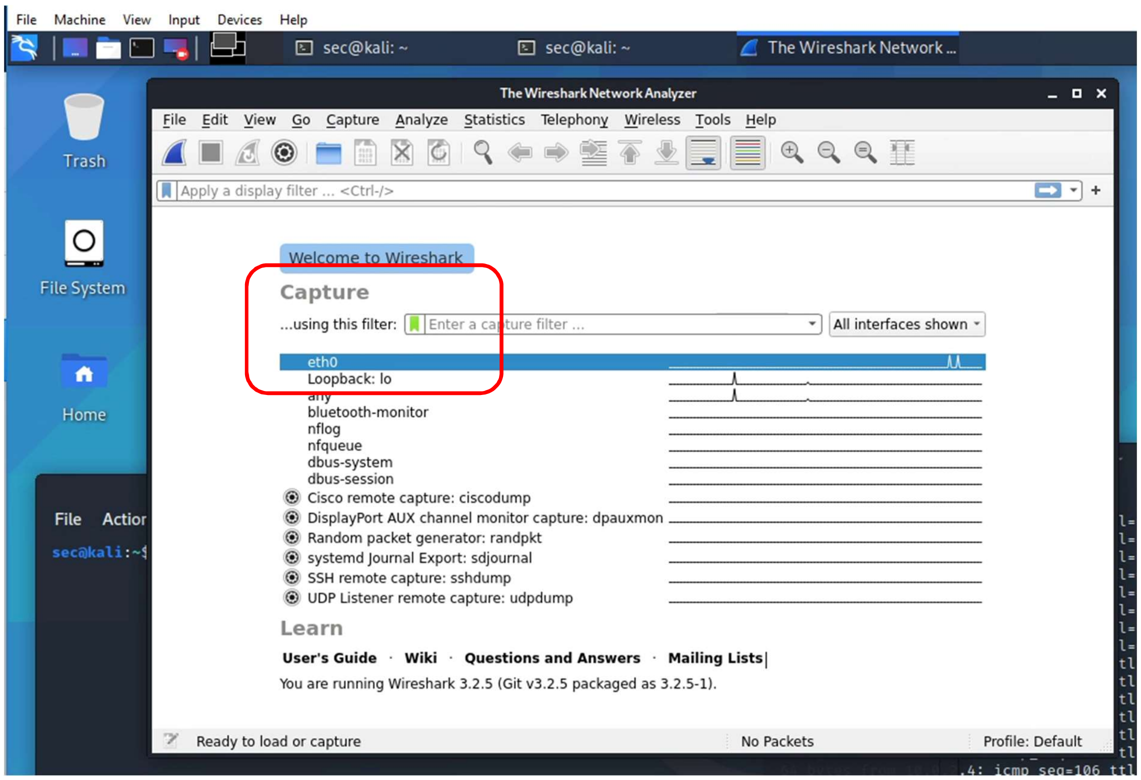
```
sec@kali:~$ sudo tcpdump
```

In the other window, send a few pings to the server, then cancel both using Ctrl+C. What packets you'll see will depend a little on what's happening on your machine at that moment, but a private connection between two VMs should be fairly quiet. You should mostly see packets relevant to the ping requests. If the network hasn't been up long, your machine will need to perform an Address Resolution Protocol (ARP) request to determine the hardware address of the server. Once this is back, it sends off an ICMP packet requesting a reply, and with any luck it will get one. Notice that each subsequent ping exchange increments the seq parameter.

Packet Analysis

Packet analysis takes the output of packet capture and pieces together what protocols are in use, and what events have taken place. For example, we'll see that an SSH session has a very specific handshake phase, with a key exchange, before the commencement of a secure session. It's possible to direct the output of tcpdump or other capture tools into analysis software, but Wireshark lets us capture and analyse within the same UI. This, and the number of features the software has, makes Wireshark one of the most popular applications with security experts, admins and hackers alike.

You can run Wireshark from the application menu on the top left of the screen, or from the terminal. You'll need to enter a password to provide the privileges needed to access the network card, then you'll see the user interface:


File Machine View Input Devices Help
sec@kali: ~
sec@kali: ~
The Wireshark Network ...
Trash
File System
Home
File Actior
sec@kali:~$
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
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists

```
eth0
Loopback: lo
any
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
```

You are running Wireshark 3.2.5 (Git v3.2.5 packaged as 3.2.5-1).

Ready to load or capture   No Packets   Profile: Default

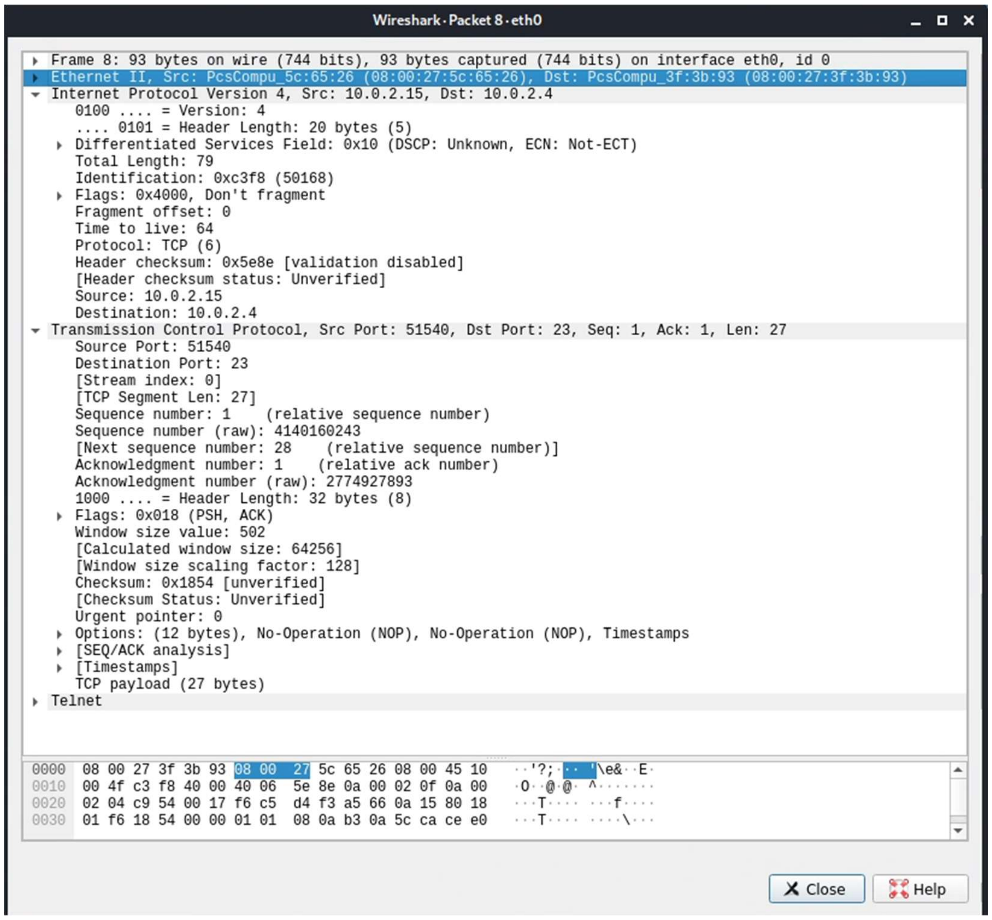
We can begin capturing packets from the red highlighted area on the left. Double click on eth0, or click Capture and select it. The interface will change to a packet list, similar to tcpdump. Head to a terminal window and begin a Telnet session.

```
sec@kali:~$ telnet [ip-of-server-vm]
```

Type in your username and password, then once the authentication completes you can type exit to leave. Head back to Wireshark and stop capturing packets using the red square button on the toolbar. You'll see a long list of packets, starting with TCP [SYN] followed by a combination of TELNET and TCP [ACK], with perhaps a few ARP or DHCP packets around, too. The SYN (synchronize) and ACK (acknowledge) packets are the TCP protocol starting communication, before Telnet data begins. Throughout the Telnet we will see more acknowledgements, in response to most TCP packets.

Remember that TCP is just a frame in which Telnet data sits, even the packets marked as Telnet by Wireshark are wrapped in TCP packets, which are themselves inside IP and Ethernet packets. We can make our analysis a little easier by removing all except for the Telnet-relevant packets. Simply add the word "telnet" to the filter bar above the packets, then press return.

You can analyse the inner details of a single packet by double-clicking on it. If you do, you will see something like this:



```
Frame 8: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_5c:65:26 (08:00:27:5c:65:26), Dst: PcsCompu_3f:3b:93 (08:00:27:3f:3b:93)
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.4
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 79
    Identification: 0xc3f8 (50168)
    Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x5e8e [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.15
    Destination: 10.0.2.4
Transmission Control Protocol, Src Port: 51540, Dst Port: 23, Seq: 1, Ack: 1, Len: 27
    Source Port: 51540
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 27]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 4140160243
    [Next sequence number: 28    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 2774927893
    1000 .... = Header Length: 32 bytes (8)
    Flags: 0x018 (PSH, ACK)
    Window size value: 502
    [Calculated window size: 64256]
    [Window size scaling factor: 128]
    Checksum: 0x1854 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    [SEQ/ACK analysis]
    [Timestamps]
    TCP payload (27 bytes)
Telnet

0000  08 00 27 3f 3b 93 08 00  27 5c 65 26 08 00 45 10   ··'?;···'\e&··E·
0010  00 4f c3 f8 40 00 40 06  5e 8e 0a 00 02 0f 0a 00   ·O··@·@·^·······
0020  02 04 c9 54 00 17 f6 c5  d4 f3 a5 66 0a 15 80 18   ···T·····f······
0030  01 f6 18 54 00 00 01 01  08 0a b3 0a 5c ca ce e0   ···T········\···
```

The raw data for the packet is at the bottom. The meaning of each byte of data is shown as a hierarchy above. Notice we have an Ethernet packet, encapsulating IP and TCP packets. Last of all, a small amount of Telnet data right at the bottom. Some of the packets will have as little as a single byte of Telnet information in. Some packets, like HTTP or file transfer, can be much larger.

Now that you have a grasp of looking at a single packet, try to work out what's happening over the entire conversation. Can you work out:

- How many packets are required to perform the handshake and confirmation of session settings, before authentication starts?
- How does Telnet attempt to obfuscate the username and password during authentication?

As you can see, Telnet does a pretty poor job of obscuring your password. Perhaps that's because Telnet, FTP, and others were designed at a time where security wasn't paramount. Or maybe the developers just didn't consider that people might want your username and password. Clearly we aren't in that position now, the Telnet protocol is unsuitable for most uses in modern systems.

A far better alternative is SSH. SSH uses a brief handshake, then a key exchange mechanism to establish a shared symmetric private key. Authentication credentials are sent after encryption has begun, only man-in-the-middle during the handshakes or a programming error somewhere in the protocol could be expected to break the encryption once it's begun. In this server's OpenSSH implementation, your chances as a hacker are not good.

Start a new Wireshark capture, and then initiate and leave an SSH session. Stop capture again, and begin an examination of the packets. You'll see the TCP handshake, and SSH handshake, followed by a key exchange, then simply "Encrypted packet" from then on. See if you can work out what's going on in the key exchange process. Here are some hints:

- Take a look at the key exchange init packets. Notice the lists of acceptable methods for the various parts of the SSH process.
- Key exchange mechanisms are listed in order of preference, so the first one available on both systems will get used (slightly different to TLS where the server chooses based on the client's list). Examine the key exchange packets, but don't worry too much about the exact payload of the packets. Those interested can see an outline of what these packets are here: https://tools.ietf.org/html/rfc5656#page-7
- Once we see the new keys message, we can't read the contents of any future packets. The last encrypted packet will likely contain the SSH2_MSG_ CHANNEL_CLOSE message, after which you will see the TCP connection is also closed by the software.

Conclusion

In this lab session you've learned how to use packet sniffing to determine what is happening on a network. We've just looked at small examples, the kind you might use to diagnose connection or protocol issues. Packet sniffing has a much wider variety of uses, ranging from finding out what websites people are visiting, through to the beginnings of man in the middle attacks.