The University of Newcastle
School of Information and Physical Sciences

SENG2250/6250 System and Network Security
Week 11 Lab

This lab should be conducted under the Windows 10 VM. Access your virtual lab here:

https://cybersec-vra04.newcastle.edu.au/

Username: student

Password: $tud3nt

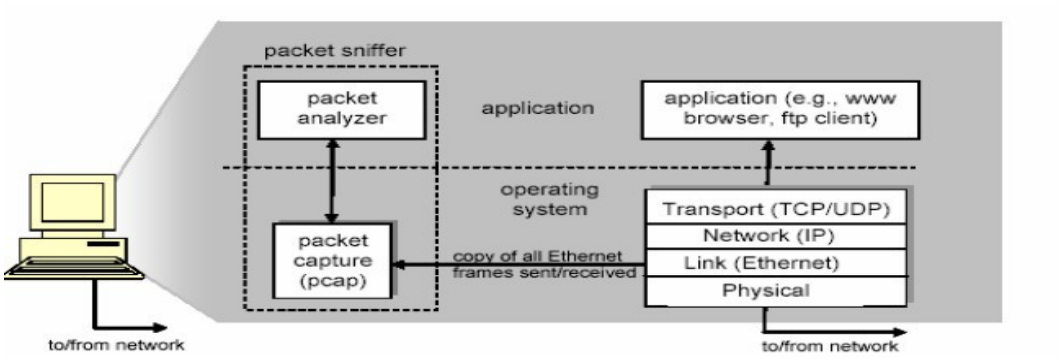
The objectives of this lab are twofold:

1) Introduction to Wireshark.

2) Inspect security differences between Telnet and SSH for remote control.

Part 1: Introduction to Wireshark

Packet sniffers (such as Wireshark) are applications that capture packets that are seen by a machine's network interface. Packet sniffers can be used to troubleshoot networks and also in application and network software development. Moreover, packet sniffers can be used in security attacks, such as eavesdropping.



2.1 Theory

When a sniffer runs on a system, it grabs all the packets that come into and goes out of the Network Interface Card (NIC) of the machine on which the sniffer is installed. This means that, if the NIC is set to the promiscuous mode, then it will receive all the packets sent to the network if that network is connected by a hub (if the network is switched, this won't happen). A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a copy of packets that are sent/received from/by application and protocols executing on your machine. In cases where the adversary wants to sniff packets that are not intended to be sent to it (such as in switched networks), they may perform an ARP poisoning attack. This is an active attack which involves sending a spoofed Address Resolution Protocol message into the

LAN, causing the other nodes to “think” that the adversary is also a different one in that network. The adversary may use this attack to capture (or even alter) all packets within the LAN by impersonating the router, or they may choose to only target nodes.

The packet sniffer, shown within the dashed rectangle in Figure 1 is an addition to the usual software in your computer, and consists of two parts. The packet capture library receives a copy of every link layer frame that is sent from or received by your computer. Messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP are all eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In Figure 1, the assumed physical media is an Ethernet network, and so all upper layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you all messages sent/received from/by all protocols and applications executing in your computer.

The second component of a packet sniffer is the packet analyser, which displays the contents of all fields within a protocol message. In order to do so, the packet analyser must “understand” the structure of all messages exchanged by protocols. For example, the packet analyser understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. It understands the TCP segment structure, so it can extract the application protocol message contained in the TCP segment. Finally, if it understands the application protocol (typically if it is a common one such as HTTP) the packet analyser can extract some information from the application packet (such as reading the first bytes of an HTTP message, which must contain one of the strings “GET,” “POST,” or “HEAD”).

The following is a very brief introduction to Wireshark. For more information go to this site:

www.wireshark.org/docs/

2.2 Introduction to Wireshark

Note: This information is extracted from Wireshark User’s Guide. Please refer to the URL above for more details.

Wireshark is a network packet analyser. A network packet analyser will capture network packets and display that packet data in as detailed a manner as possible.

You can think of a network packet analyser as a measuring device used to examine what's going on inside a network cable, just like a voltmeter is used by an electrician to examine what's going on inside an electric cable (but at a higher level, of course). In the past, such tools were either very expensive, proprietary, or both. However, open-source and freeware examples of these tools are now available. Wireshark is perhaps one of the best open-source packet analysers available today. Wireshark is an open-source software project, and is released under the GNU General Public License (GPL). You can freely use Wireshark on any number of computers you like, without worrying about license keys or fees or such. In addition, all source code is freely available under the GPL. Because of that, it is very easy for people to add new protocols to Wireshark, either as plugins, or built into the source, and they often do!

2.2.1 A brief history of Wireshark

In late 1997, Gerald Combs needed a tool for tracking down networking problems and wanted to learn more about networking, so he started writing Ethereal (the former name of the Wireshark project) as a way to solve both problems.

Ethereal was initially released, after several pauses in development, in July 1998 as version 0.2.0. Within days, patches, bug reports, and words of encouragement started arriving, so Ethereal was on its way to success.

Not long after that, Gilbert Ramirez saw its potential and contributed a low-level dissector to it. In October 1998, Guy Harris of Network Appliance was looking for something better than TCPView, so he started applying patches and contributing dissectors to Ethereal.

In late 1998, Richard Sharpe, who was giving TCP/IP courses, saw its potential on such courses, and started looking at it to see if it supported the protocols he needed. While it didn't at that point, new protocols could be easily added. So, he started contributing dissectors and contributing patches. The list of people who have contributed to Ethereal has become very long since then, and almost all of them started with a protocol that they needed that Ethereal did not already handle. So, they copied an existing dissector and contributed the code back to the team.

In 2006 the project moved house and re-emerged under a new name: Wireshark.

Here are some examples people use Wireshark for:

- **network administrators use it to troubleshoot network problems**
- **network security engineers use it to examine security problems**
- **developers use it to debug protocol implementations**
- **people use it to learn network protocol internals**

Besides these examples, Wireshark can be helpful in many other situations too.

2.2.2 Features

The following are some of the many features Wireshark provides:

- Capture live packet data from a network interface.
- Display packets with very detailed protocol information.
- Open and Save packet data captured.
- Import and Export packet data from and to a lot of other capture programs.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.
- Create various statistics.

... and a lot more!

Here are some things Wireshark does not provide:

- Wireshark isn't an intrusion detection system. It will not warn you when someone does strange things on your network that he/she isn't allowed to do. However, if strange things happen, Wireshark might help you figure out what is really going on.
- Wireshark will not manipulate things on the network, it will only "measure" things from it.
- Wireshark doesn't send packets on the network or do other active things (except for name resolutions, but even that can be disabled).

2.2.3 Using Wireshark – Understanding the Capture Menu

The Wireshark interface has five major components:

- The **command menus** are standard pulldown menus located at the top of the window. Of interest to us now are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data and exit the Wireshark application. The Capture menu allows you to begin packet capture.
- The **packet-listing pane** displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is not a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest-level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.
- The **packet-header details pane** provides details about the packet selected (highlighted) in the packet listing window. (To select a packet in the packet listing window, place the cursor over the packet's one-line summary in the packet listing window and click with the left mouse button.). These details include information about the Ethernet frame and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the right-pointing or down-pointing arrowhead to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are also provided.
- The **packet-contents pane** displays the entire contents of the captured frame, in both ASCII and hexadecimal format.
- Towards the top of the Wireshark graphical user interface, is the **packet display filter field**, into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window (and hence the packet-header and packet contents windows).

In this exercise, we'll use the "Capture Menu" to capture packets that correspond to HTTP/HTTPS messages. Before we begin this exercise, we must first get familiar with the Capture Menu.

2.2.4 Capturing Packets Using Wireshark

Figure 2 shows the page of Wireshark once some packets have been captured.

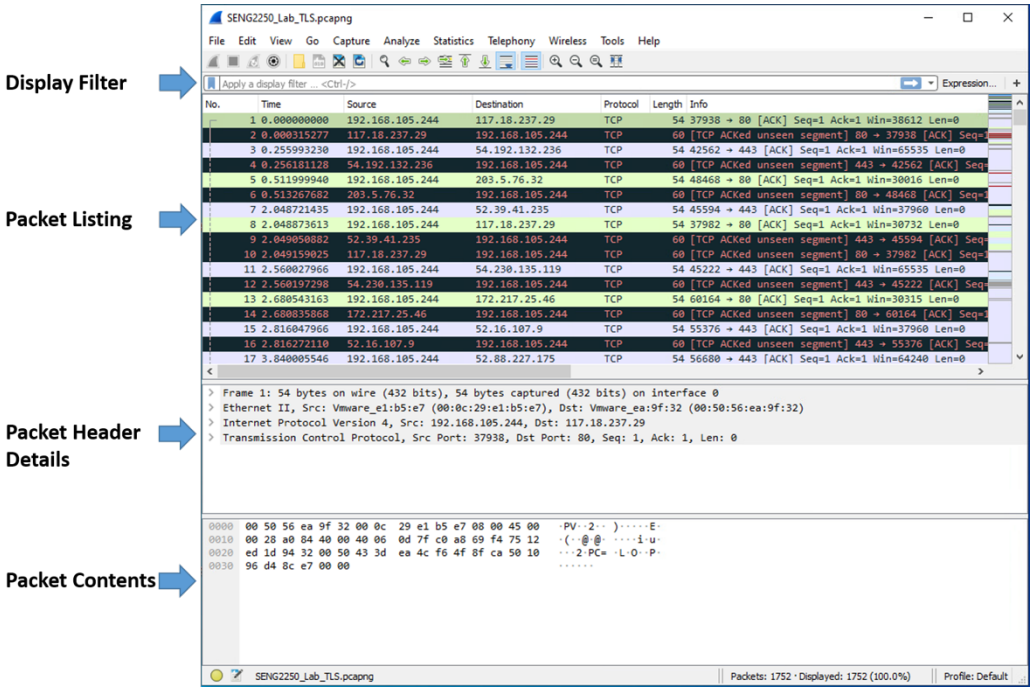


Figure 2: Wireshark Graphical User Interface

The **packet listing pane** shows all the network packets that have been captured while you have been running Wireshark in capture mode – one packet per line. When you click on a packet, that packet is displayed in different ways in the lower two panes.

The **bottom packet contents** pane shows the packet in raw format as it was captured. This information is very low level and mostly not very interesting. You can see three vertical sections. The first column gives you the offset of the octet. The next 16 two-digit columns show you the octet contents in hexadecimal (that is four-bit numbers 0- 15 are represented by 0-9, then A-F). You should get used to hexadecimal numbers for communications work since it represents four-bit entities neatly and we will be using hexadecimal more in future labs. The next section shows the same data as ASCII characters. Wireshark does the best it can displaying the characters, but not all 8-bit patterns (hexadecimal codes) correspond to printable characters, especially codes < 0x20. Non-printable characters are simply represented by a dot '.'.

The **middle packet-header details pane** is where you will look most. This pane displays Wireshark's interpretation of the packet. The lines here represent different layers of the network stack, albeit upside down to how we usually think of the network stack. The top line represents the physical layer. The second line the data-link layer where you will mainly see Ethernet packets, but it could be another data-link protocol. The third line is the network layer, which is mainly IP. The fourth line is the transport layer, normally TCP, but it could be UDP. If Wireshark finds an application protocol like HTTP that it can interpret, it will also give you information about that on the fifth line.

In the toolbar, you will find some handy shortcuts for starting and stopping captures. Since a capture can capture many packets, the filter text box allows you to enter criteria to display relevant packets.

Depending on how you would like to capture the packet, there are several ways to go about it.

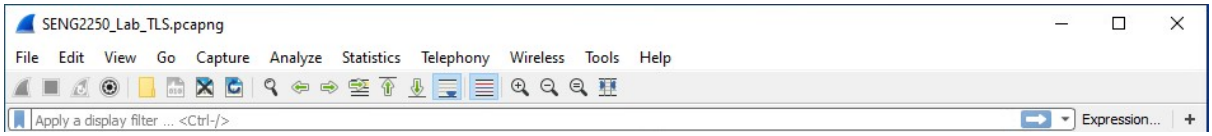


Figure 3: Wireshark Menu Bar

In Figure 3, choose the "Options" sub-menu in the "Capture" drop-down listing. Once you choose this, you will have the listing of various network interfaces on your machine that can be seen from your computer's NIC(s). An example is shown in Figure 4, where we have five interfaces.

Task 1: Capture Packets for HTTP/HTTPS Traffics

1. Log into your Windows VM then start Wireshark.
2. Capture network traffic on **Ethernet** interface. To start with, we can choose the "Capture" menu (Figure 4) and click "Options…". Then, select "Npcap Loopback Adapter" (Figure 5) and click Start. As the name implies, "Capture" menu is provided for the users to perform Packet Capture, and it also provides several options for suiting the situations and the conditions that the analysts have in mind while performing the process of capturing the packets. Analysts could even set filters to avoid capturing unwanted traffic.

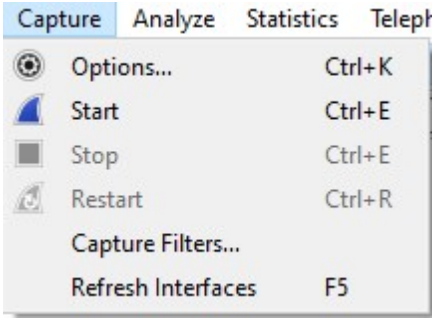


Figure 4: Wireshark Capture tab

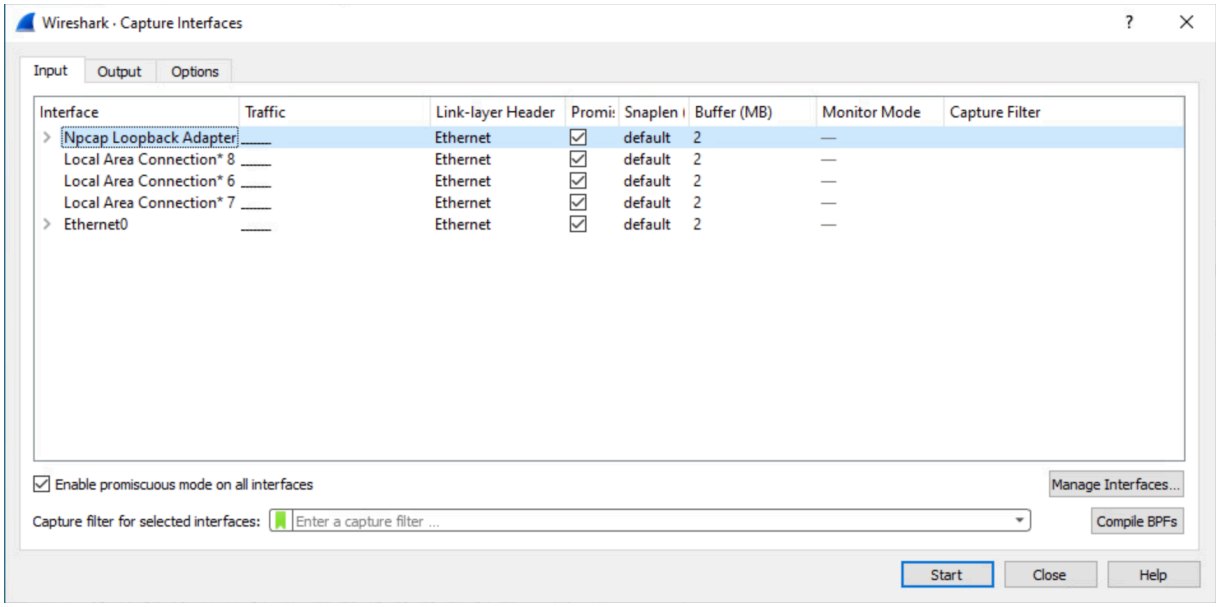


Figure 5: Wireshark Capture Interface

3. Start the Apache server. To accomplish this, run the XAMPP control panel at ("C:\xampp\xampp-control.exe"). Then, start the Apache server.

4. Open Firefox and access (the secure website created in Lab 09)

   https://localhost/dashboard

5. Stop capture in Wireshark.
6. Enter "tls" to the Display Filter textbox and find the TLS packets. Browse the captured TLS (handshake) packets, it should be similar to what you have inspected in Lab 09 (Figure 6).

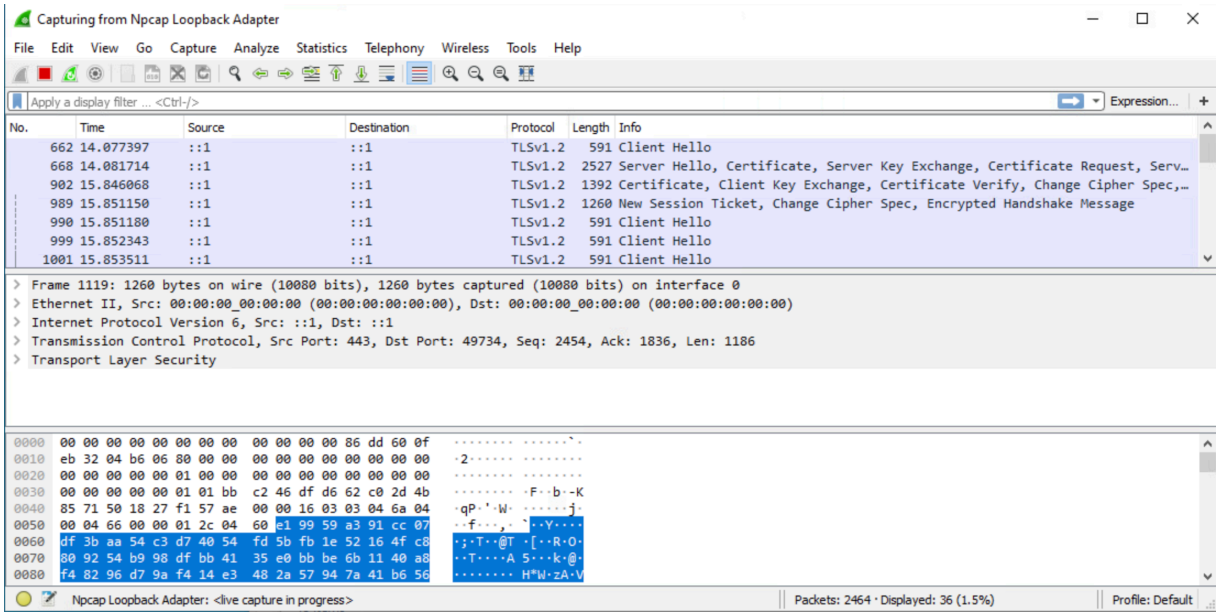


Figure 6: Captured TLS traffic

Part 2: Telnet and SSH Security Inspection

Task 2: Install and Enable Telnet on Windows

In this task, we will first install Telnet server and enable the services. As Telnet is not a secure remote control application, some of Windows versions removed it from the system. We need to install third party software for Telnet server. In practice, telnet should no longer be used for secure (control) communication, such as delivering username/password or sensitive information. On the other hand, Telnet Client is still widely supported in Windows, while this may need to be enabled prior to use.

To accomplish this task, we use KpyM Telnet Server (which is installed on the VM by the administrator).

1. Enable the Telnet server. Search "**Setup KpyM Telnet SSH Server**" in the Windows Start menu. Then choose "**start service**".

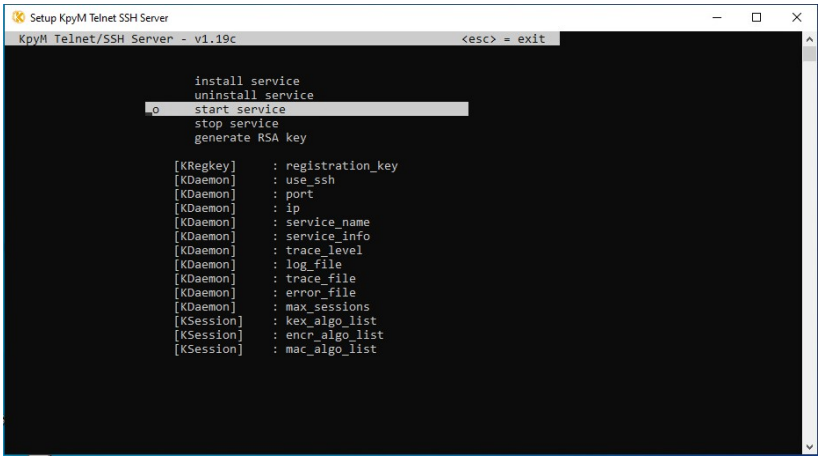


Figure 7: Start the telnet server.

2. Enable Telnet client. Go to Control Panel → Programs and Features → Turn Windows features on or off. Find the Telnet Client and tick it, then click OK. (You may need to restart your computer.)

Task 3: Capture and Inspect Telnet Traffics

In this task, you will capture and compare the network traffics of Telnet and SSH applications.

1. Open Wireshark and capture packets on the loopback interface.
2. Open PuTTY, use “localhost” as the Host Name. Select “Telnet” as the connection type, then click Open. You could also use “telnet” command from “Command Prompt”.
3. Login to the Telnet server. In this task, you can type any character string as username and password (it does not have to be real). Note: after entering the username and password, you may receive error messages, but it is fine in this task.
4. Go to Wireshark and search “telnet” in the Display Filter panel. Select one Telnet packet. Right-click and choose Follow → TCP Stream (Figure 8).

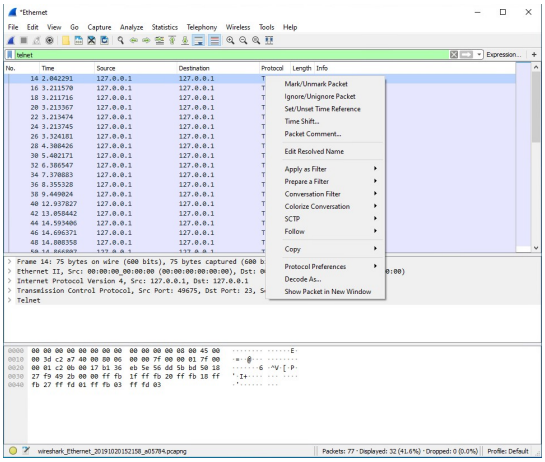


Figure 8: Capture Telnet packets.

Can you find the username and password you entered during the login phase?

Task 4: Capture and Inspect SSH Traffics

1. Uninstall KpyM Telnet Server and reinstall it as SSH server:

Uninstall service -> Select the use_ssh option and change the value from 0 to 1, and press enter to save -> Select the port option and change the value from 23 to 22, and press enter to save -> Install service -> start service

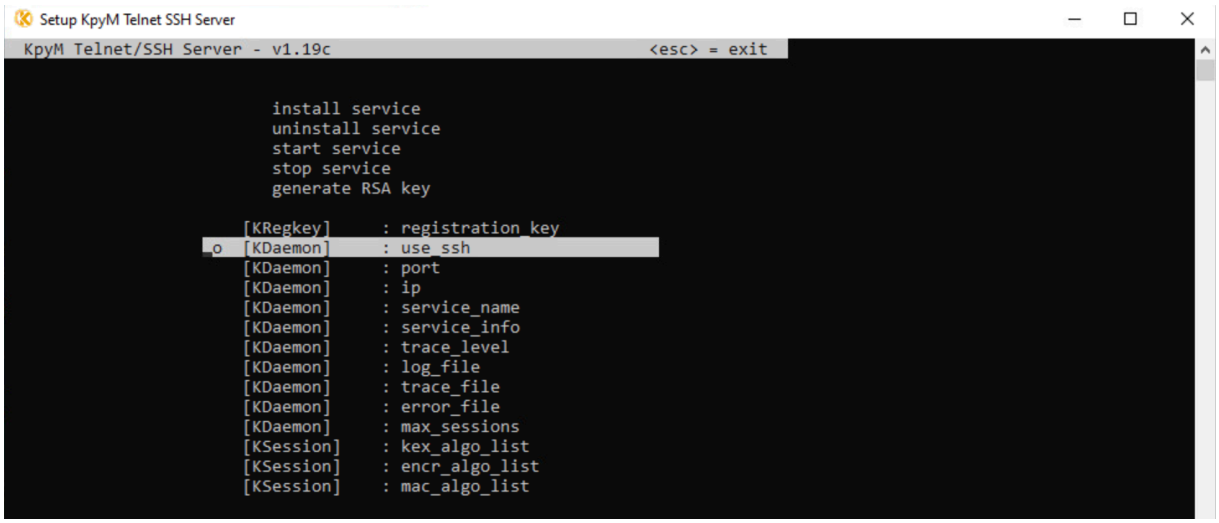


Figure 9: Reinstall KpyM telnet server

2. Open Wireshark and capture packets on the loopback interface.
3. Open PuTTY, use “localhost” as the Host Name. Select “SSH” as the connection type, then click Open.
4. Login to the server. In this task, you can type any character string as username and password (it does not have to be real). Note: after entering the username and password, you may receive error messages, but it is fine.

Any warning message prompts out? Why?

5. Stop Capturing and find the SSH packets.
    a. How does it do the handshake? Is it different from the SSL handshake? How?
    b. Can you find the username and password from the captured message?