CySA+ Lab Series

Lab 06: Packet Analysis

Document Version: 2022-10-10

Material in this Lab Aligns to the Following	
CompTIA CySA+ (CS0-002) Exam Objectives	1.2 - Given a scenario, utilize threat intelligence to support organizational security 1.3 - Given a scenario, perform vulnerability managemenactivitieses 1.4 - Given a scenario, analyze the output from common vulnerability tools 3.1 - Given a scenario, analyze data as part of security monitoring activities 3.2 - Given a scenario, implement configuration changes to existing controls to improve security 3.3 - Explain the importance of proactive threat hunting 4.4 - Given a scenario, utilize basic digital forensics techniques
All-In-One CompTIA CySA+ Second Edition ISBN-13: 978-1260464306 Chapters	2: Threat Intelligence in Support of Organizational Security 3: Vulnerability Management Activities 4: Vulnerability Assessment Tools 11: Data Analysis in Security Monitoring Activities 12: Implement Configuration Changes to Existing Controls to Improve Security 13: The Importance of Proactive Threat Hunting 18: Utilize Basic Digital Forensics Techniques

Contents

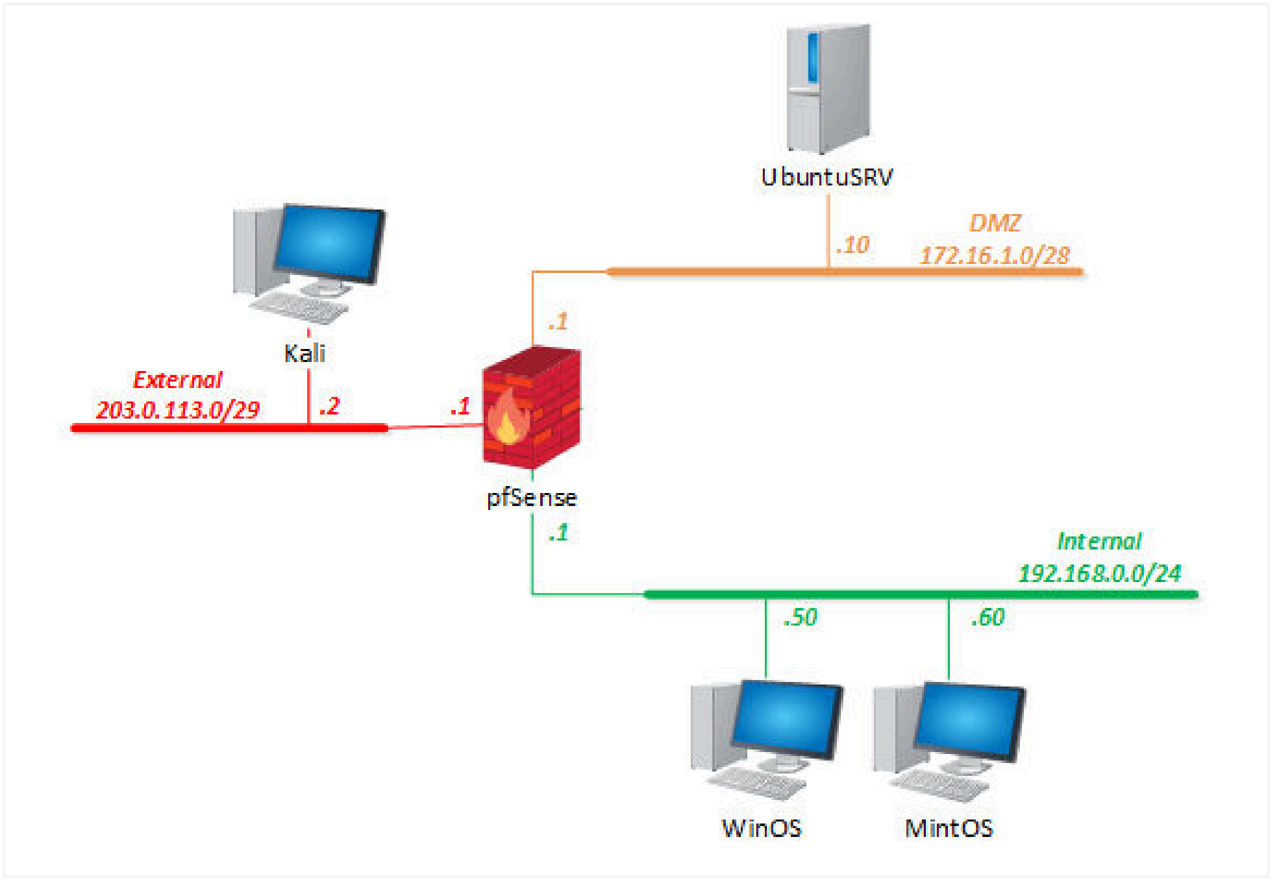
Introduction

In this lab, you will explore various methods for recording and analyzing network traffic using both the CLI and GUI.

Objective

- Capture packets with TCPDUMP
- Capture and analyze packets with Wireshark
- View TCP Streams
- Analyze encrypted traffic

Lab Topology



Lab Settings

The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

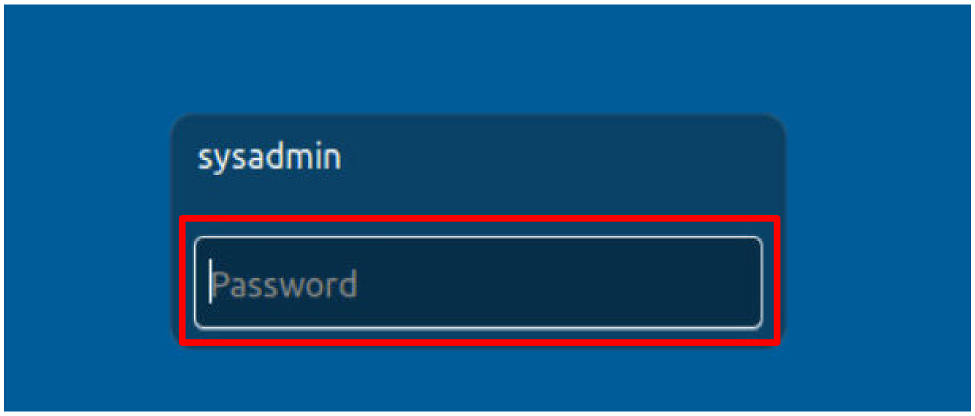
Virtual Machine	IP Address	Account	Password
WinOS (Server 2019)	`192.168.0.50`	`Administrator`	`NDGlabpass123!`
MintOS (Linux Mint)	`192.168.0.60`	`sysadmin`	`NDGlabpass123!`
OSSIM (Alien Vault)	`172.16.1.2`	`root`	`NDGlabpass123!`
UbuntuSRV (Ubuntu Server)	`172.16.1.10`	`sysadmin`	`NDGlabpass123!`
Kali	`203.0.113.2`	`sysadmin`	`NDGlabpass123!`

pfSense	`203.0.113.1`
	`172.16.1.1`
	`192.168.0.1`
		`admin`	`NDGlabpass123!`

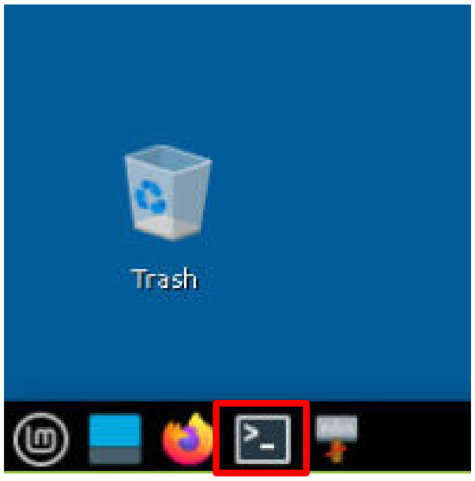
1 Analyze Network Traffic Using TCPDUMP and Wireshark

tcpdump is a command-line utility that is used to capture packets passing through the network interface card for analysis. The packets that are captured during the *tcpdump* generate network activity and can be saved in a **pcap** (packet capture) file for analysis using *Wireshark*.

1. Set the focus to the **MintOS** computer.
2. Log in as **sysadmin** using the password: `NDGlabpass123!`



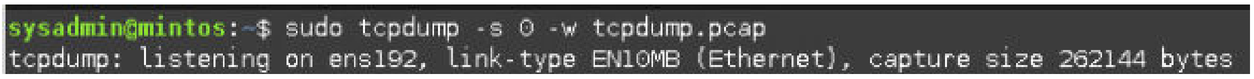
3. Click on the **Terminal** icon in the taskbar at the bottom of the screen.



4. The *tcpdump* packet analyzer opens a monitoring session on a selected interface. Packets are copied into a **pcap** file where they can be analyzed. Start *tcpdump* using the default packet capture size as 262,144 bytes (**-s 0** for snapshot length) and write the raw packets to the file **tcpdump.pcap** (**-w**) by typing the following command:

```
sudo tcpdump –s 0 –w tcpdump.pcap
```

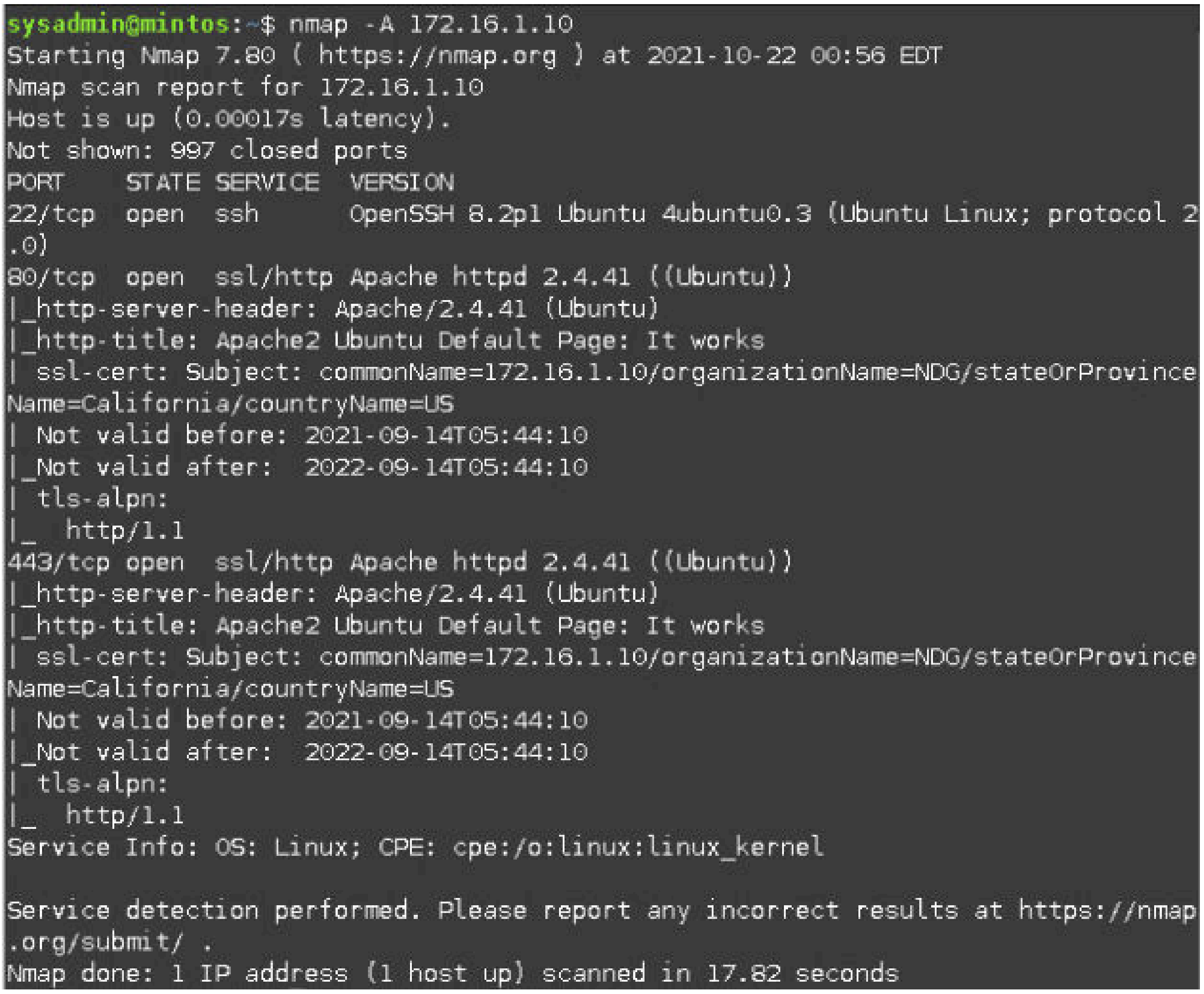
5. When asked for the **[sudo]** password, type: `NDGlabpass123!`



```
sysadmin@mintos:~$ sudo tcpdump -s 0 -w tcpdump.pcap
tcpdump: listening on ens192, link-type EN10MB (Ethernet), capture size 262144 bytes
```

6. While leaving that terminal window open, open a second terminal window by clicking on the **Terminal** icon in the taskbar.
7. Generate network traffic by typing the following command:

```
nmap –A 172.16.1.10
```

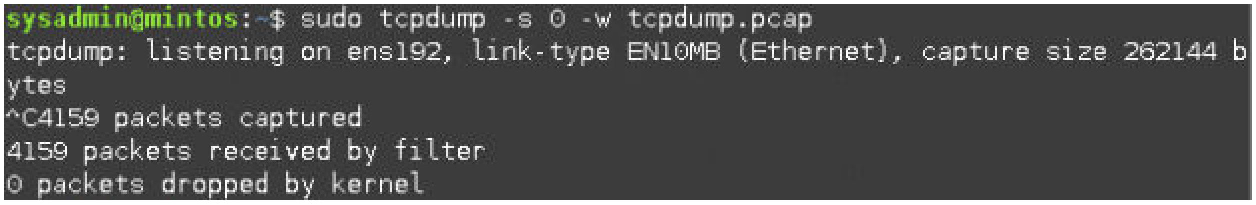


```
sysadmin@mintos:~$ nmap -A 172.16.1.10
Starting Nmap 7.80 ( https://nmap.org ) at 2021-10-22 00:56 EDT
Nmap scan report for 172.16.1.10
Host is up (0.00017s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2
.0)
80/tcp  open  ssl/http Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
| ssl-cert: Subject: commonName=172.16.1.10/organizationName=NDG/stateOrProvince
Name=California/countryName=US
| Not valid before: 2021-09-14T05:44:10
|_Not valid after:  2022-09-14T05:44:10
| tls-alpn:
|_  http/1.1
443/tcp open  ssl/http Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
| ssl-cert: Subject: commonName=172.16.1.10/organizationName=NDG/stateOrProvince
Name=California/countryName=US
| Not valid before: 2021-09-14T05:44:10
|_Not valid after:  2022-09-14T05:44:10
| tls-alpn:
|_  http/1.1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.82 seconds
```

It might take up to 2 minutes for the scan to complete.

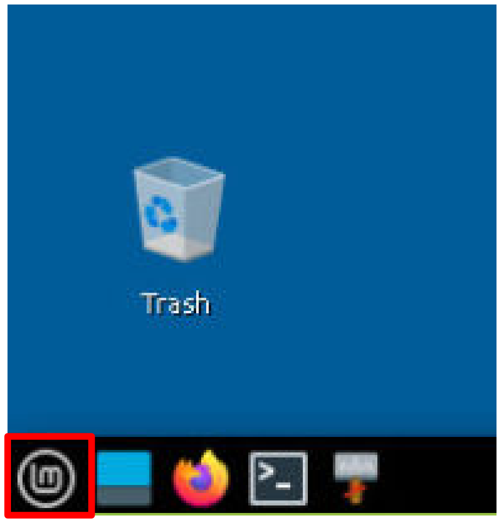
To make the scan time shorter and the dump file smaller and easier to navigate, we are only scanning one IP address, the address of the *UbuntuSRV* computer. Normally, you would scan the whole subnet.

8. When the scan has completed, go back to the terminal window that has *tcpdump* open and stop the process by pressing **CTRL+C**.

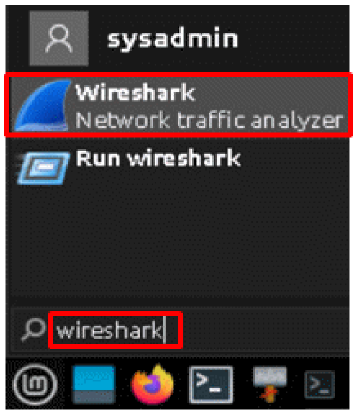


```
sysadmin@mintos:~$ sudo tcpdump -s 0 -w tcpdump.pcap
tcpdump: listening on ens192, link-type EN10MB (Ethernet), capture size 262144 b
ytes
^C4159 packets captured
4159 packets received by filter
0 packets dropped by kernel
```

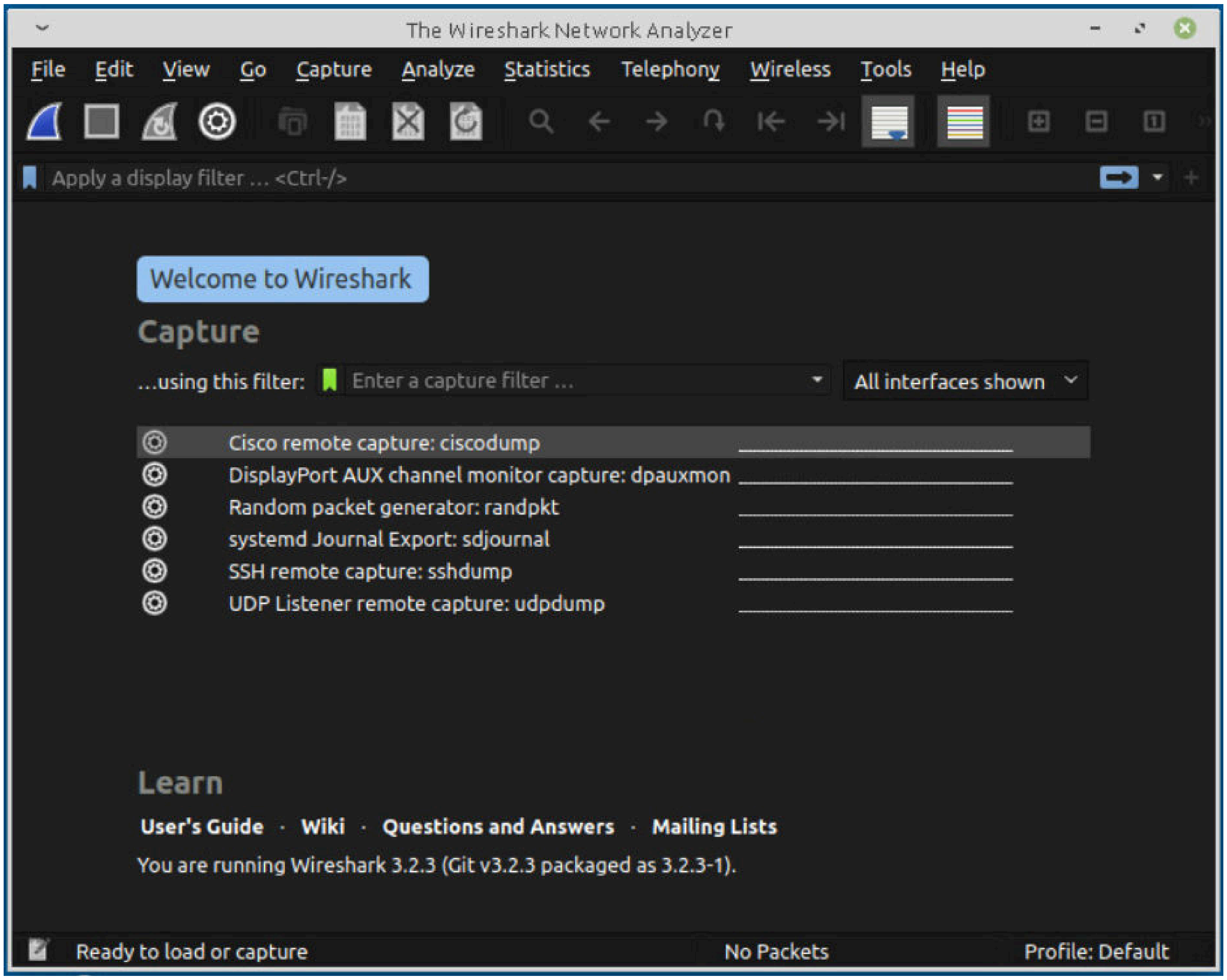
9. Close the two terminal windows.
10. Click on the **Start** button in the lower-left corner of the screen.



11. In the search box, type `wireshark`. When the results are shown, click on the **Wireshark** link.



12. You will see the *Welcome to Wireshark* window.

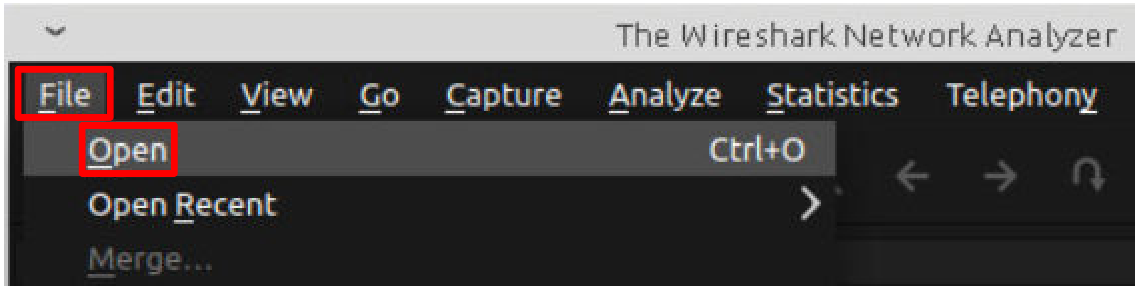

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Welcome to Wireshark
Capture
…using this filter:
Enter a capture filter …
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists

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

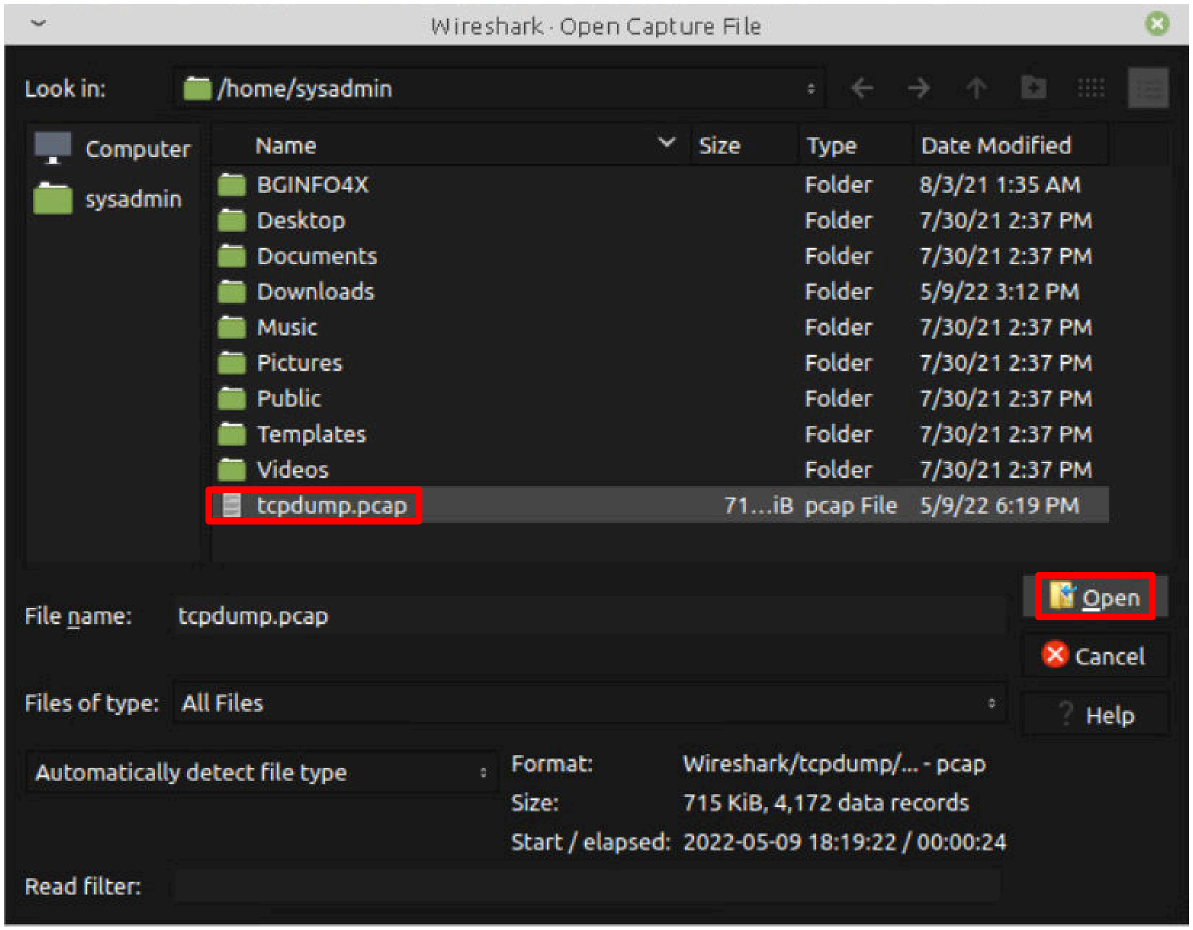
You are running Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1).

Ready to load or capture   No Packets   Profile: Default

13. On the *Wireshark Network Analyzer* window, click on **File** in the menu bar, and then click **Open**.


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony
Open Ctrl+O
Open Recent
Merge…

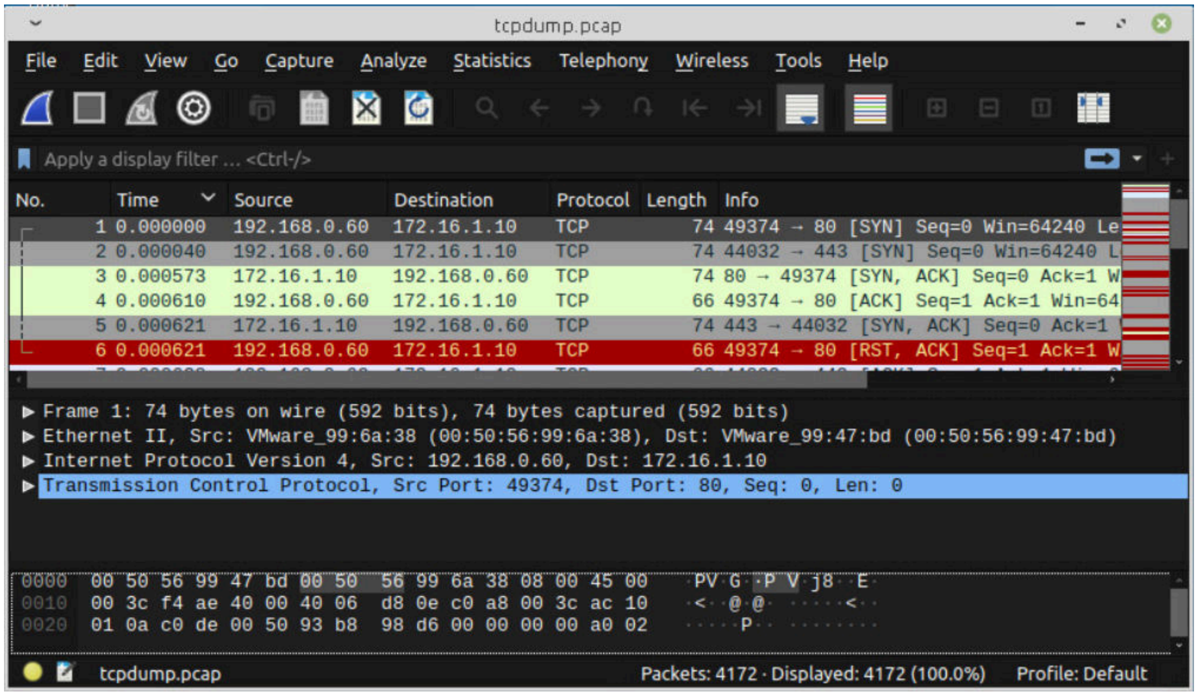
14. In the *Wireshark - Open Capture File* window, at the bottom of the file list, select the **tcpdump.pcap** file and then click on the **Open** button.



Name	Size	Type	Date Modified
BGINFO4X		Folder	8/3/21 1:35 AM
Desktop		Folder	7/30/21 2:37 PM
Documents		Folder	7/30/21 2:37 PM
Downloads		Folder	5/9/22 3:12 PM
Music		Folder	7/30/21 2:37 PM
Pictures		Folder	7/30/21 2:37 PM
Public		Folder	7/30/21 2:37 PM
Templates		Folder	7/30/21 2:37 PM
Videos		Folder	7/30/21 2:37 PM
tcpdump.pcap	71…iB	pcap File	5/9/22 6:19 PM

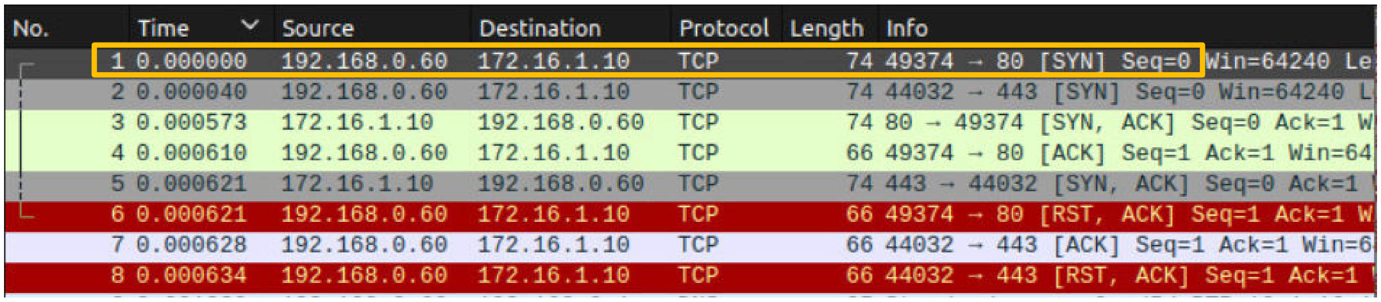
File name: tcpdump.pcap
Files of type: All Files
Automatically detect file type
Format: Wireshark/tcpdump/... - pcap
Size: 715 KiB, 4,172 data records
Start / elapsed: 2022-05-09 18:19:22 / 00:00:24
Read filter:

Looking at the results of the capture, we can pull quite a bit of information about the session.



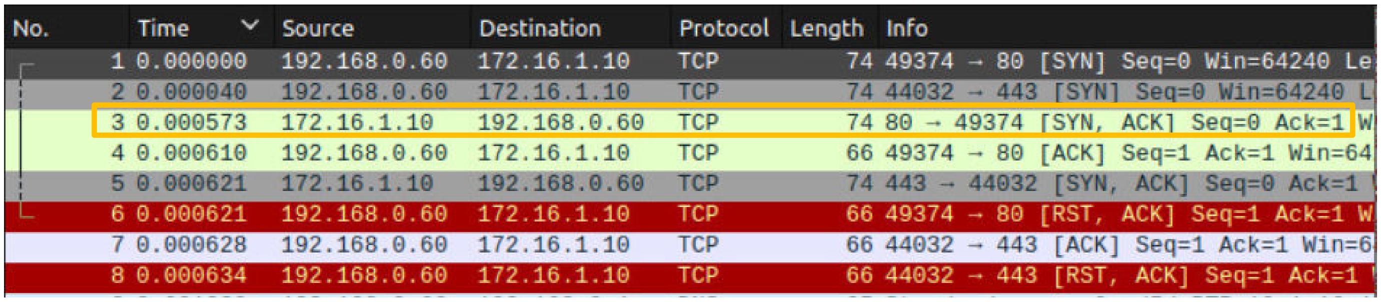
```
No.  Time      Source        Destination   Protocol Length Info
   1 0.000000  192.168.0.60  172.16.1.10   TCP      74 49374 → 80 [SYN] Seq=0 Win=64240 Le
   2 0.000040  192.168.0.60  172.16.1.10   TCP      74 44032 → 443 [SYN] Seq=0 Win=64240 L
   3 0.000573  172.16.1.10   192.168.0.60  TCP      74 80 → 49374 [SYN, ACK] Seq=0 Ack=1 W
   4 0.000610  192.168.0.60  172.16.1.10   TCP      66 49374 → 80 [ACK] Seq=1 Ack=1 Win=64
   5 0.000621  172.16.1.10   192.168.0.60  TCP      74 443 → 44032 [SYN, ACK] Seq=0 Ack=1
   6 0.000621  192.168.0.60  172.16.1.10   TCP      66 49374 → 80 [RST, ACK] Seq=1 Ack=1 W

▶ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▶ Ethernet II, Src: VMware_99:6a:38 (00:50:56:99:6a:38), Dst: VMware_99:47:bd (00:50:56:99:47:bd)
▶ Internet Protocol Version 4, Src: 192.168.0.60, Dst: 172.16.1.10
▶ Transmission Control Protocol, Src Port: 49374, Dst Port: 80, Seq: 0, Len: 0

0000  00 50 56 99 47 bd 00 50  56 99 6a 38 08 00 45 00   ·PV·G··P V·j8··E·
0010  00 3c f4 ae 40 00 40 06  d8 0e c0 a8 00 3c ac 10   ·<··@·@· ·····<··
0020  01 0a c0 de 00 50 93 b8  98 d6 00 00 00 00 a0 02   ·····P·· ········

tcpdump.pcap          Packets: 4172 · Displayed: 4172 (100.0%)     Profile: Default
```

15. When testing for open ports, *nmap* will send a **SYN** from the *MintOS* computer at **192.168.0.60** to *UbuntuSRV* at **172.16.1.10**, port **80**. Scroll down the list of packets in the top pane and find the first **SYN** packet.



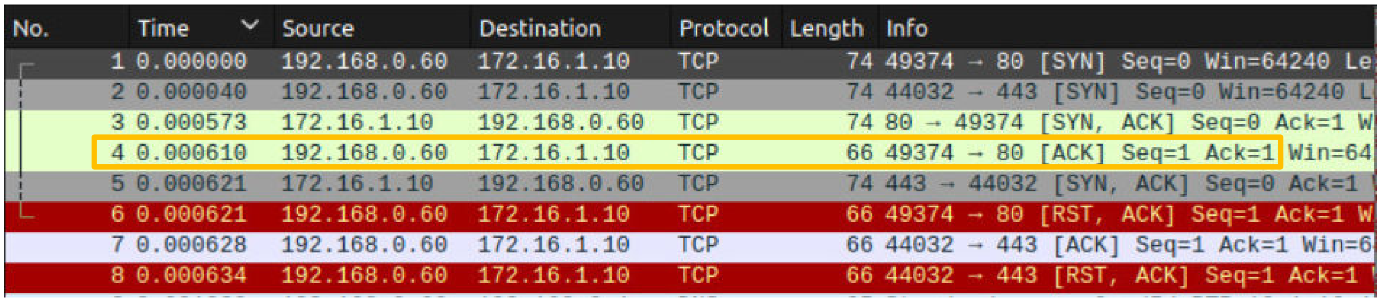
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.60	172.16.1.10	TCP	74	49374 → 80 [SYN] Seq=0 Win=64240 Le
2	0.000040	192.168.0.60	172.16.1.10	TCP	74	44032 → 443 [SYN] Seq=0 Win=64240 L
3	0.000573	172.16.1.10	192.168.0.60	TCP	74	80 → 49374 [SYN, ACK] Seq=0 Ack=1 W
4	0.000610	192.168.0.60	172.16.1.10	TCP	66	49374 → 80 [ACK] Seq=1 Ack=1 Win=64
5	0.000621	172.16.1.10	192.168.0.60	TCP	74	443 → 44032 [SYN, ACK] Seq=0 Ack=1
6	0.000621	192.168.0.60	172.16.1.10	TCP	66	49374 → 80 [RST, ACK] Seq=1 Ack=1 W
7	0.000628	192.168.0.60	172.16.1.10	TCP	66	44032 → 443 [ACK] Seq=1 Ack=1 Win=6
8	0.000634	192.168.0.60	172.16.1.10	TCP	66	44032 → 443 [RST, ACK] Seq=1 Ack=1

If the port is open, the *UbuntuSRV* will reply with a **SYN-ACK** to the *MintOS* computer.



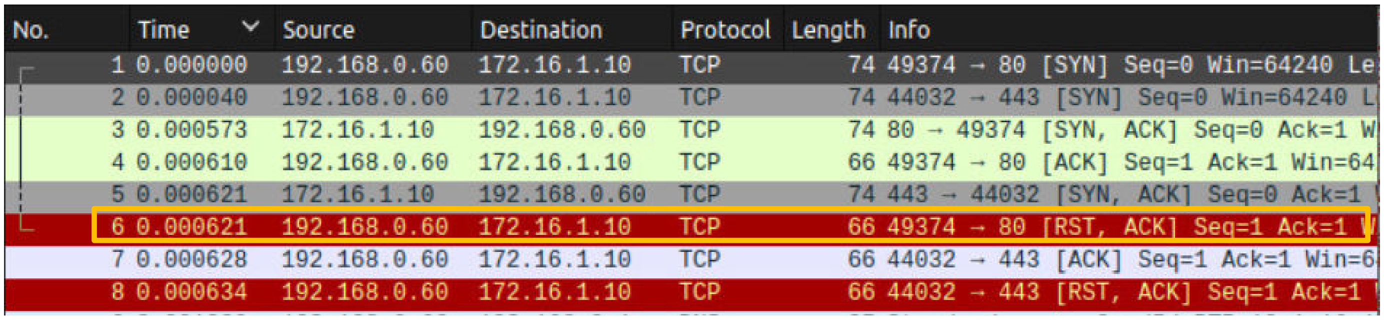
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.60	172.16.1.10	TCP	74	49374 → 80 [SYN] Seq=0 Win=64240 Le
2	0.000040	192.168.0.60	172.16.1.10	TCP	74	44032 → 443 [SYN] Seq=0 Win=64240 L
3	0.000573	172.16.1.10	192.168.0.60	TCP	74	80 → 49374 [SYN, ACK] Seq=0 Ack=1 W
4	0.000610	192.168.0.60	172.16.1.10	TCP	66	49374 → 80 [ACK] Seq=1 Ack=1 Win=64
5	0.000621	172.16.1.10	192.168.0.60	TCP	74	443 → 44032 [SYN, ACK] Seq=0 Ack=1
6	0.000621	192.168.0.60	172.16.1.10	TCP	66	49374 → 80 [RST, ACK] Seq=1 Ack=1 W
7	0.000628	192.168.0.60	172.16.1.10	TCP	66	44032 → 443 [ACK] Seq=1 Ack=1 Win=6
8	0.000634	192.168.0.60	172.16.1.10	TCP	66	44032 → 443 [RST, ACK] Seq=1 Ack=1

Then *nmap* will send an **ACK** from the *MintOS* computer to the *UbuntuSRV* computer.



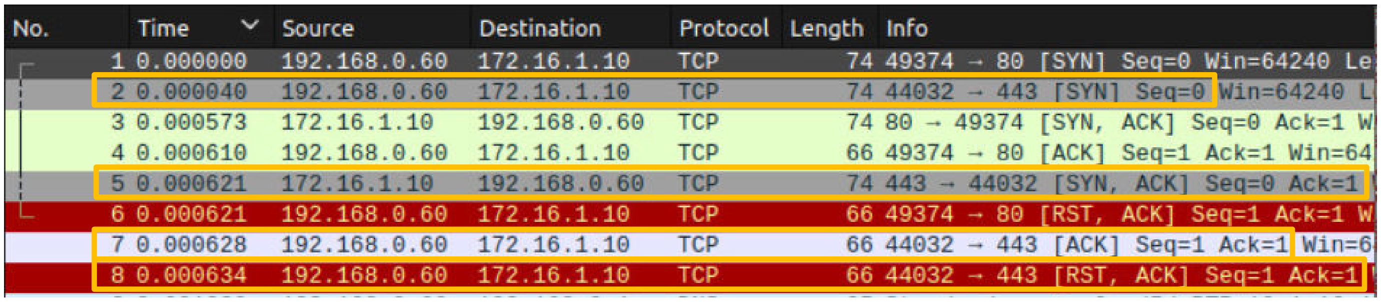
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.60	172.16.1.10	TCP	74	49374 → 80 [SYN] Seq=0 Win=64240 Le
2	0.000040	192.168.0.60	172.16.1.10	TCP	74	44032 → 443 [SYN] Seq=0 Win=64240 L
3	0.000573	172.16.1.10	192.168.0.60	TCP	74	80 → 49374 [SYN, ACK] Seq=0 Ack=1 W
4	0.000610	192.168.0.60	172.16.1.10	TCP	66	49374 → 80 [ACK] Seq=1 Ack=1 Win=64
5	0.000621	172.16.1.10	192.168.0.60	TCP	74	443 → 44032 [SYN, ACK] Seq=0 Ack=1
6	0.000621	192.168.0.60	172.16.1.10	TCP	66	49374 → 80 [RST, ACK] Seq=1 Ack=1 W
7	0.000628	192.168.0.60	172.16.1.10	TCP	66	44032 → 443 [ACK] Seq=1 Ack=1 Win=6
8	0.000634	192.168.0.60	172.16.1.10	TCP	66	44032 → 443 [RST, ACK] Seq=1 Ack=1

Once *nmap* finds that port 80 is open, it is no longer interested in the open port, so it sends back an **RST** to close the connection and moves on to scanning the next port.



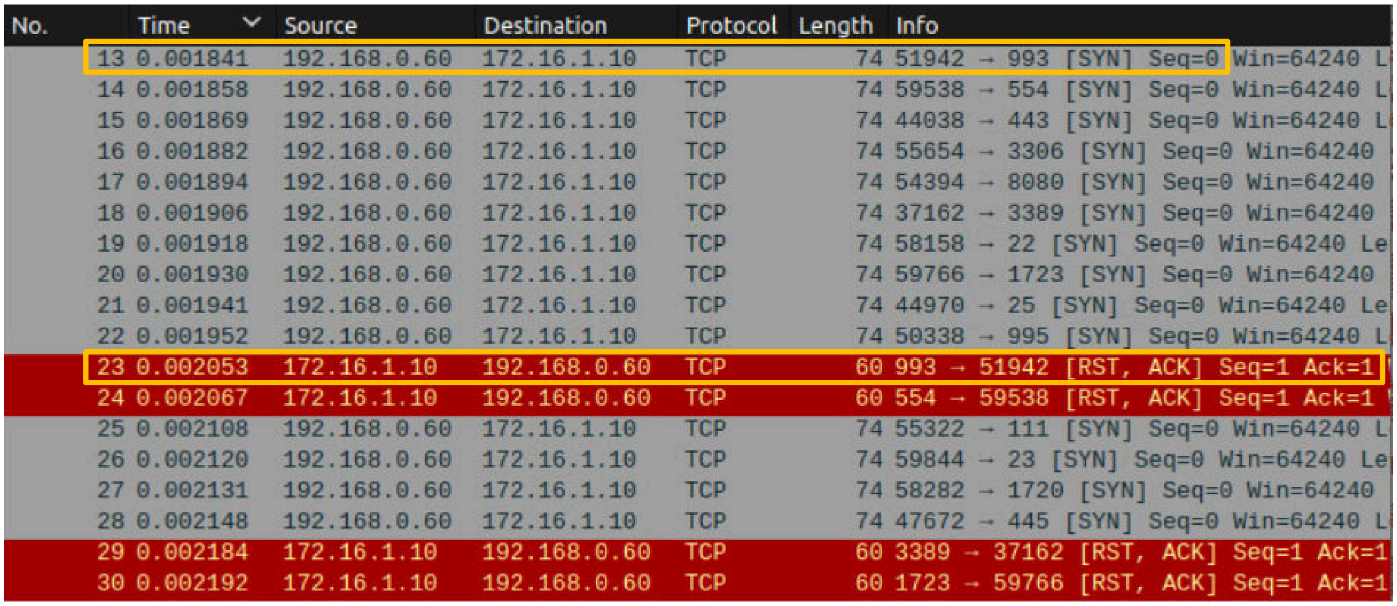
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.60	172.16.1.10	TCP	74	49374 → 80 [SYN] Seq=0 Win=64240 Le
2	0.000040	192.168.0.60	172.16.1.10	TCP	74	44032 → 443 [SYN] Seq=0 Win=64240 L
3	0.000573	172.16.1.10	192.168.0.60	TCP	74	80 → 49374 [SYN, ACK] Seq=0 Ack=1 W
4	0.000610	192.168.0.60	172.16.1.10	TCP	66	49374 → 80 [ACK] Seq=1 Ack=1 Win=64
5	0.000621	172.16.1.10	192.168.0.60	TCP	74	443 → 44032 [SYN, ACK] Seq=0 Ack=1
6	0.000621	192.168.0.60	172.16.1.10	TCP	66	49374 → 80 [RST, ACK] Seq=1 Ack=1 W
7	0.000628	192.168.0.60	172.16.1.10	TCP	66	44032 → 443 [ACK] Seq=1 Ack=1 Win=6
8	0.000634	192.168.0.60	172.16.1.10	TCP	66	44032 → 443 [RST, ACK] Seq=1 Ack=1

Looking at the output again, you can also see that port **443** is also open.



```
No.   Time      Source        Destination   Protocol Length Info
    1 0.000000  192.168.0.60  172.16.1.10   TCP         74 49374 → 80 [SYN] Seq=0 Win=64240 Le
    2 0.000040  192.168.0.60  172.16.1.10   TCP         74 44032 → 443 [SYN] Seq=0 Win=64240 L
    3 0.000573  172.16.1.10   192.168.0.60  TCP         74 80 → 49374 [SYN, ACK] Seq=0 Ack=1 W
    4 0.000610  192.168.0.60  172.16.1.10   TCP         66 49374 → 80 [ACK] Seq=1 Ack=1 Win=64
    5 0.000621  172.16.1.10   192.168.0.60  TCP         74 443 → 44032 [SYN, ACK] Seq=0 Ack=1
    6 0.000621  192.168.0.60  172.16.1.10   TCP         66 49374 → 80 [RST, ACK] Seq=1 Ack=1 W
    7 0.000628  192.168.0.60  172.16.1.10   TCP         66 44032 → 443 [ACK] Seq=1 Ack=1 Win=6
    8 0.000634  192.168.0.60  172.16.1.10   TCP         66 44032 → 443 [RST, ACK] Seq=1 Ack=1
```

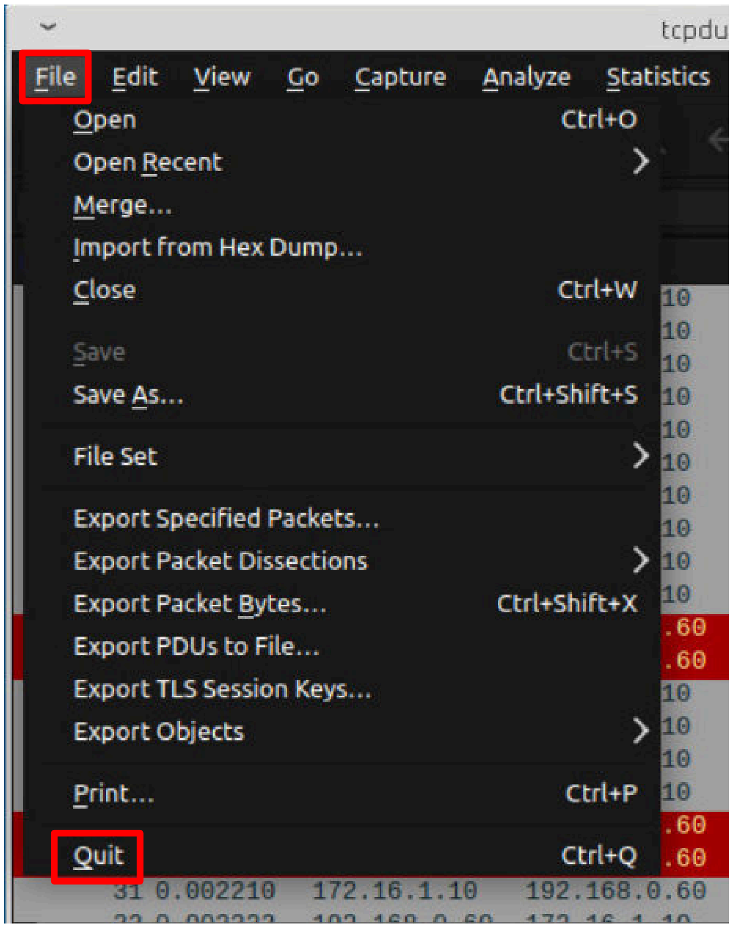
16. If you scroll down on the top pane, you will see when *nmap* sent a **SYN** packet from the *MintOS* computer at **192.168.0.60** to port **993** on the *UbuntuSRV* at **172.16.1.10**; the *UbuntuSRV* computer returned an **RST-ACK** packet indicating that port **993** is closed.



```
No.   Time      Source        Destination   Protocol Length Info
   13 0.001841  192.168.0.60  172.16.1.10   TCP         74 51942 → 993 [SYN] Seq=0 Win=64240 L
   14 0.001858  192.168.0.60  172.16.1.10   TCP         74 59538 → 554 [SYN] Seq=0 Win=64240 L
   15 0.001869  192.168.0.60  172.16.1.10   TCP         74 44038 → 443 [SYN] Seq=0 Win=64240 L
   16 0.001882  192.168.0.60  172.16.1.10   TCP         74 55654 → 3306 [SYN] Seq=0 Win=64240
   17 0.001894  192.168.0.60  172.16.1.10   TCP         74 54394 → 8080 [SYN] Seq=0 Win=64240
   18 0.001906  192.168.0.60  172.16.1.10   TCP         74 37162 → 3389 [SYN] Seq=0 Win=64240
   19 0.001918  192.168.0.60  172.16.1.10   TCP         74 58158 → 22 [SYN] Seq=0 Win=64240 Le
   20 0.001930  192.168.0.60  172.16.1.10   TCP         74 59766 → 1723 [SYN] Seq=0 Win=64240
   21 0.001941  192.168.0.60  172.16.1.10   TCP         74 44970 → 25 [SYN] Seq=0 Win=64240 Le
   22 0.001952  192.168.0.60  172.16.1.10   TCP         74 50338 → 995 [SYN] Seq=0 Win=64240 L
   23 0.002053  172.16.1.10   192.168.0.60  TCP         60 993 → 51942 [RST, ACK] Seq=1 Ack=1
   24 0.002067  172.16.1.10   192.168.0.60  TCP         60 554 → 59538 [RST, ACK] Seq=1 Ack=1
   25 0.002108  192.168.0.60  172.16.1.10   TCP         74 55322 → 111 [SYN] Seq=0 Win=64240 L
   26 0.002120  192.168.0.60  172.16.1.10   TCP         74 59844 → 23 [SYN] Seq=0 Win=64240 Le
   27 0.002131  192.168.0.60  172.16.1.10   TCP         74 58282 → 1720 [SYN] Seq=0 Win=64240
   28 0.002148  192.168.0.60  172.16.1.10   TCP         74 47672 → 445 [SYN] Seq=0 Win=64240 L
   29 0.002184  172.16.1.10   192.168.0.60  TCP         60 3389 → 37162 [RST, ACK] Seq=1 Ack=1
   30 0.002192  172.16.1.10   192.168.0.60  TCP         60 1723 → 59766 [RST, ACK] Seq=1 Ack=1
```

nmap also sent **SYN** packets to the *UbuntuSRV* at ports **554**, **3389,** and **1723** and the *UbuntuSRV* returned **RST-ACK** packets to the *MintOS* computer.

17. Close *Wireshark* by clicking on **File** in the menu bar and then click on **Quit.**



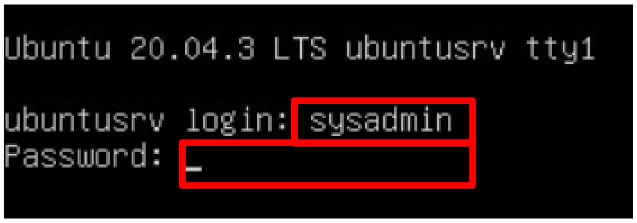
```
31 0.002210   172.16.1.10   192.168.0.60
```

2 TCP Handshake Using Scapy

"Scapy is a powerful interactive packet manipulation program. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. It can easily handle most classical tasks like scanning, tracerouting, probing, unit tests, attacks or network discovery."

https://scapy.net/

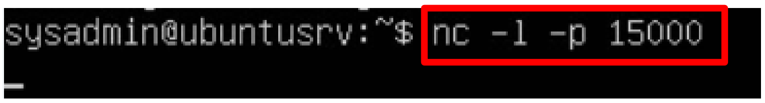
1. We need to open a port on the *UbuntuSRV* as a target for the *Scapy* crafted packet using *netcat Listener* to provide an open communication channel. Change the focus to the **UbuntuSRV** computer to open the server page and log in using the account `sysadmin` with the password: `NDGlabpass!`



```
Ubuntu 20.04.3 LTS ubuntusrv tty1

ubuntusrv login: sysadmin
Password: _
```

2. Type the following command to open port **15000**:

```
nc -l -p 15000
```



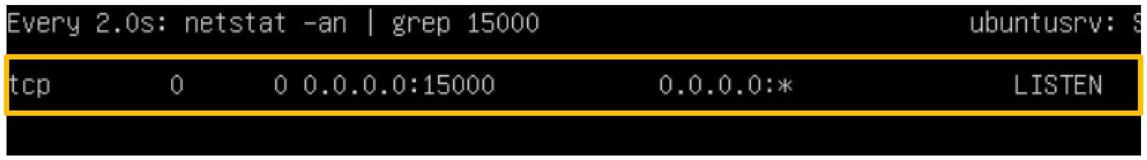
```
sysadmin@ubuntusrv:~$ nc -l -p 15000
_
```

You will not see any response after typing the command. This might come as a surprise when first encountered. You might think the system has crashed, but it hasn't.

3. Press **ALT+F2** to open another terminal session and log in to the session with the username `sysadmin` and the password: `NDGlabpass123!`

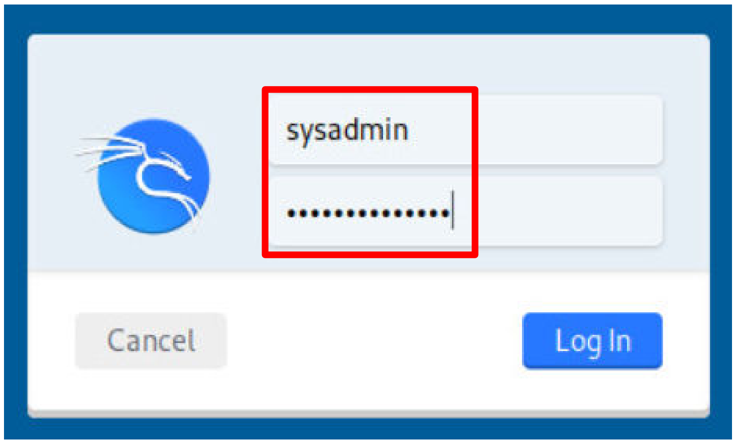
4. Type the following command to get a real-time, constantly updated report of connections to port **15000**.

```
watch "netstat -an | grep 15000"
```

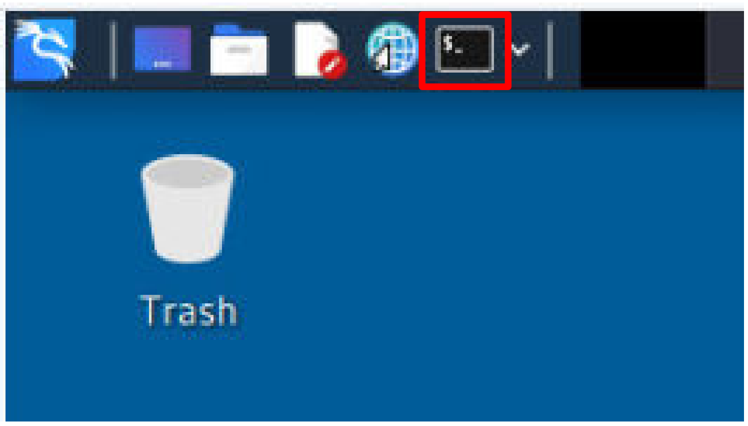


```
Every 2.0s: netstat -an | grep 15000                          ubuntusrv: S

tcp        0      0 0.0.0.0:15000           0.0.0.0:*               LISTEN
```

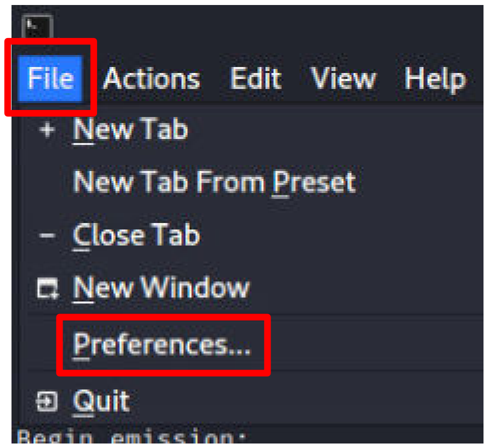
5. Change the focus to the **Kali** computer.
6. Log in as `sysadmin` using the password: `NDGlabpass123!`



7. Click on the **Terminal** icon in the taskbar at the top of the screen.

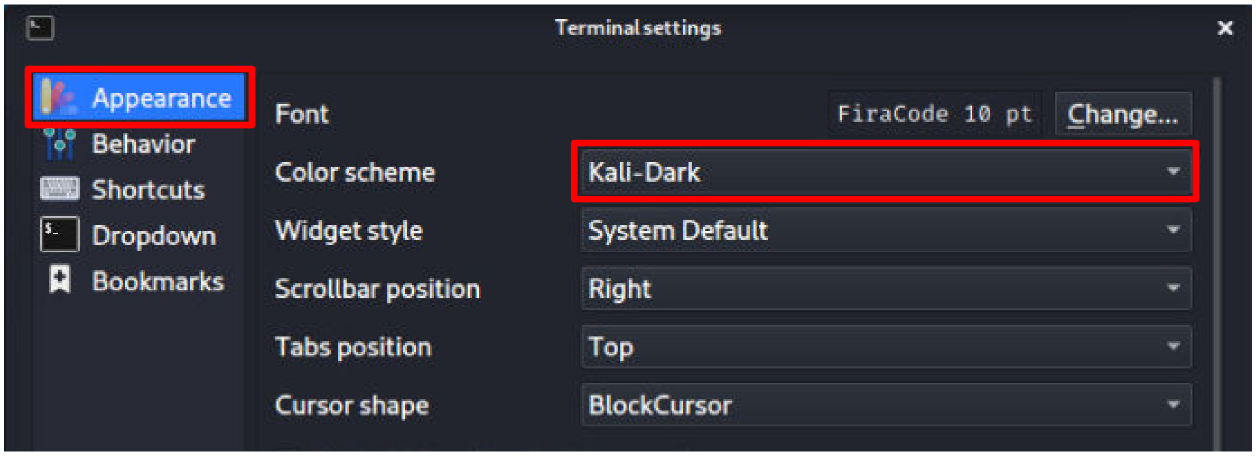


8. Change the appearance of the terminal window to make the output easier to read. In the menu of the terminal window, click **File>Preferences**.

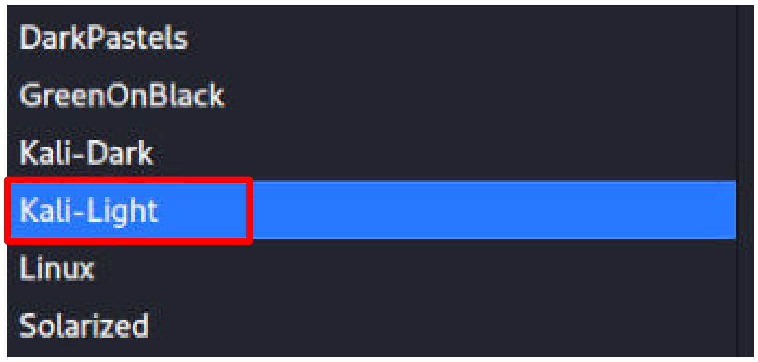


```
Begin emission:
```

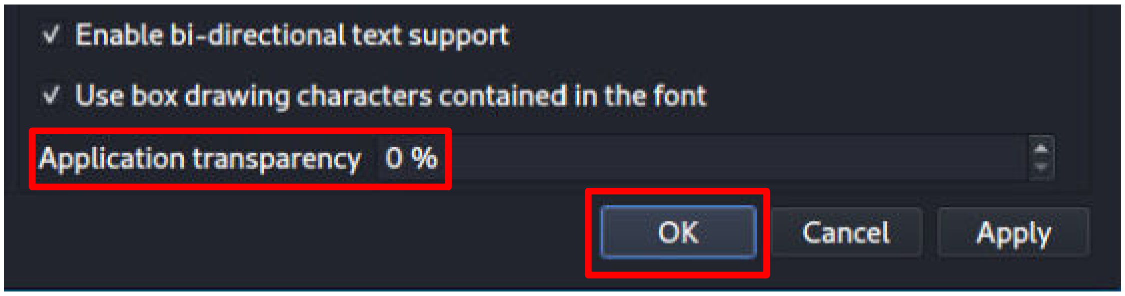
9. Select the **Appearance** menu option on the left of the window if it is not already selected. On the right side of the window, click the **list arrow** in the *Color Scheme* field. Then select **Kali-Light**.



Font FiraCode 10 pt
Color scheme Kali-Dark
Widget style System Default
Scrollbar position Right
Tabs position Top
Cursor shape BlockCursor



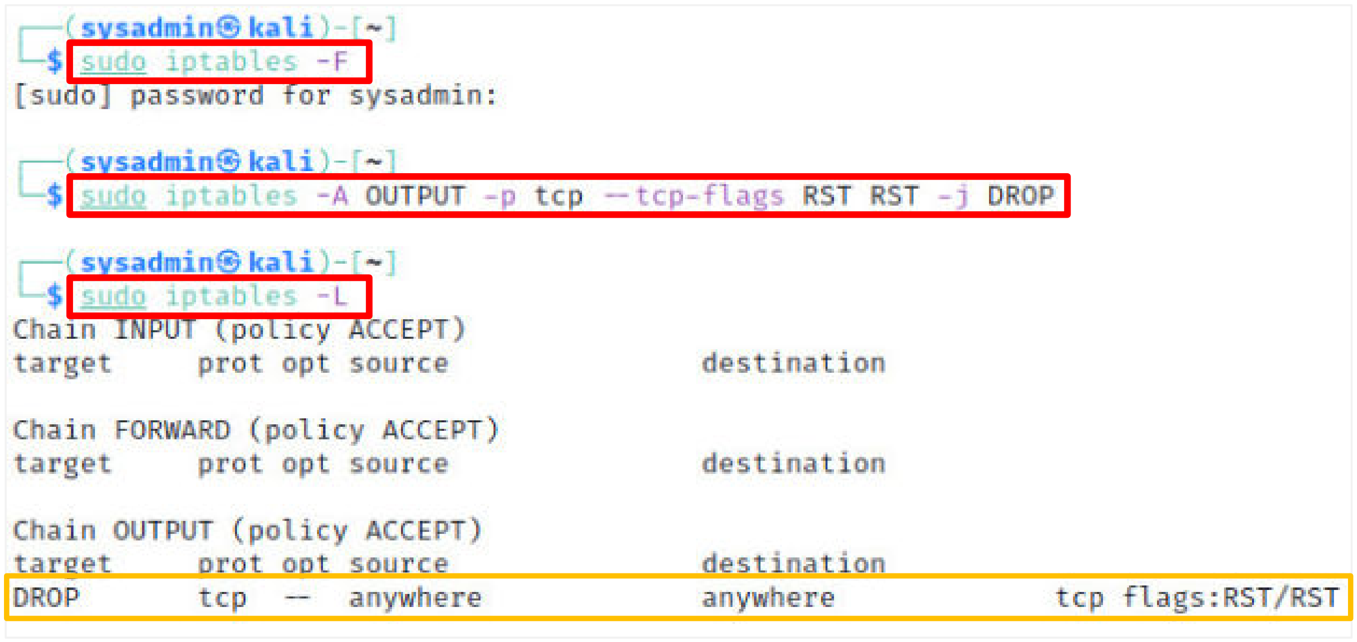
10. At the bottom of the window, in the *Application Transparency* field, change the value to `0`. Finally, click the **OK** button at the bottom of the window.



✓ Enable bi-directional text support
✓ Use box drawing characters contained in the font
Application transparency 0 %

11. *Scapy* will be sending a "raw" **TCP SYN** packet, but the Linux kernel on the *UbuntuSRV* computer will only accept **SYN** packets from kernel routines to open connections. The *Scapy* will send an **RST** packet to cancel the improper connection. To prevent the **RST** from dropping the connection, there needs to be an **iptables** firewall rule. In the terminal, type the following commands. The first command flushes out the old rules, the second command adds a rule to drop all outbound **RST** packets from the *Kali* computer, and the third command lists the firewall rules. If asked for the **[sudo] password for sysadmin,** type: `NDGlabpass123!`

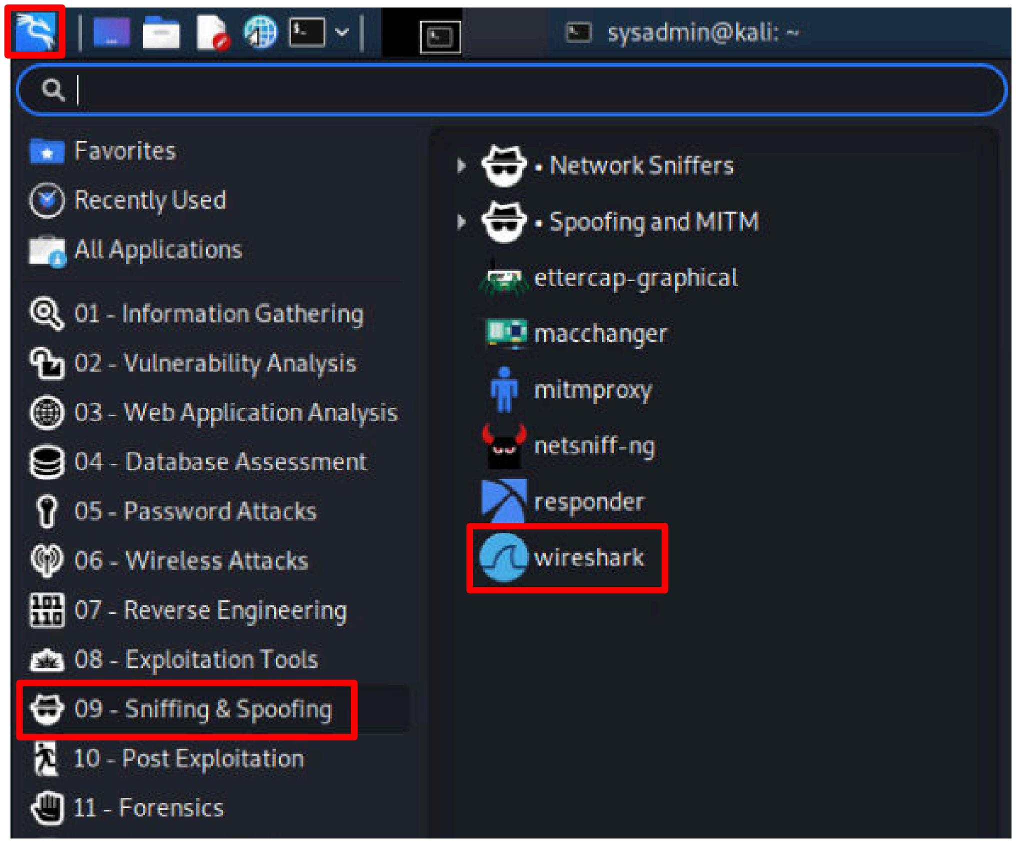
```
sudo iptables -F
sudo iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP
sudo iptables -L
```



```
┌──(sysadmin㉿kali)-[~]
└─$ sudo iptables -F
[sudo] password for sysadmin:

┌──(sysadmin㉿kali)-[~]
└─$ sudo iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP

┌──(sysadmin㉿kali)-[~]
└─$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere             tcp flags:RST/RST
```

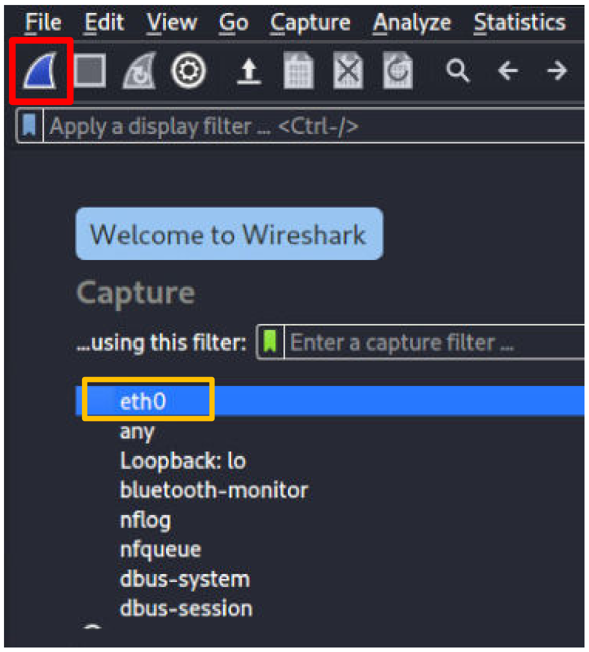
12. Leave the terminal window open and continue to the next step.

13. Open *Wireshark* by navigating to **Kali Start>Sniffing & Spoofing>Wireshark**.



You will want to adjust the size and location of the *Scapy* window and the Wireshark page so that you can easily click between the two.

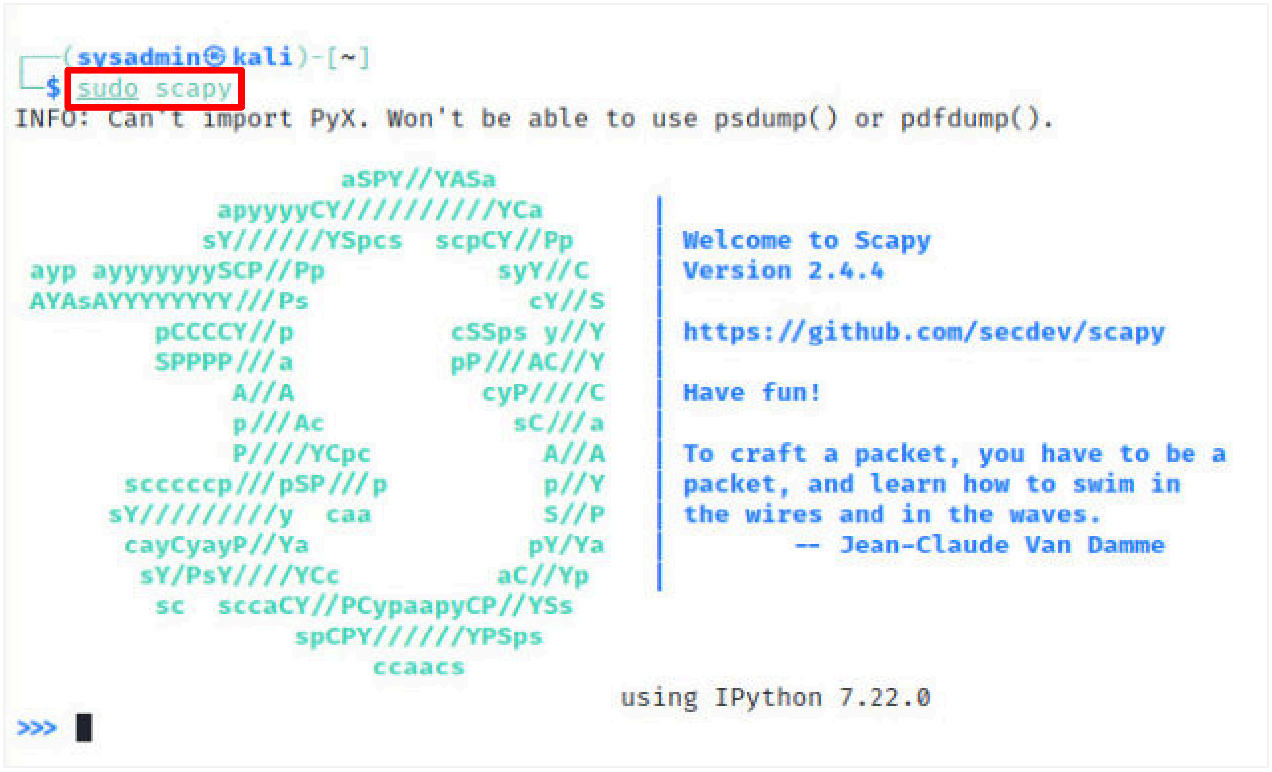
14. In the opening window, make sure that **eth0** is highlighted and click on the **Start Capturing Packets** button in the toolbar.



15. Return to the terminal window.
16. Type the following command to start *Scapy*:

```
sudo scapy
```

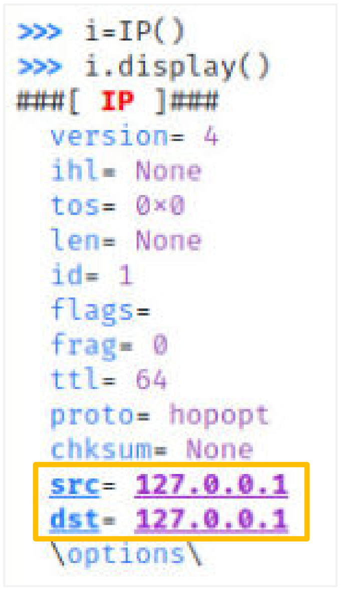
If asked for the **[sudo] password for sysadmin,** type: `NDGlabpass123!`



```
┌──(sysadmin㉿kali)-[~]
└─$ sudo scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.4.4
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | To craft a packet, you have to be a
       scccccp///pSP///p          p//Y   | packet, and learn how to swim in
      sY/////////y  caa           S//P   | the wires and in the waves.
       cayCyayP//Ya              pY/Ya   |        -- Jean-Claude Van Damme
        sY/PsY////YCc          aC//Yp    |
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
                                       using IPython 7.22.0
>>> 
```

17. In this step, you will create an **IP Packet** and change the source and destination **IP addresses**. In the *Scapy* window, at the **>>>** prompt, execute these commands. The first command will create an **IP Packet Object**, and the second command will display the packet with all of the default attributes.

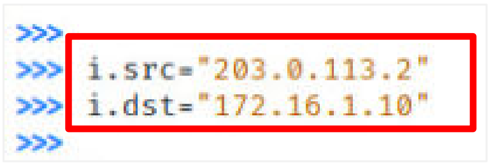
```
i=IP()
i.display()
```



```
>>> i=IP()
>>> i.display()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= hopopt
  chksum= None
  src= 127.0.0.1
  dst= 127.0.0.1
  \options\
```

18. The source (**src**) and destination (**dst**) IP addresses default to **127.0.0.1** (loopback address). These need to be changed to the IP addresses of **Kali (203.0.113.2)**, the source and **UbuntuSRV (172.16.1.10)**, the destination. Type these commands to change the IP addresses and to display the updated packet attributes:

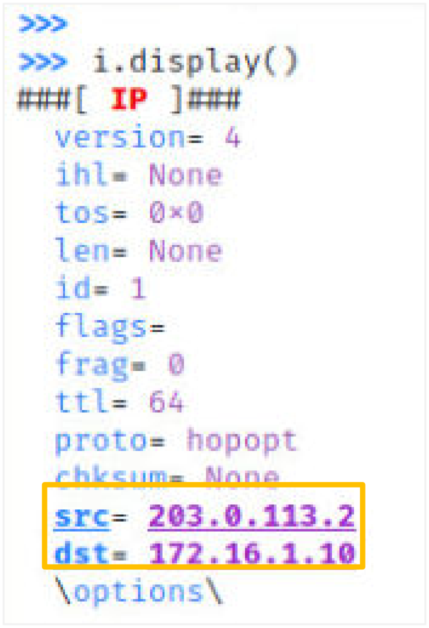
```
i.src=”203.0.113.2”
i.dst=”172.16.1.10”
```



```
>>>
>>> i.src="203.0.113.2"
>>> i.dst="172.16.1.10"
>>>
```

19. To check the **IP** packet header, type the following command:

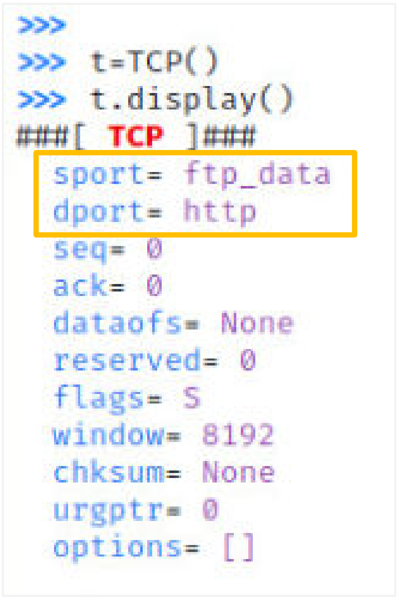
```
i.display()
```



```
>>>
>>> i.display()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= hopopt
  chksum= None
  src= 203.0.113.2
  dst= 172.16.1.10
  \options\
```

20. In this step, you will create a **TCP Packet** and change the source and destination **TCP Ports**. In the *Scapy* window, at the **>>>** prompt, execute these commands. The first command will create a **TCP Packet Object**, and the second command will display the packet with all of the default attributes.

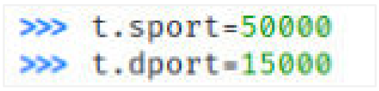
```
t=TCP()
t.display()
```



```
>>>
>>> t=TCP()
>>> t.display()
###[ TCP ]###
  sport= ftp_data
  dport= http
  seq= 0
  ack= 0
  dataofs= None
  reserved= 0
  flags= S
  window= 8192
  chksum= None
  urgptr= 0
  options= []
```

21. The source port (**sport**) on the **Kali** computer needs to be changed to a port in the private port range, and the destination port (**dport**) to the *UbuntuSRV* computer needs to be changed to the open target port (**15000**). Type these commands to change the port addresses and to display the updated packet attributes:

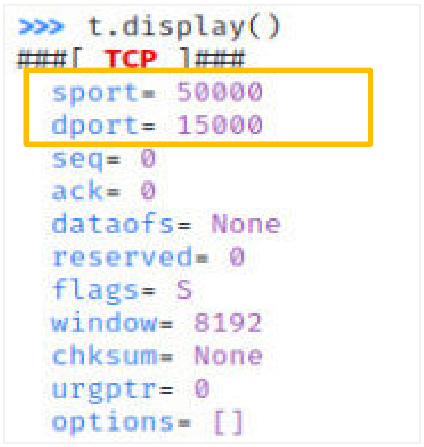
```
t.sport=50000
t.dport=15000
```



```
>>> t.sport=50000
>>> t.dport=15000
```

22. To check the **TCP** packet header, type the following command:

```
t.display()
```

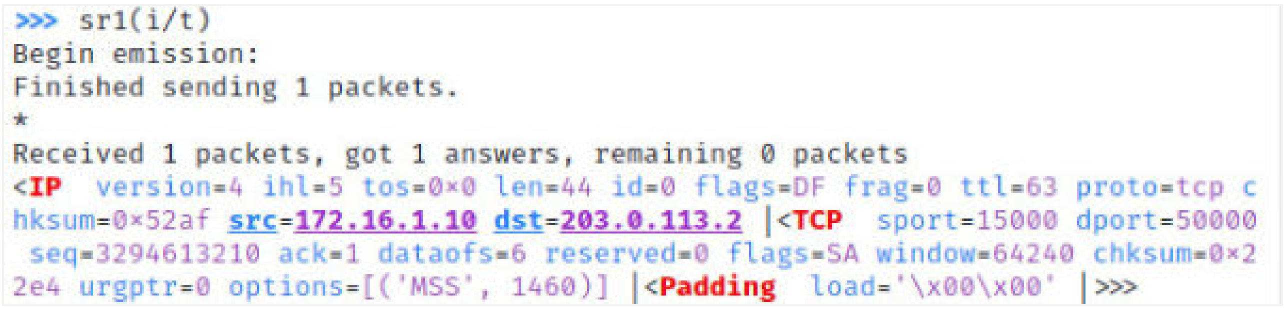


```
>>> t.display()
###[ TCP ]###
  sport= 50000
  dport= 15000
  seq= 0
  ack= 0
  dataofs= None
  reserved= 0
  flags= S
  window= 8192
  chksum= None
  urgptr= 0
  options= []
```

23. Type the following command to send a *SYN* packet with the crafted IP (i) and TCP (t) headers from the *Scapy* to the *UbuntuSRV computer*:

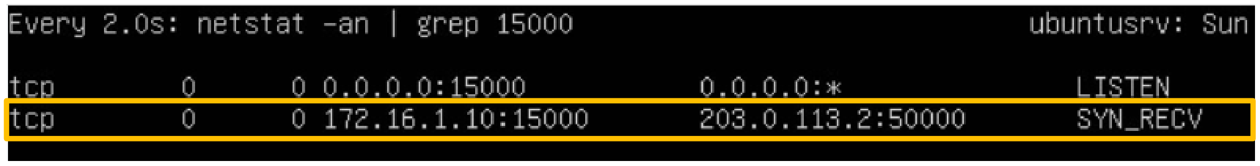
```
sr1(i/t)
```

24. The command will send the **IP (Layer 3)** and **TCP (Layer 4)** packets that were just crafted to the *UbuntuSRV* computer. When *UbuntuSRV* receives the packet, it will send back its response packet.



```
>>> sr1(i/t)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
<IP  version=4 ihl=5 tos=0x0 len=44 id=0 flags=DF frag=0 ttl=63 proto=tcp c
hksum=0x52af src=172.16.1.10 dst=203.0.113.2 |<TCP  sport=15000 dport=50000
 seq=3294613210 ack=1 dataofs=6 reserved=0 flags=SA window=64240 chksum=0x2
2e4 urgptr=0 options=[('MSS', 1460)] |<Padding  load='\x00\x00' |>>>
```

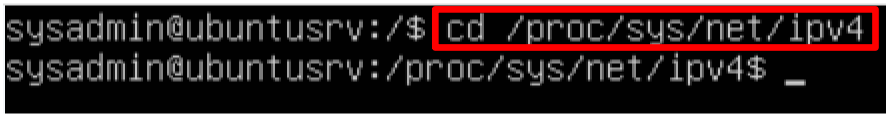
25. Return to the **UbuntuSRV** computer, and you will see that the connection is in the **SYN_RECV** state and is waiting for the **ACK** to complete the connection.


```
Every 2.0s: netstat -an | grep 15000                                  ubuntusrv: Sun

tcp        0      0 0.0.0.0:15000           0.0.0.0:*               LISTEN
tcp        0      0 172.16.1.10:15000       203.0.113.2:50000       SYN_RECV
```

This shows the *UbuntuSRV* has received the **[SYN]** packet from the *Kali* computer and replied with a **[SYN,ACK]** and is now waiting for the *Kali* computer to complete the three-way handshake and send the **[ACK].** Wait for 60 seconds, and you should see the connection time out and be removed from the *netstat* list. The default setting for *Ubuntu* is to wait 60 seconds before giving up on the connection. A hacker would have to change the [**ACK**] packet and send it to the *UbuntuSRV* computer within the 60-second limit. For this lab, it will be much easier for you to change the **TCP_SYNACK_RETRIES** value on the *UbuntuSRV* computer giving you more time.

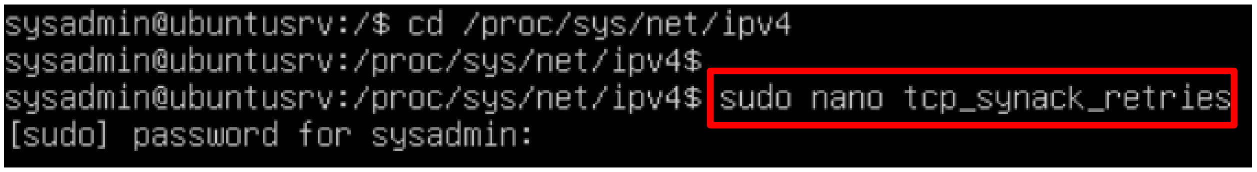
26. Press **ALT+F3** to open another terminal session and log in to the session with the username `sysadmin` and the password: `NDGlabpass123!`
27. Change the directory by typing the following command:

```
cd /proc/sys/net/ipv4
```

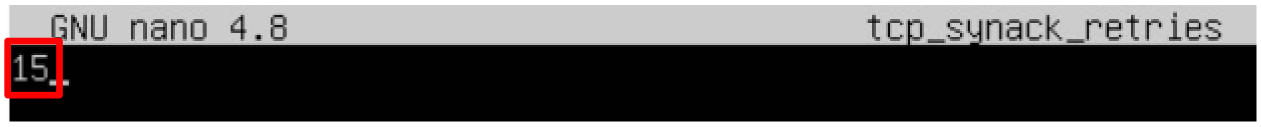

```
sysadmin@ubuntusrv:/$ cd /proc/sys/net/ipv4
sysadmin@ubuntusrv:/proc/sys/net/ipv4$ _
```

28. Using *nano*, edit the **tcp_synack_retries** file using the following command. If asked for the **[sudo] password for sysadmin** use: `NDGlabpass123!`

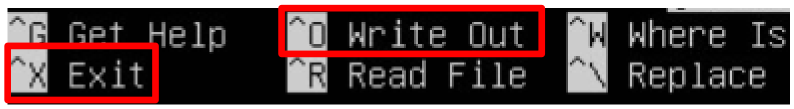
```
sudo nano tcp_synack_retries
```


```
sysadmin@ubuntusrv:/$ cd /proc/sys/net/ipv4
sysadmin@ubuntusrv:/proc/sys/net/ipv4$
sysadmin@ubuntusrv:/proc/sys/net/ipv4$ sudo nano tcp_synack_retries
[sudo] password for sysadmin:
```

29. When the editor opens, change the 5 to 15, which will give you three minutes.

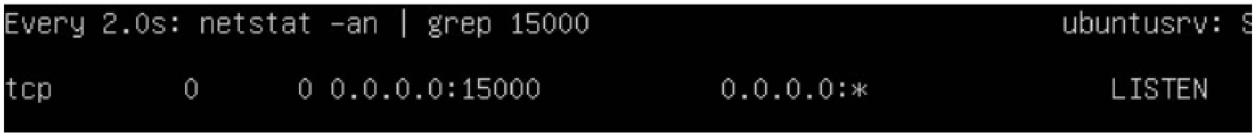

```
  GNU nano 4.8                                     tcp_synack_retries
15_
```

30. Save the changes to the file by pressing **CTRL+O**, press **Enter** to save it using the same file name, and exit the editor by pressing **CTRL+X.**



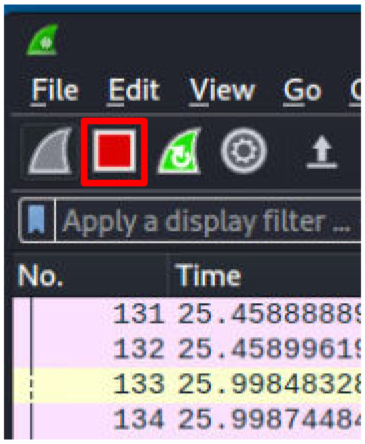
```
^G Get Help    ^O Write Out    ^W Where Is
^X Exit        ^R Read File    ^\ Replace
```

31. Press **ALT+F2** to return to the session where the *netstat* is monitoring the connections.



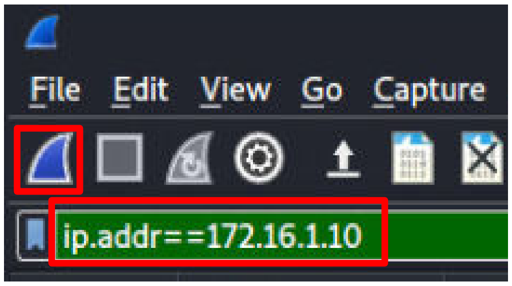
```
Every 2.0s: netstat -an | grep 15000                         ubuntusrv: S

tcp        0      0 0.0.0.0:15000           0.0.0.0:*               LISTEN
```

32. Change the focus to the **Kali** computer and switch to *Wireshark* window.
33. Stop the scan by clicking the **Stop Capturing Packet** button.

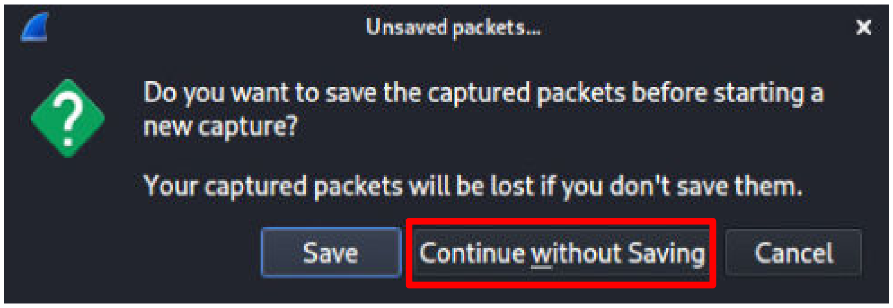


```
No.   Time
131 25.45888889
132 25.45899619
133 25.99848328
134 25.99874484
```

34. Go to the **FILTER** box and type in the filter `ip.addr==172.16.1.10` and press **Enter**. Then click on the **Start Capturing Packets** button in the toolbar.



35. When the *Unsaved Packets* window pops up, click on **Continue Without Saving.**


Unsaved packets...

Do you want to save the captured packets before starting a new capture?

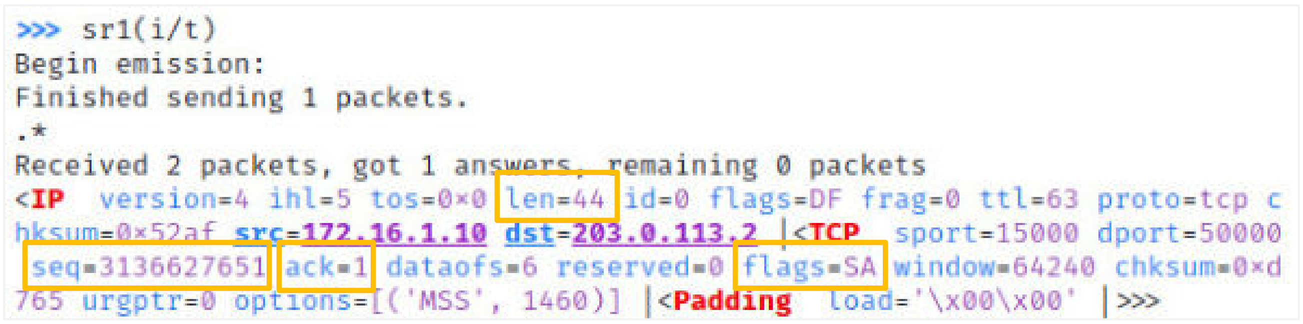
Your captured packets will be lost if you don't save them.

There are packets being captured, but since we have set a filter to only show packets with a source or destination address of 172.16.1.10 you should not see any packets … yet.

36. Switch to the terminal window running *Scapy* and type the following command to resend the **SYN** packet from the *Kali* computer to the *UbuntuSRV* computer:

```
sr1(i/t)
```

37. You will see the reply packet from the *UbuntuSRV*. Make a note of the **SEQ** number. This is the number that will need to **add 1** to for the return **ACK**.


```
>>> sr1(i/t)
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
<IP  version=4 ihl=5 tos=0x0 len=44 id=0 flags=DF frag=0 ttl=63 proto=tcp c
hksum=0x52af src=172.16.1.10 dst=203.0.113.2 |<TCP  sport=15000 dport=50000
 seq=3136627651 ack=1 dataofs=6 reserved=0 flags=SA window=64240 chksum=0xd
765 urgptr=0 options=[('MSS', 1460)] |<Padding  load='\x00\x00' |>>>
```

Examine the reply in *Scapy*, above. Note these items:

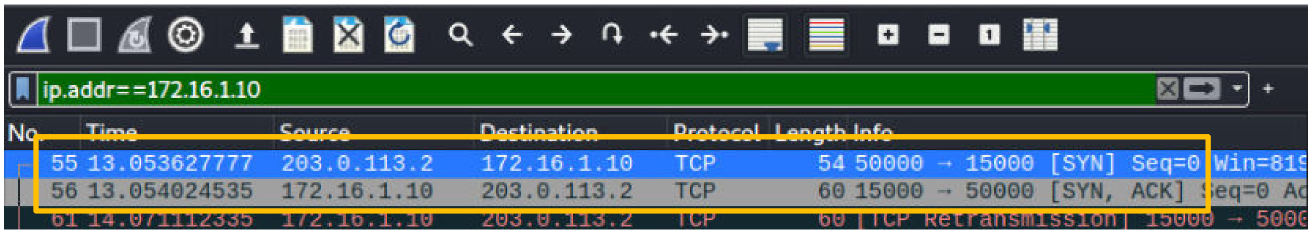
- **len** is the length of the packet being sent from the Kali computer. It will be used by the *UbuntuSRV* computer to calculate the ACK number.

- **seq** is the sequence number which is a 32-bit number used to keep track of the amount of data being sent, in this case, from the Kail computer to *UbuntuSRV* computer. This starting number is random, and subsequent SEQ numbers will be the initial SEQ number plus the ACK number that is returned. In the example, the initial SEQ number is **3136627651** (which will not be the same when you perform the lab). Make a note of the number on your session.

- **ack** is the acknowledgment number, also a 32-bit number, which is returned from the connected host, in this case, from the *UbuntuSRV* computer back to the *Kali* computer. The initial ACK number that will be returned is the LEN of the received packet plus 1 (the initial ACK from the *Kali* computer is set to 1 since there has not been an acknowledgment packet received from the *UbuntuSRV* computer). Subsequent ACK numbers will be the current ACK number plus the length of the received packet.

- **TCP flags** show the particular state of the connection and can provide useful information for troubleshooting or handling control of a connection. The **SA** flag shows a SYN and an ACK. The SYN flag is used to synchronize the initial sequence number and is only present in the first packet sent from the *Kali* computer. The ACK flag acknowledges packets that were received.

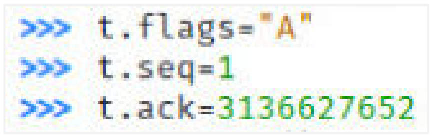
38. Switch to the *Wireshark* window, and you will see the **[SYN]** and **[SYN, ACK]** packets and wait for the *Scapy*-crafted *[ACK].*



```
ip.addr==172.16.1.10
No.  Time          Source        Destination   Protocol Length Info
55 13.053627777  203.0.113.2   172.16.1.10   TCP      54 50000 → 15000 [SYN] Seq=0 Win=819
56 13.054024535  172.16.1.10   203.0.113.2   TCP      60 15000 → 50000 [SYN, ACK] Seq=0 Ac
61 14.071112335  172.16.1.10   203.0.113.2   TCP      60 [TCP Retransmission] 15000 → 5000
```

39. Switch to the terminal window running *Scapy*. At the **>>>** prompt, make the following changes to the crafted TCP packet:

```
t.flags=”A”
t.seq=1
t.ack=<< take the seq number from the packet that was received and add 1 >>
```

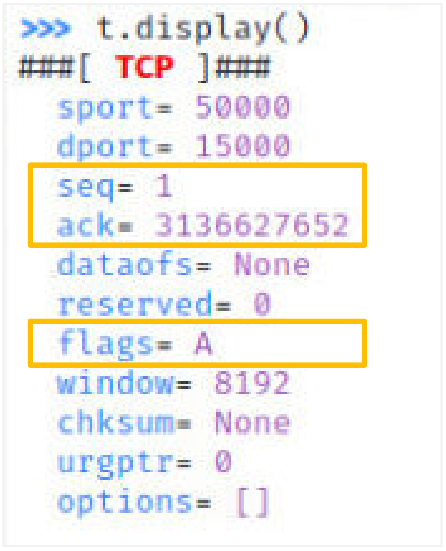


```
>>> t.flags="A"
>>> t.seq=1
>>> t.ack=3136627652
```

Where you are changing the flags field with **A** for **ack** packet, set the **seq** to **1,** which will be the next sequence number, relative to the initial sequence number **0,** and the **ack,** which is the sequence number returned in the [**SYN,ACK**] from the *UbuntuSRV* plus 1.

40. Check the crafted **TCP** packet by typing the following:

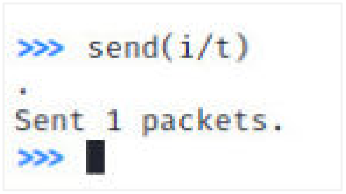
```
t.display()
```



```
>>> t.display()
###[ TCP ]###
  sport= 50000
  dport= 15000
  seq= 1
  ack= 3136627652
  dataofs= None
  reserved= 0
  flags= A
  window= 8192
  chksum= None
  urgptr= 0
  options= []
```

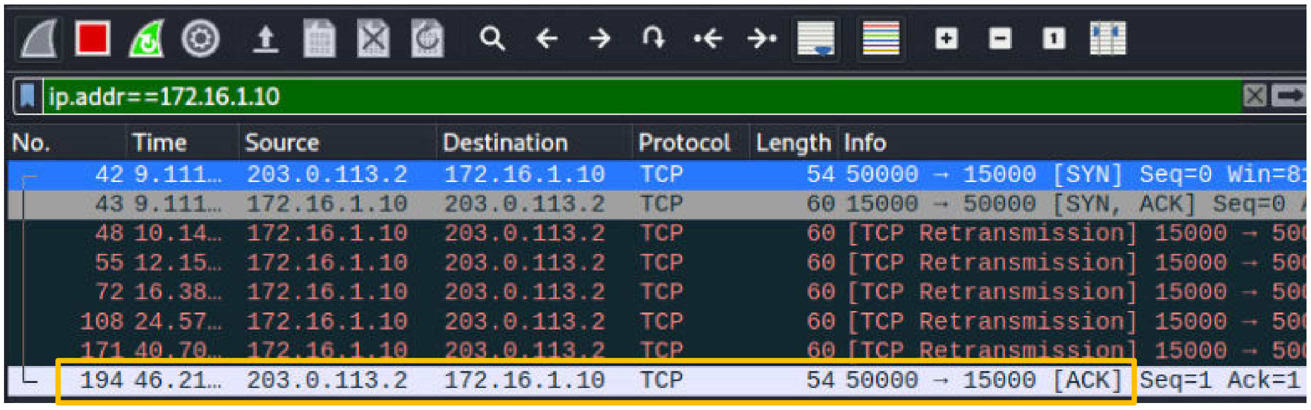
41. Send the crafted **TCP** packet back to the *UbuntuSRV* computer by typing the following command:

```
send(i/t)
```



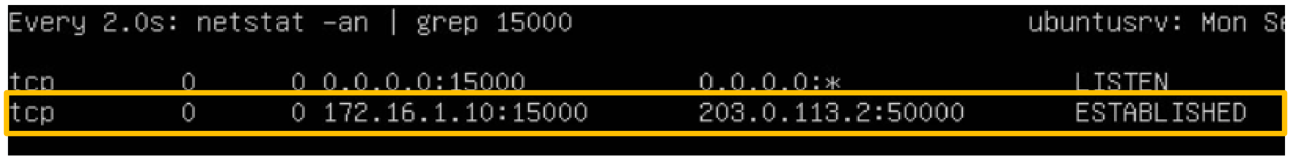
```
>>> send(i/t)
.
Sent 1 packets.
>>> 
```

42. Switch to the *Wireshark* window, and you see the **[ACK]** packet has been sent as a response to the **[SYN, ACK]** packet.



```
ip.addr==172.16.1.10
No.  Time      Source        Destination   Protocol Length Info
 42  9.111…    203.0.113.2   172.16.1.10   TCP      54  50000 → 15000 [SYN] Seq=0 Win=8
 43  9.111…    172.16.1.10   203.0.113.2   TCP      60  15000 → 50000 [SYN, ACK] Seq=0 
 48  10.14…    172.16.1.10   203.0.113.2   TCP      60  [TCP Retransmission] 15000 → 500
 55  12.15…    172.16.1.10   203.0.113.2   TCP      60  [TCP Retransmission] 15000 → 500
 72  16.38…    172.16.1.10   203.0.113.2   TCP      60  [TCP Retransmission] 15000 → 500
108  24.57…    172.16.1.10   203.0.113.2   TCP      60  [TCP Retransmission] 15000 → 500
171  40.70…    172.16.1.10   203.0.113.2   TCP      60  [TCP Retransmission] 15000 → 500
194  46.21…    203.0.113.2   172.16.1.10   TCP      54  50000 → 15000 [ACK] Seq=1 Ack=1
```

43. Change focus to the **UbuntuSRV** computer and press **ALT+2** if the *netstat* listening session is not shown.



```
Every 2.0s: netstat -an | grep 15000                                ubuntusrv: Mon Se

tcp        0      0 0.0.0.0:15000          0.0.0.0:*               LISTEN
tcp        0      0 172.16.1.10:15000      203.0.113.2:50000       ESTABLISHED
```

Notice that the TCP connection now shows **ESTABLISHED**. You now have a backdoor into the *UbuntuSRV* computer.

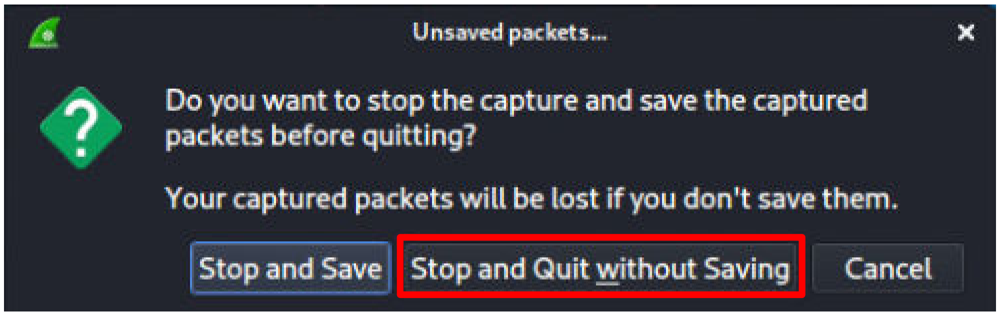
44. Press **+C** to end the *netstat* watching session and press **ALT+F1** to go back to the *netstat* listening session, and press **CTRL+C** to end the process.
45. Return to the **Kali** computer and exit *Scapy* by typing the command:

```
quit
```

… and close the terminal window by typing the command:

```
exit
```

46. Close the *Wireshark* window by clicking the **X** in the upper-right corner and when the *Unsaved Packets* window pops up, click on **Stop and Quit Without Saving.**



Unsaved packets...

Do you want to stop the capture and save the captured packets before quitting?

Your captured packets will be lost if you don't save them.

This task was inspired by a lab from Dr. Sam Bowne
Professor, City College of San Francisco.

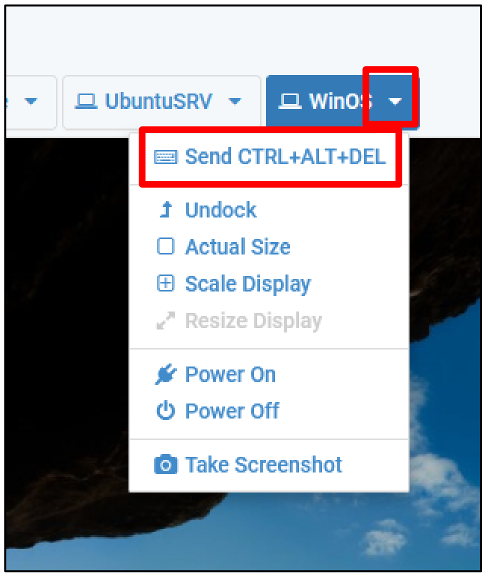
3 Analyzing Encrypted Traffic Using Wireshark

Encryption at the Presentation Layer (Layer 6) of the OSI model is handled by both SSL, which has been largely deprecated, and **TLS,** which is the current standard. Normally you would think that packets that are encrypted cannot be viewed and analyzed, but Wireshark does have a method to assist in decrypting packets using a Pre-Master Secret Key.

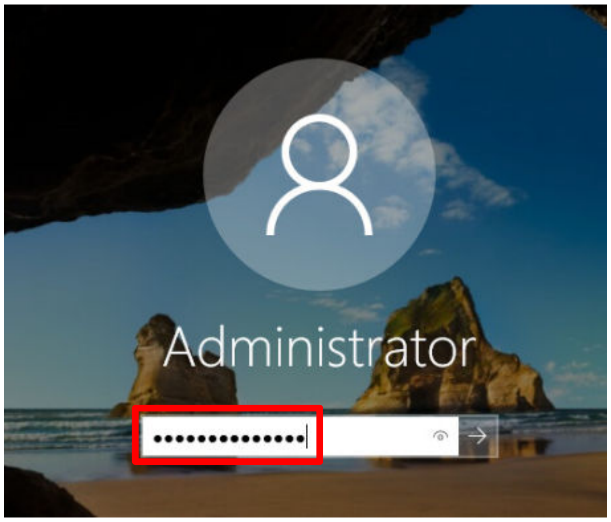
For this part of the lab, you will be using the *WinOS* computer.

3.1 Setting the Pre-Master Secret Key Path Environment Variable

1. Change focus to the **WinOS** computer.
2. Bring up the login window by sending a Ctrl + Alt+ Delete. To do this, click the **WinOS** dropdown menu and click **Send CTRL+ALT+DEL**.

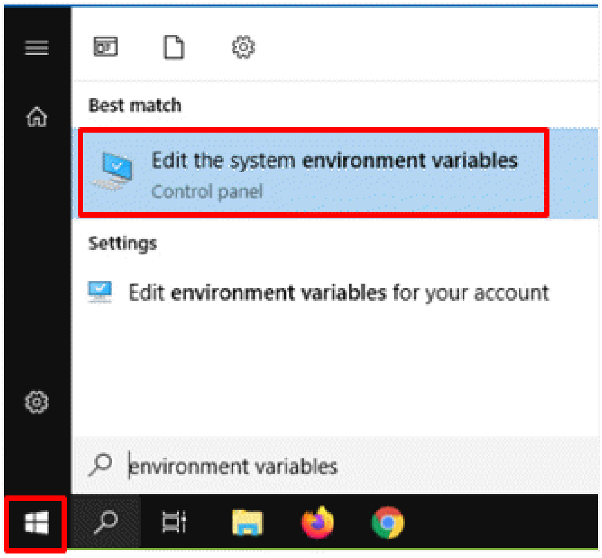


3. Log in as *Administrator* using the password: `NDGlabpass123!`

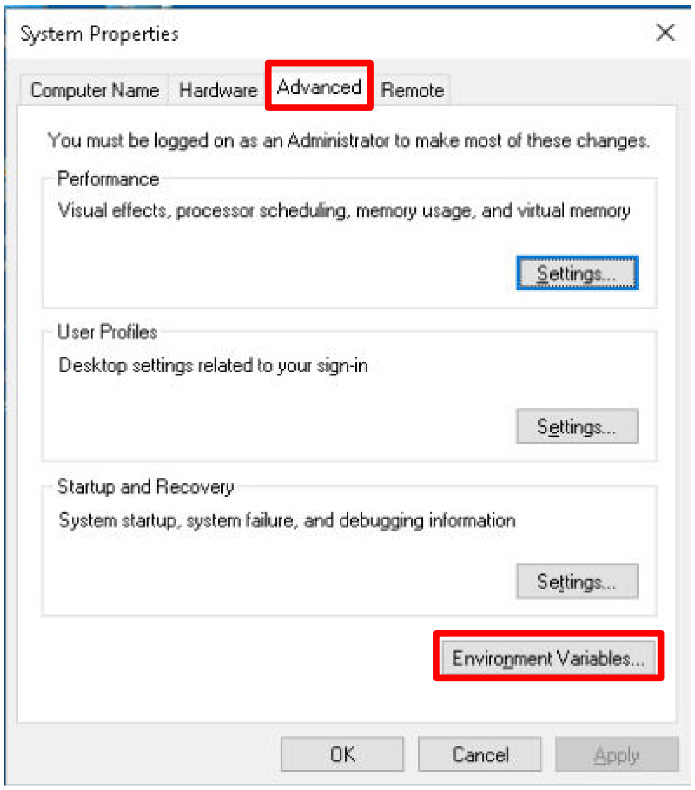


The first thing you need to do is to set a *Windows Environment Variable* to hold the path for the *Pre-Master Secret Key* that will be captured from a secure website.

4. Click on the **Windows Start** button, type in `environment variables` and click on **Edit the system environment variables** from the *Best Match* search result.


Best match
Edit the system environment variables
Control panel
Settings
Edit environment variables for your account
environment variables

5. Click on the **Advanced** tab, then click the **Environment Variables** button:


System Properties

Computer Name | Hardware | Advanced | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

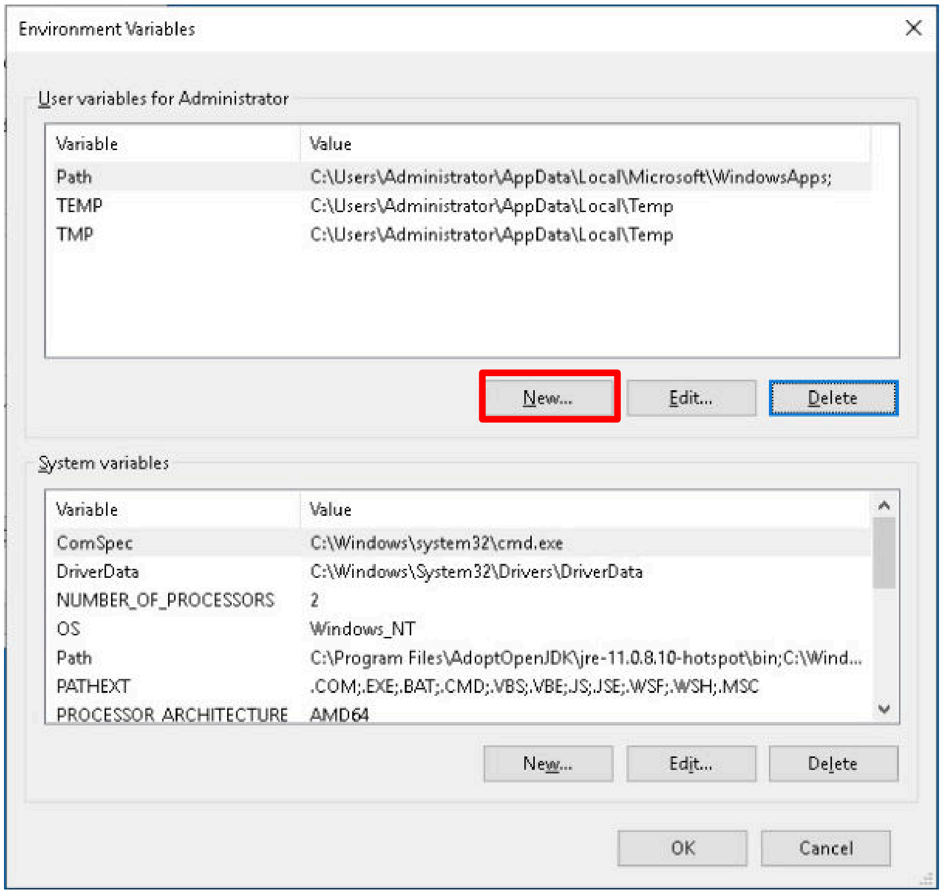
User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK Cancel Apply

6. In the *Environment Variables* window, click on the **New** button under *User Variables* for *Administrator*.



Environment Variables

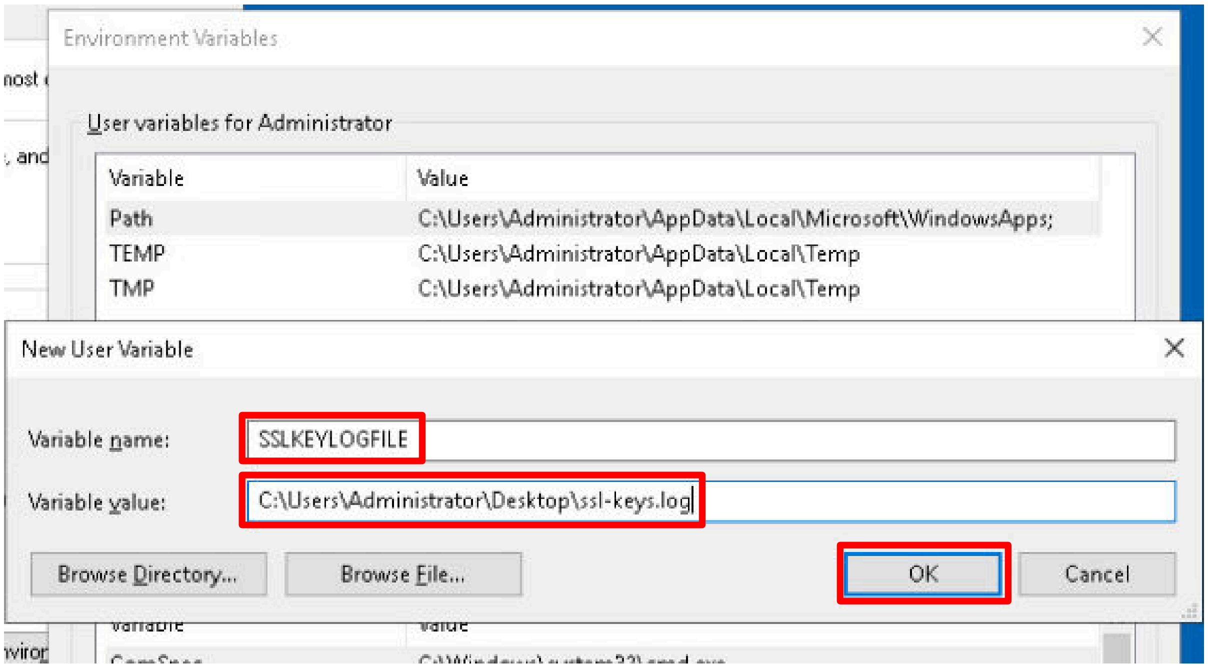
User variables for Administrator

Variable	Value
Path	C:\Users\Administrator\AppData\Local\Microsoft\WindowsApps;
TEMP	C:\Users\Administrator\AppData\Local\Temp
TMP	C:\Users\Administrator\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	2
OS	Windows_NT
Path	C:\Program Files\AdoptOpenJDK\jre-11.0.8.10-hotspot\bin;C:\Wind...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

7. In the *New User Variables* window, type `SSLKEYLOGFILE` for the *Variable name* and type `C:\Users\Administrator\Desktop\ssl-keys.log` for the *Variable Value* and click the **OK** button.



New User Variable

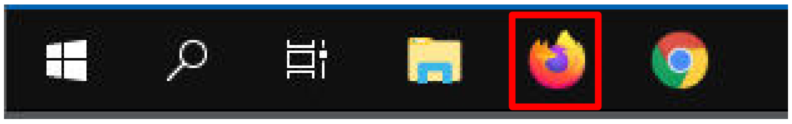
Variable name: SSLKEYLOGFILE

Variable value: C:\Users\Administrator\Desktop\ssl-keys.log

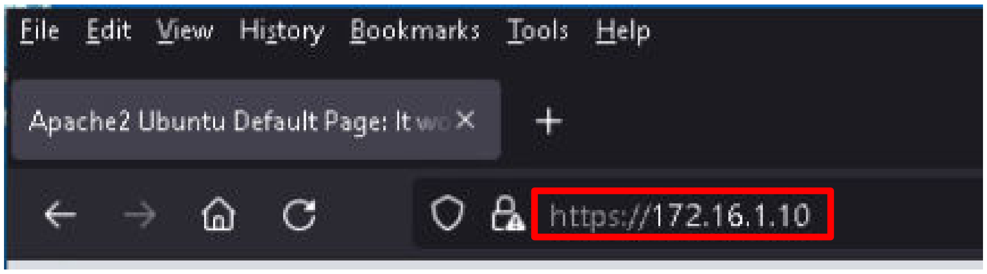
8. Click the **OK** button on the Environment Variables window to save the settings.
9. Close the System Properties window by clicking the **OK** button.
10. Keep the **WinOS** computer open for the next task.

3.2 Capture a Pre-Master Shared Key from a Secure Website

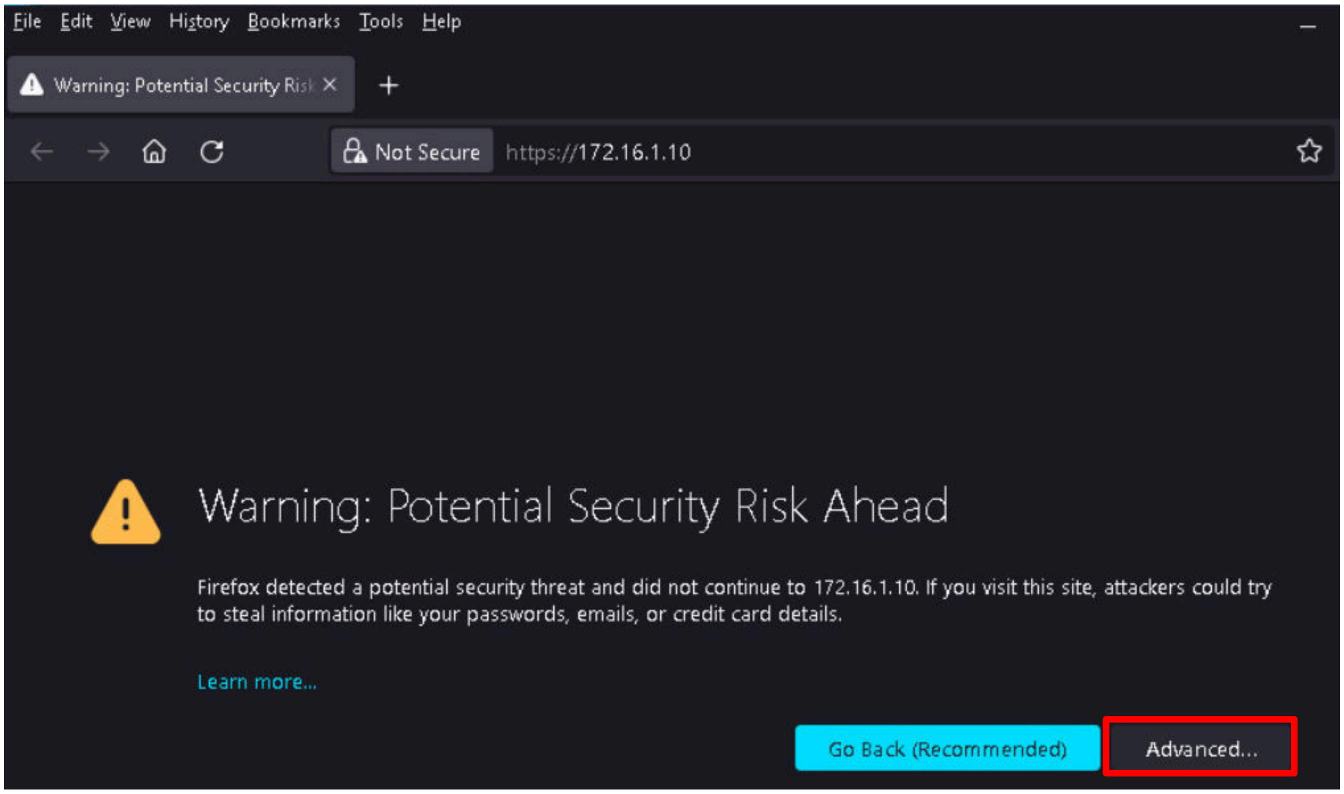
1. On the *WinOS* computer, click on the **Firefox** browser icon in the taskbar to open a web browser.



2. You need to capture the **Pre-Master Share Key** and store it in the **ssl-keys.log** file. In the browser address bar, type `https://172.16.1.10`.


File Edit View History Bookmarks Tools Help
Apache2 Ubuntu Default Page: It wo
https://172.16.1.10

3. On the *Warning: Potential Security Risk Ahead*, click the **Advanced** button.

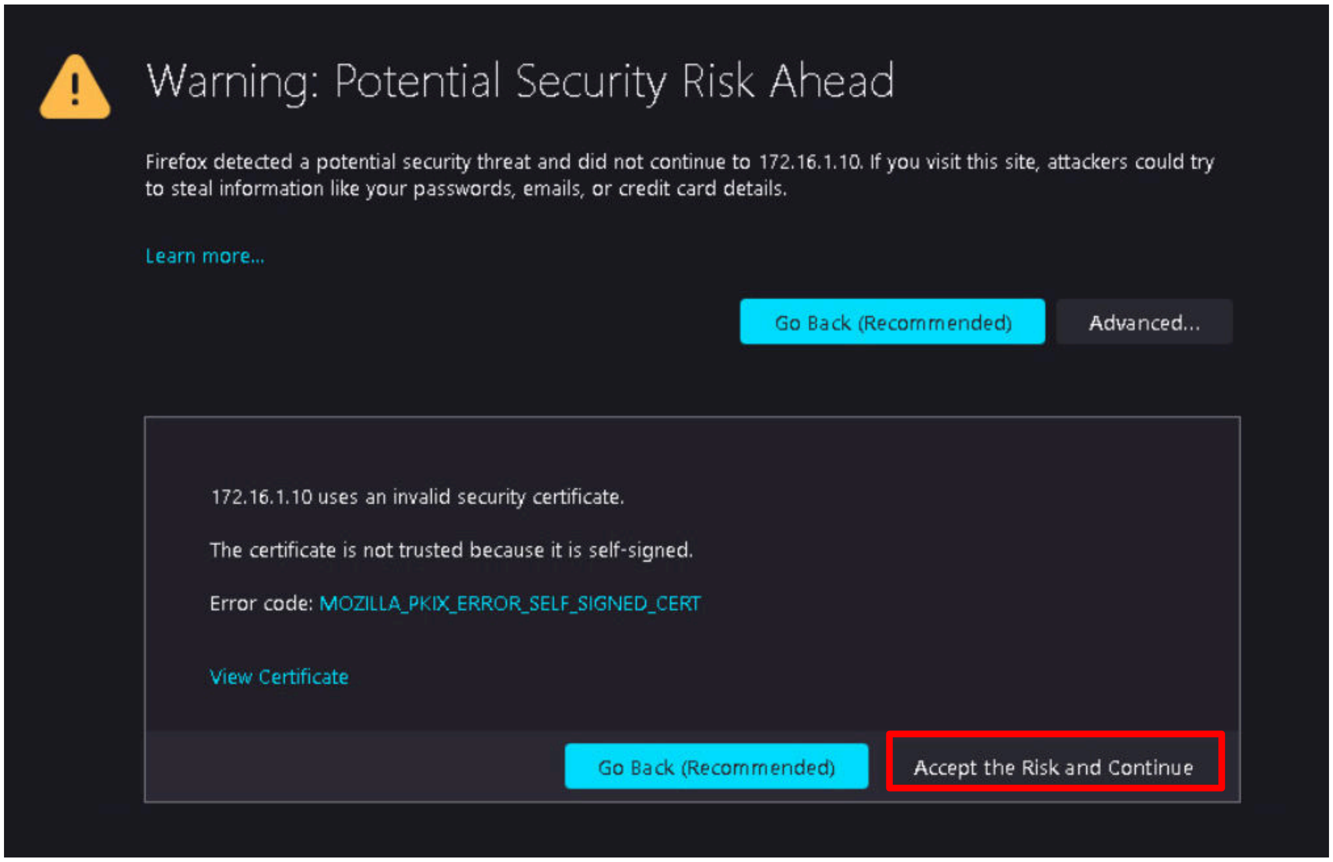

File Edit View History Bookmarks Tools Help
Warning: Potential Security Risk
Not Secure
https://172.16.1.10

Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to 172.16.1.10. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

Learn more...
Go Back (Recommended)
Advanced...

4. Scroll to the bottom of the window and click the **Accept the Risk and Continue** button.



Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to 172.16.1.10. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

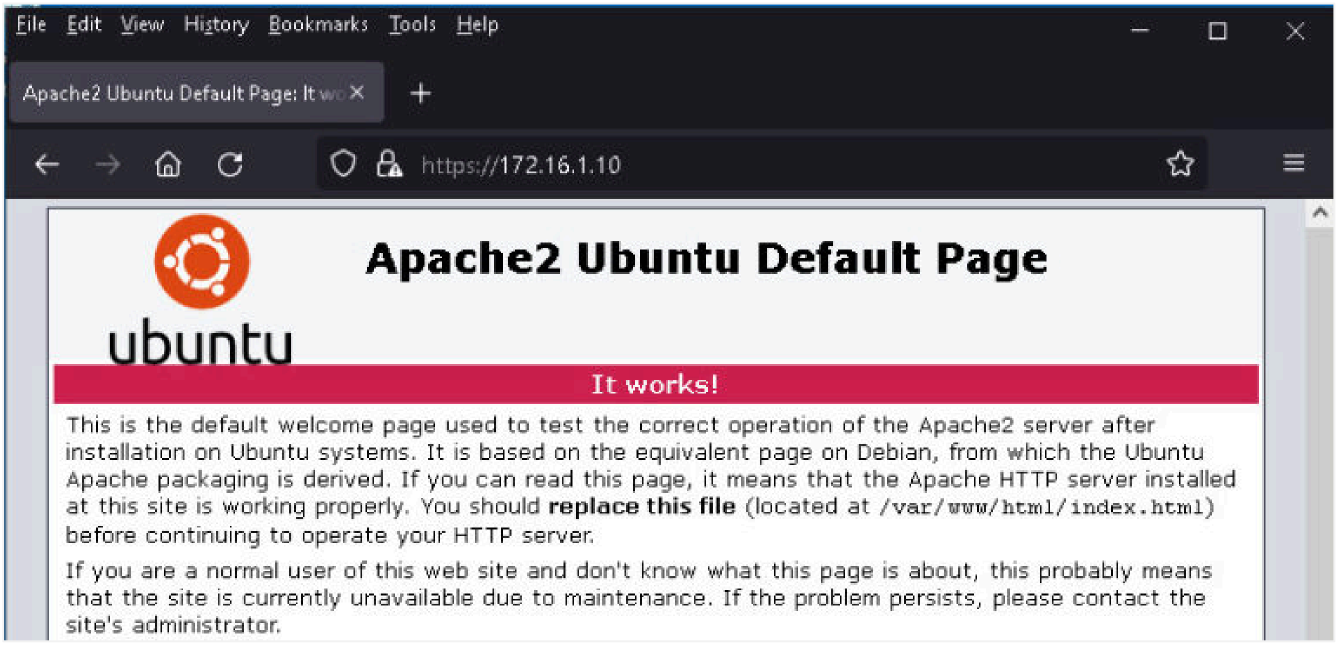
Learn more...

172.16.1.10 uses an invalid security certificate.

The certificate is not trusted because it is self-signed.

Error code: MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT

You will see the *Apache2 Ubuntu Default* page.



File Edit View History Bookmarks Tools Help
Apache2 Ubuntu Default Page: It wo
https://172.16.1.10
ubuntu

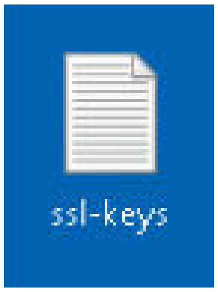
Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

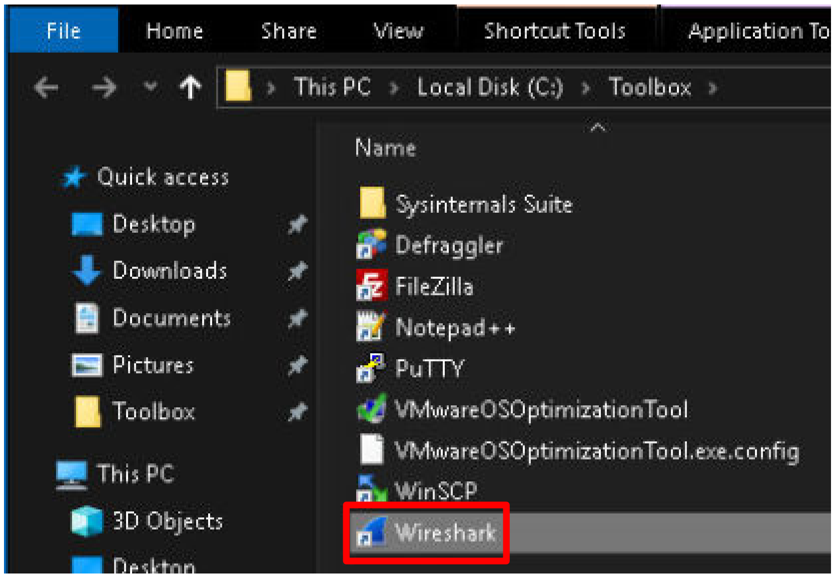
You will notice a file named **ssl-keys** will have popped up on your desktop (the extension is hidden by default).



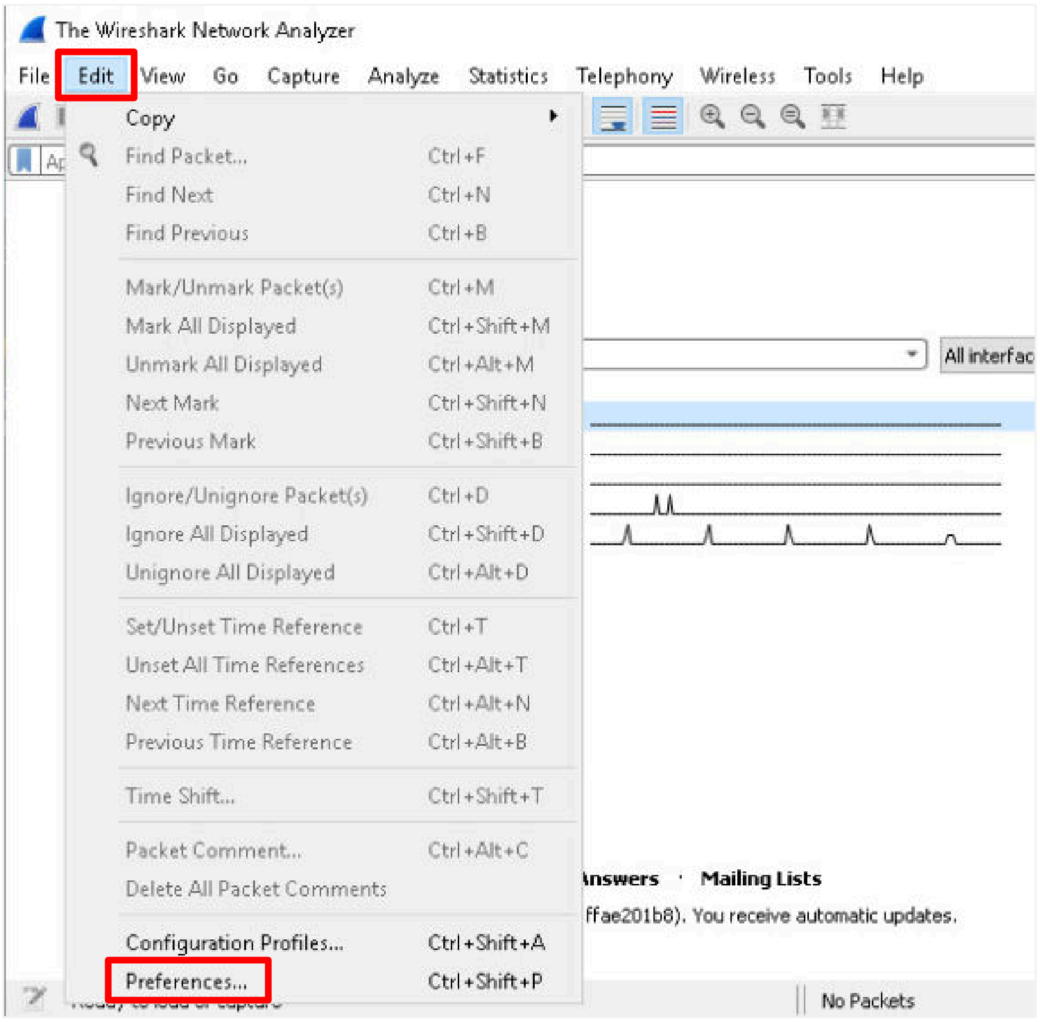
5. Close the **Firefox** browser.

3.3 Configure Wireshark

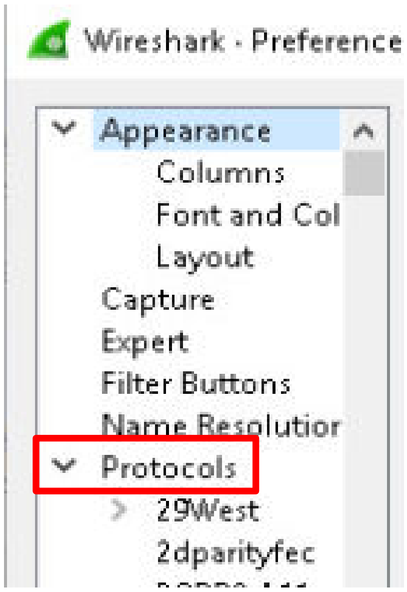
1. Open the **Toolbox** folder on the desktop and double-click on the **Wireshark** icon.



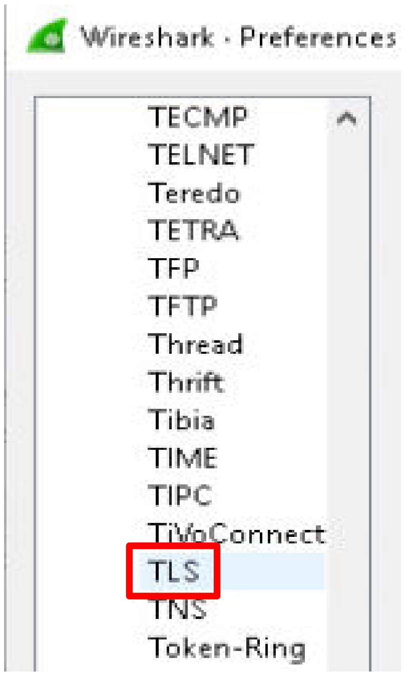
2. When the *Wireshark* window opens, click on **Edit** in the menu bar, then click **Preferences.**


The Wireshark Network Analyzer
File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Wireless
Tools
Help
Copy
Find Packet... Ctrl+F
Find Next Ctrl+N
Find Previous Ctrl+B
Mark/Unmark Packet(s) Ctrl+M
Mark All Displayed Ctrl+Shift+M
Unmark All Displayed Ctrl+Alt+M
Next Mark Ctrl+Shift+N
Previous Mark Ctrl+Shift+B
Ignore/Unignore Packet(s) Ctrl+D
Ignore All Displayed Ctrl+Shift+D
Unignore All Displayed Ctrl+Alt+D
Set/Unset Time Reference Ctrl+T
Unset All Time References Ctrl+Alt+T
Next Time Reference Ctrl+Alt+N
Previous Time Reference Ctrl+Alt+B
Time Shift... Ctrl+Shift+T
Packet Comment... Ctrl+Alt+C
Delete All Packet Comments
Configuration Profiles... Ctrl+Shift+A
Preferences... Ctrl+Shift+P
All interfac
nswers · Mailing Lists
ffae201b8). You receive automatic updates.
No Packets

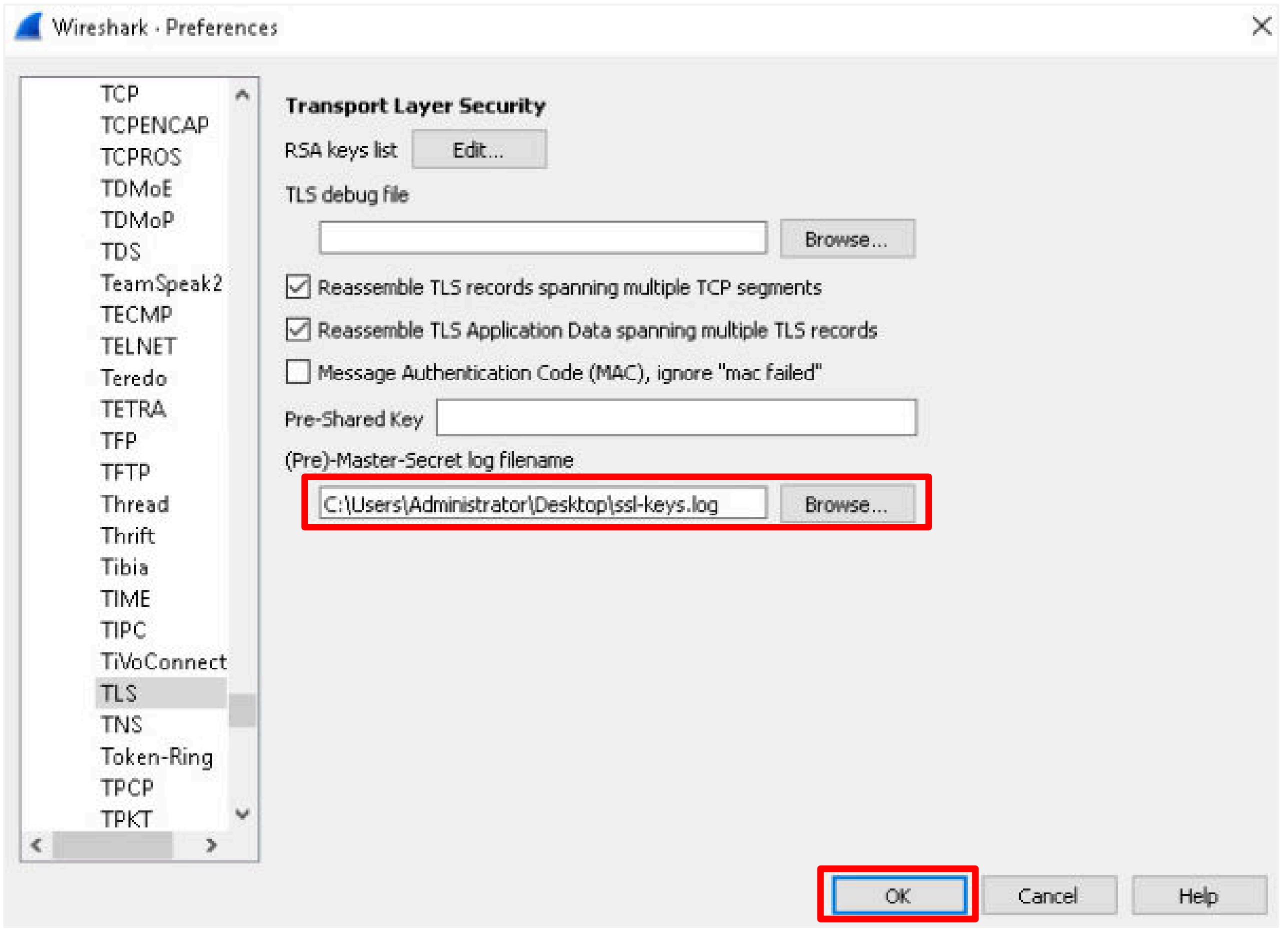
3. Expand the **Protocols** inventory list.



Then, scroll down and click on **TLS.**



4. In the *Preferences* window, under *(Pre)-Master-Secret log filename*, click the **Browse** button and go to the **Desktop** folder and click on the **ssl-keys.log** file. Click **OK** to close the *Preferences* window.



Transport Layer Security

RSA keys list

TLS debug file

[✓] Reassemble TLS records spanning multiple TCP segments

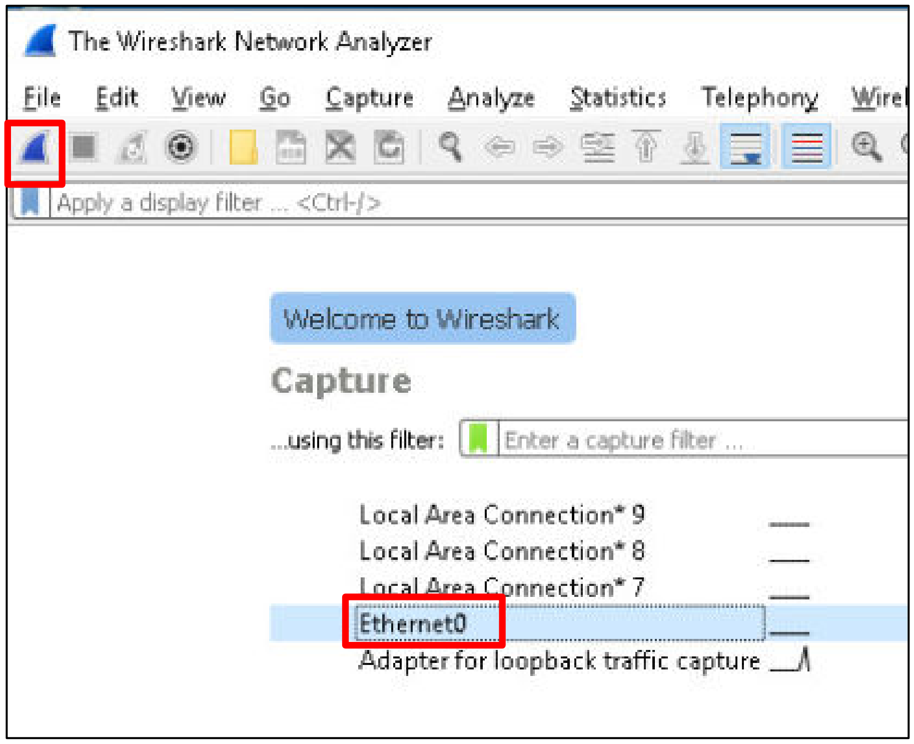
[✓] Reassemble TLS Application Data spanning multiple TLS records

[ ] Message Authentication Code (MAC), ignore "mac failed"

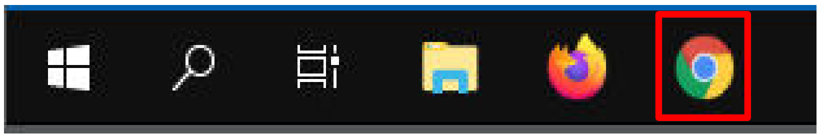
Pre-Shared Key

(Pre)-Master-Secret log filename: C:\Users\Administrator\Desktop\ssl-keys.log

5. In the *Wireshark* window, click on **Ethernet0** and then click on the **Start Capturing Packets** button.


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wirel
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
Local Area Connection* 9
Local Area Connection* 8
Local Area Connection* 7
Ethernet0
Adapter for loopback traffic capture

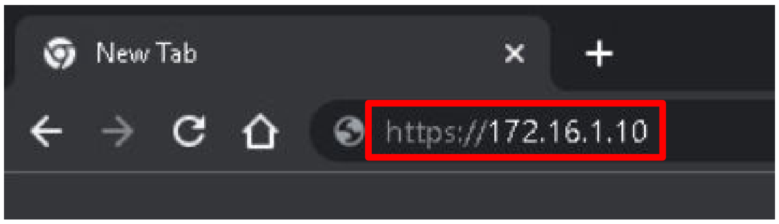
6. Click on the **Chrome** browser icon in the taskbar to open a web browser.



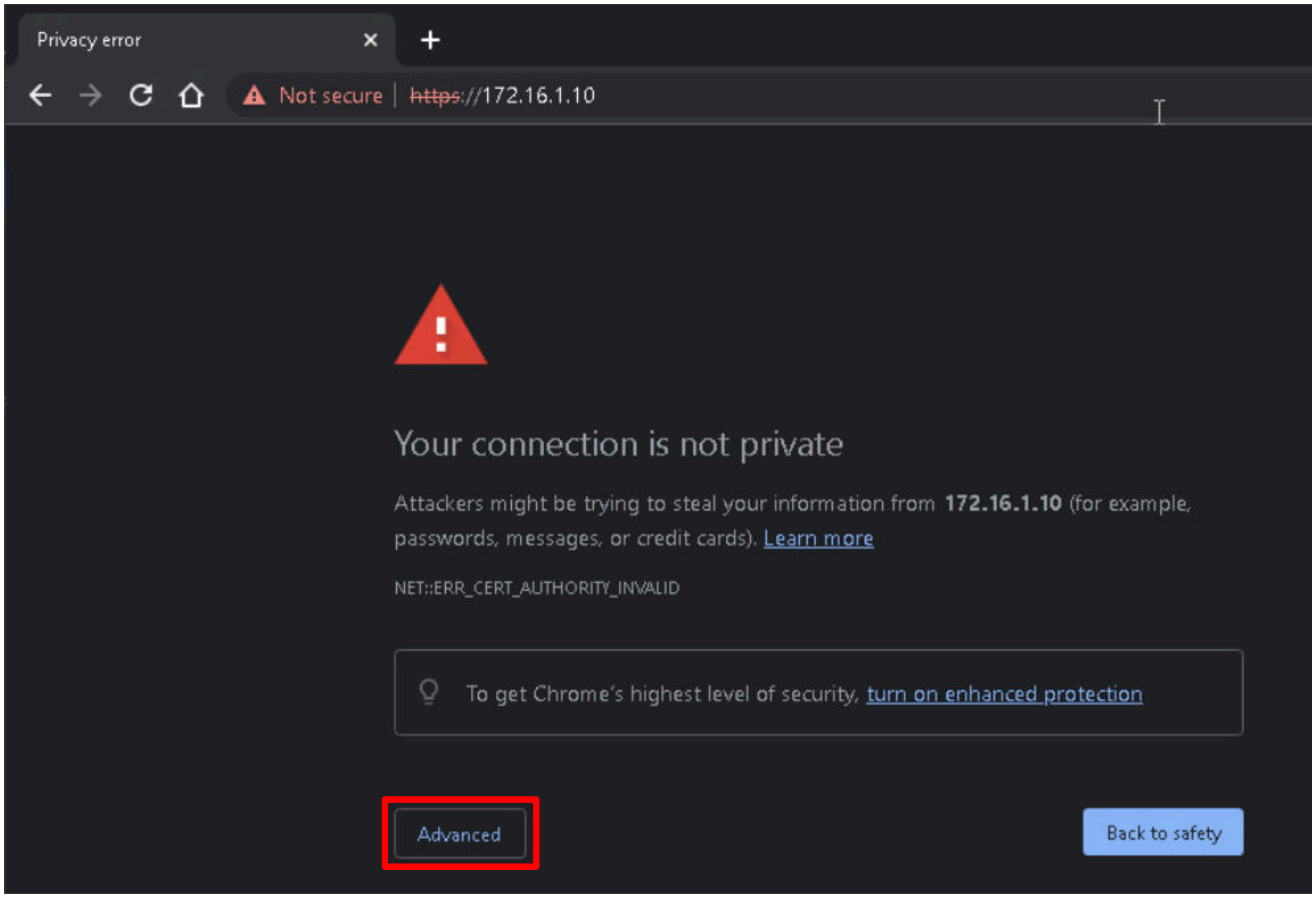
You will use the *Chrome* web browser for this part of the task because the *Firefox* web browser will cache the Apache2 web page, and will not send any HTTPS: requests.

You could achieve the same results in Firefox by going to Firefox's **Settings / Privacy & Security** then clear **Cookies and Site Data** and **History**

7. Open the web page to capture the HTTP packets. Type in the address `https://172.16.1.10`.



6. On the *Your Connection is not Private* page, click the **Advanced** button.



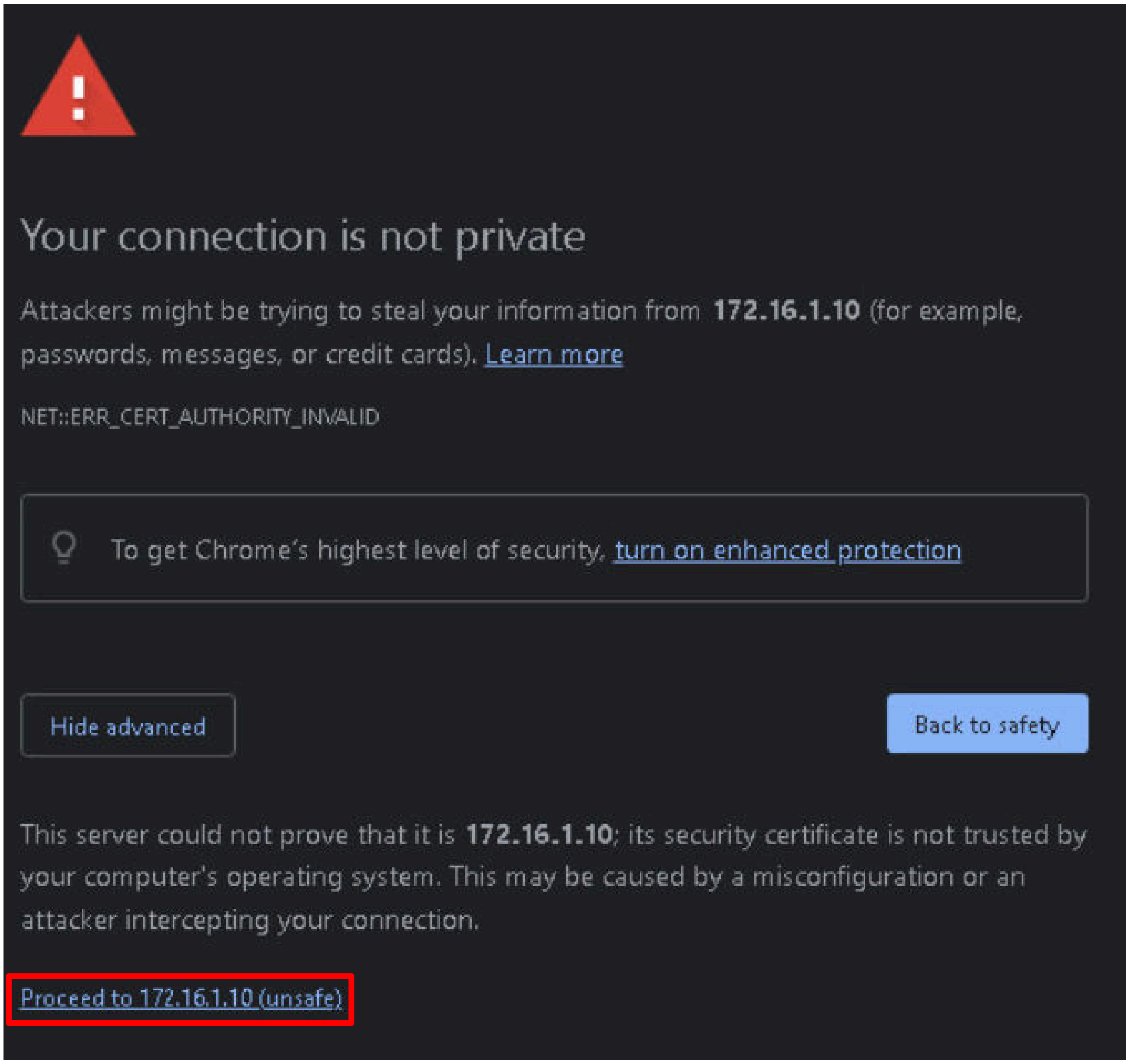
Your connection is not private

Attackers might be trying to steal your information from **172.16.1.10** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

To get Chrome's highest level of security, turn on enhanced protection

7. At the bottom of the window, click the **Proceed to 172.16.1.10 (unsafe)** link.



Your connection is not private

Attackers might be trying to steal your information from **172.16.1.10** (for example, passwords, messages, or credit cards). Learn more

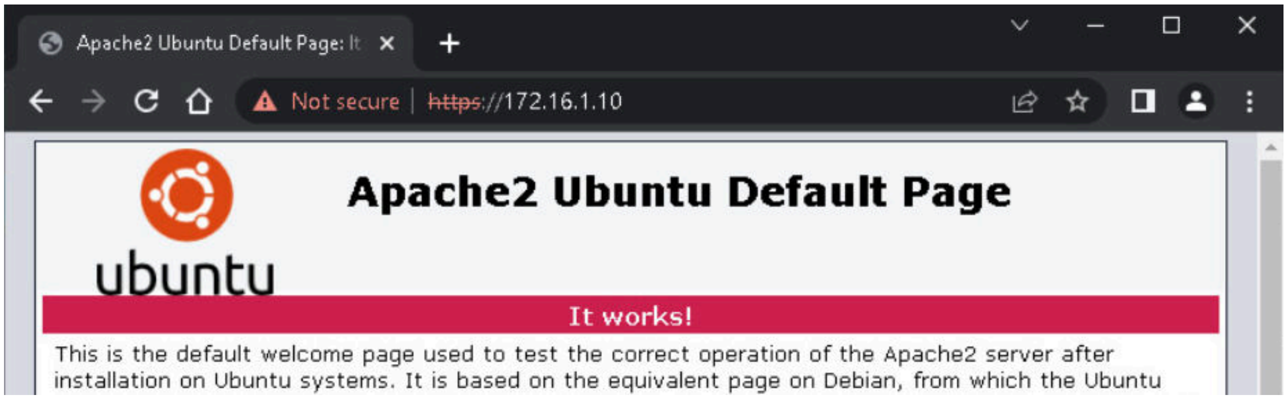
NET::ERR_CERT_AUTHORITY_INVALID

To get Chrome's highest level of security, turn on enhanced protection

This server could not prove that it is **172.16.1.10**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed to 172.16.1.10 (unsafe)

You will see the Apache2 Ubuntu Default page.

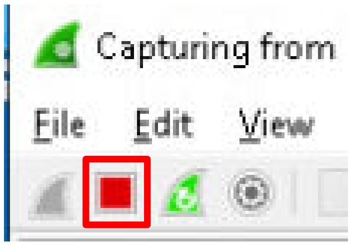


Apache2 Ubuntu Default Page

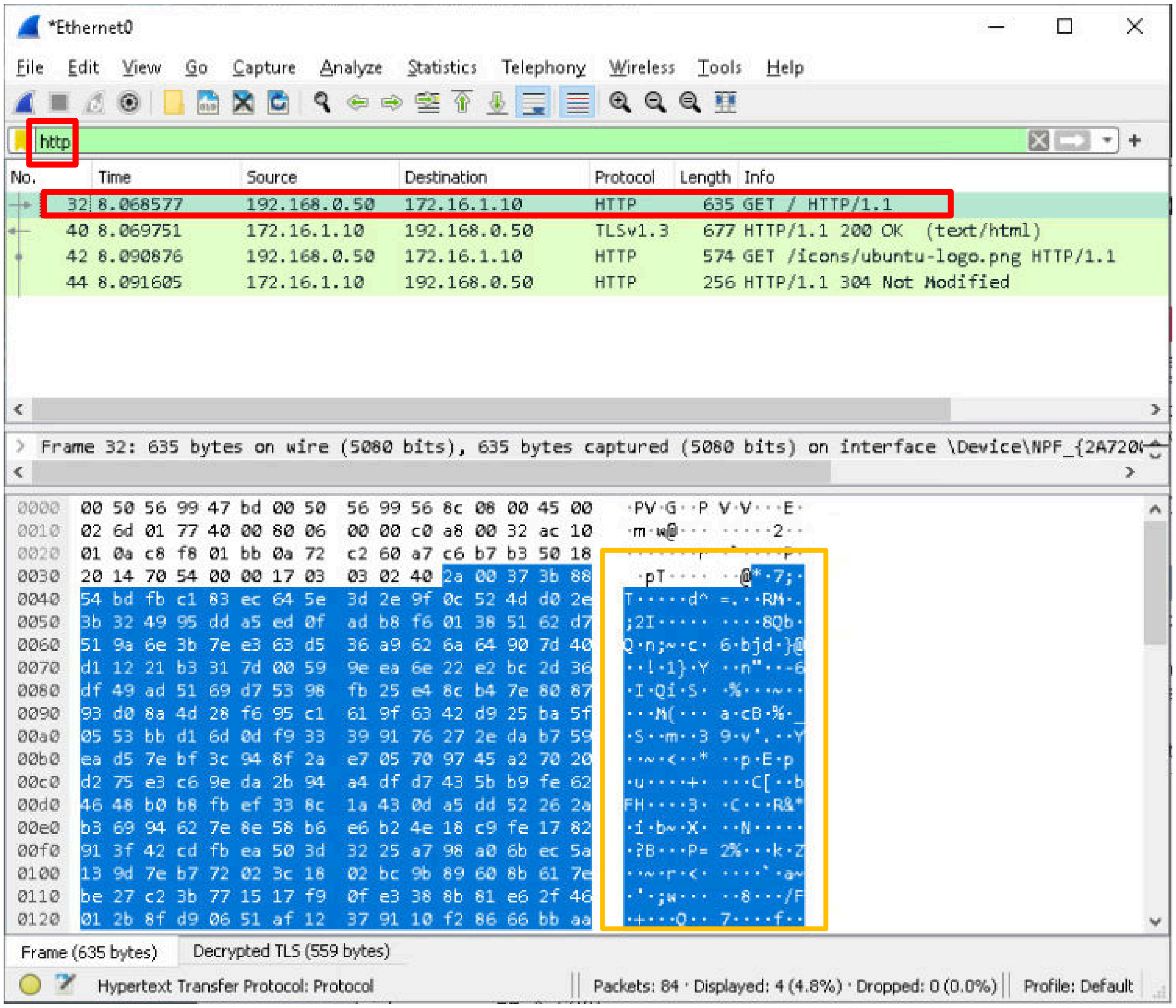
It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu

8. Go to the *Wireshark* window and stop the scan by clicking the **Stop Capturing Packet** button.

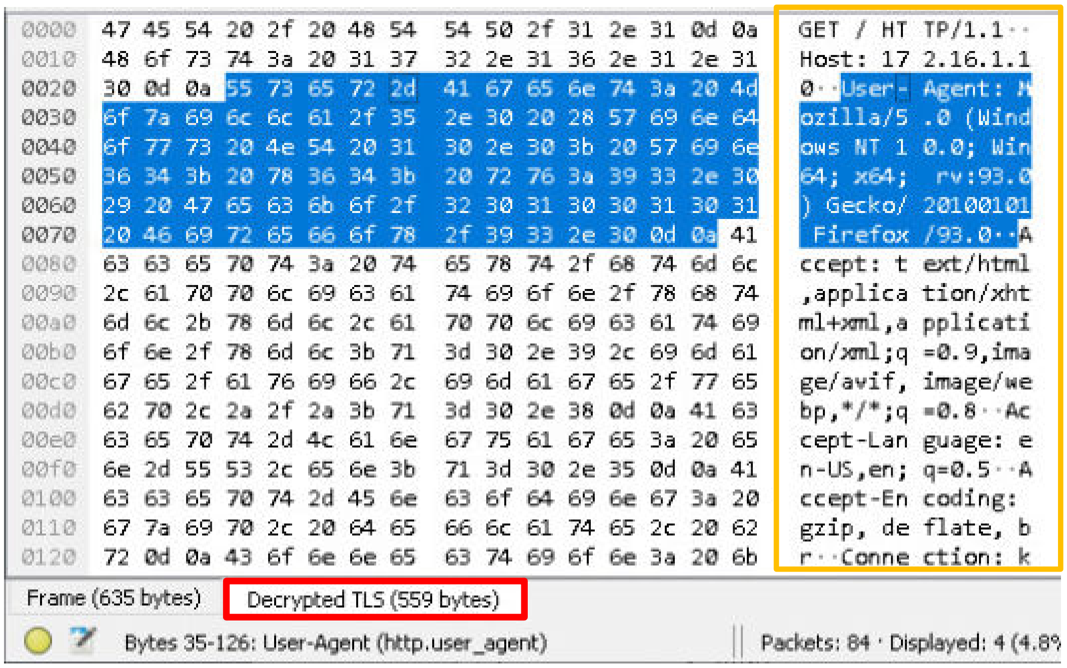


9. Go to the **FILTER** box, type in `http`, and press **Enter**. Each of the HTTP packets is encrypted. Click on the first packet that has **GET / HTTP/1.1**. The bottom panel will show the encrypted data.


*Ethernet0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
http

No.  Time      Source         Destination    Protocol  Length Info
  32 8.068577  192.168.0.50   172.16.1.10    HTTP      635 GET / HTTP/1.1
  40 8.069751  172.16.1.10    192.168.0.50   TLSv1.3   677 HTTP/1.1 200 OK  (text/html)
  42 8.090876  192.168.0.50   172.16.1.10    HTTP      574 GET /icons/ubuntu-logo.png HTTP/1.1
  44 8.091605  172.16.1.10    192.168.0.50   HTTP      256 HTTP/1.1 304 Not Modified

> Frame 32: 635 bytes on wire (5080 bits), 635 bytes captured (5080 bits) on interface \Device\NPF_{2A720

0000  00 50 56 99 47 bd 00 50  56 99 56 8c 08 00 45 00   ·PV·G··P V·V···E·
0010  02 6d 01 77 40 00 80 06  00 00 c0 a8 00 32 ac 10   ·m·w@··· ·····2··
0020  01 0a c8 f8 01 bb 0a 72  c2 60 a7 c6 b7 b3 50 18   ·······r ·`····P·
0030  20 14 70 54 00 00 17 03  03 02 40 2a 00 37 3b 88    ·pT···· ··@*·7;·
0040  54 bd fb c1 83 ec 64 5e  3d 2e 9f 0c 52 4d d0 2e   T·····d^ =.··RM·.
0050  3b 32 49 95 dd a5 ed 0f  ad b8 f6 01 38 51 62 d7   ;2I····· ····8Qb·
0060  51 9a 6e 3b 7e e3 63 d5  36 a9 62 6a 64 90 7d 40   Q·n;~·c· 6·bjd·}@
0070  d1 12 21 b3 31 7d 00 59  9e ea 6e 22 e2 bc 2d 36   ··!·1}·Y ··n"··-6
0080  df 49 ad 51 69 d7 53 98  fb 25 e4 8c b4 7e 80 87   ·I·Qi·S· ·%···~··
0090  93 d0 8a 4d 28 f6 95 c1  61 9f 63 42 d9 25 ba 5f   ···M(··· a·cB·%·_
00a0  05 53 bb d1 6d 0d f9 33  39 91 76 27 2e da b7 59   ·S··m··3 9·v'.··Y
00b0  ea d5 7e bf 3c 94 8f 2a  e7 05 70 97 45 a2 70 20   ··~·<··* ··p·E·p 
00c0  d2 75 e3 c6 9e da 2b 94  a4 df d7 43 5b b9 fe 62   ·u····+· ···C[··b
00d0  46 48 b0 b8 fb ef 33 8c  1a 43 0d a5 dd 52 26 2a   FH····3· ·C···R&*
00e0  b3 69 94 62 7e 8e 58 b6  e6 b2 4e 18 c9 fe 17 82   ·i·b~·X· ··N·····
00f0  91 3f 42 cd fb ea 50 3d  32 25 a7 98 a0 6b ec 5a   ·?B···P= 2%···k·Z
0100  13 9d 7e b7 72 02 3c 18  02 bc 9b 89 60 8b 61 7e   ··~·r·<· ····`·a~
0110  be 27 c2 3b 77 15 17 f9  0f e3 38 8b 81 e6 2f 46   ·'·;w··· ··8···/F
0120  01 2b 8f d9 06 51 af 12  37 91 10 f2 86 66 bb aa   ·+···Q·· 7····f··

Frame (635 bytes)   Decrypted TLS (559 bytes)
Hypertext Transfer Protocol: Protocol   Packets: 84 · Displayed: 4 (4.8%) · Dropped: 0 (0.0%)   Profile: Default
```

10. Earlier the **SSLKeyLogFile** environment variable was set, and the **SSL Key Log** was captured and added to the **Preferences** in *Wireshark*; the captured packets can be decrypted. Click on the **Decrypted TLS** button at the bottom of the window, and now you will see that the packet has been decrypted, and the contents can be seen in plain text.



```
0000  47 45 54 20 2f 20 48 54  54 50 2f 31 2e 31 0d 0a   GET / HT TP/1.1··
0010  48 6f 73 74 3a 20 31 37  32 2e 31 36 2e 31 2e 31   Host: 17 2.16.1.1
0020  30 0d 0a 55 73 65 72 2d  41 67 65 6e 74 3a 20 4d   0··User- Agent: M
0030  6f 7a 69 6c 6c 61 2f 35  2e 30 20 28 57 69 6e 64   ozilla/5 .0 (Wind
0040  6f 77 73 20 4e 54 20 31  30 2e 30 3b 20 57 69 6e   ows NT 1 0.0; Win
0050  36 34 3b 20 78 36 34 3b  20 72 76 3a 39 33 2e 30   64; x64;  rv:93.0
0060  29 20 47 65 63 6b 6f 2f  32 30 31 30 30 31 30 31   ) Gecko/ 20100101
0070  20 46 69 72 65 66 6f 78  2f 39 33 2e 30 0d 0a 41    Firefox /93.0··A
0080  63 63 65 70 74 3a 20 74  65 78 74 2f 68 74 6d 6c   ccept: t ext/html
0090  2c 61 70 70 6c 69 63 61  74 69 6f 6e 2f 78 68 74   ,applica tion/xht
00a0  6d 6c 2b 78 6d 6c 2c 61  70 70 6c 69 63 61 74 69   ml+xml,a pplicati
00b0  6f 6e 2f 78 6d 6c 3b 71  3d 30 2e 39 2c 69 6d 61   on/xml;q =0.9,ima
00c0  67 65 2f 61 76 69 66 2c  69 6d 61 67 65 2f 77 65   ge/avif, image/we
00d0  62 70 2c 2a 2f 2a 3b 71  3d 30 2e 38 0d 0a 41 63   bp,*/*;q =0.8··Ac
00e0  63 65 70 74 2d 4c 61 6e  67 75 61 67 65 3a 20 65   cept-Lan guage: e
00f0  6e 2d 55 53 2c 65 6e 3b  71 3d 30 2e 35 0d 0a 41   n-US,en; q=0.5··A
0100  63 63 65 70 74 2d 45 6e  63 6f 64 69 6e 67 3a 20   ccept-En coding: 
0110  67 7a 69 70 2c 20 64 65  66 6c 61 74 65 2c 20 62   gzip, de flate, b
0120  72 0d 0a 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 6b   r··Conne ction: k
```

```
Bytes 35-126: User-Agent (http.user_agent)      Packets: 84 · Displayed: 4 (4.8
```

11. The lab is now finished, and you can end the reservation.