



NETWORK SECURITY FUNDAMENTALS V2

Lab 4: Analyzing Packet Captures

Document Version: **2023-10-16**

Copyright © 2023 Network Development Group, Inc.
www.netdevgroup.com

NETLAB+ is a registered trademark of Network Development Group, Inc.

Palo Alto Networks and the Palo Alto Networks logo are trademarks or registered trademarks of Palo Alto Networks, Inc.

Contents

Introduction	3
Objective	3
Lab Topology.....	4
Lab Settings.....	5
1 Analyzing Packet Captures.....	6
1.0 Load Lab Configuration	6
1.1 Configure Wireshark to Decrypt Packet Captures.....	11
1.2 Analyze PCAP Files with Wireshark	18

Introduction

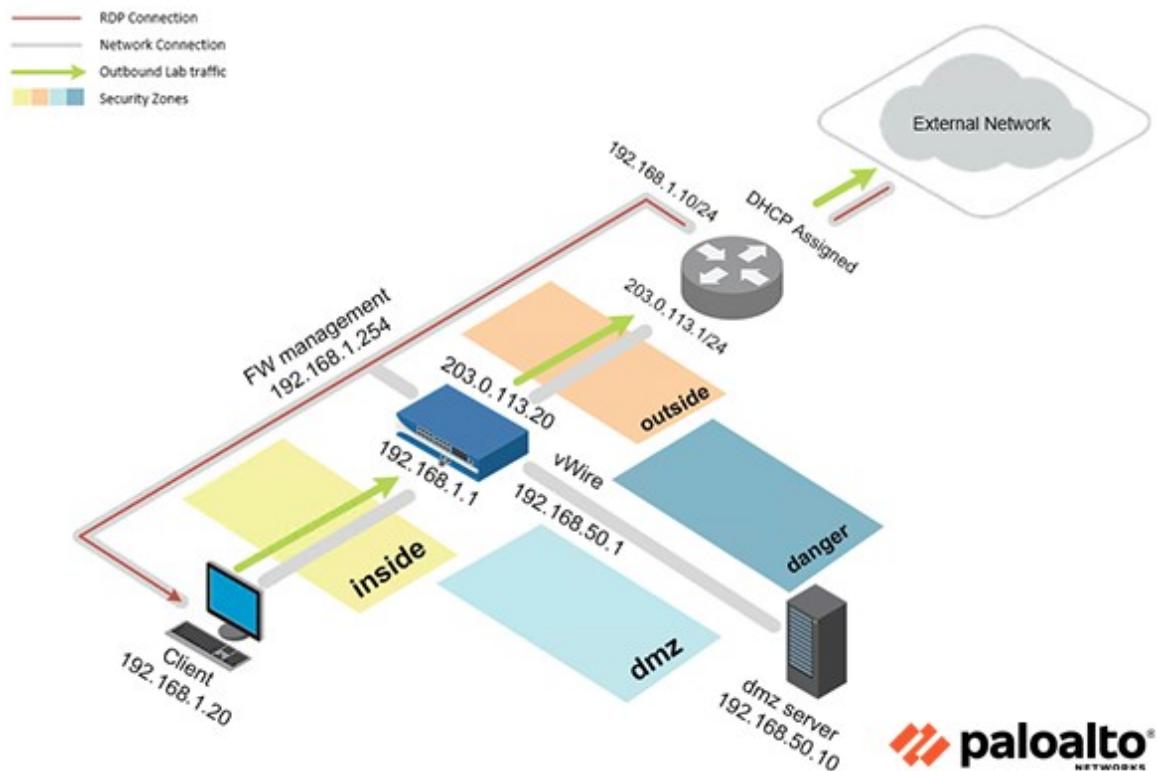
In this lab, you will configure *Wireshark* to decrypt SSL traffic and perform a packet capture to an SSL enabled site. Then, you will use *Wireshark* to explore capture files and examine the data within the packet.

Objective

In this lab, you will perform the following tasks:

- Configure Wireshark to decrypt packet captures from HTTPS sites
- Examine a HTTP Wireshark trace
- Examine a Wireshark decrypted packet trace

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.

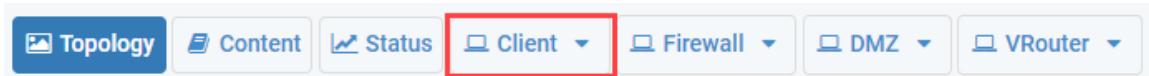
Virtual Machine	IP Address	Account (if needed)	Password (if needed)
Client	192.168.1.20	lab-user	PaloAlt0!
DMZ	192.168.50.10	root	PaloAlt0!
Firewall	192.168.1.254	admin	PaloAlt0!

1 Analyzing Packet Captures

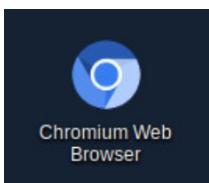
1.0 Load Lab Configuration

In this section, you will load the Firewall configuration file.

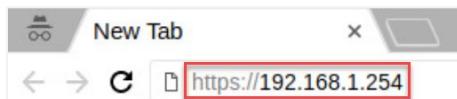
1. Click on the **Client** tab to access the Client PC.



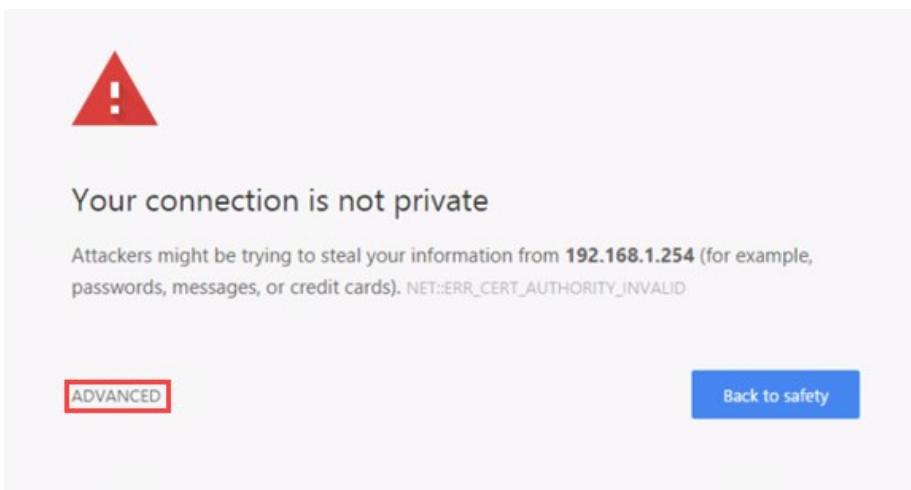
2. Log in to the Client PC with the username `lab-user` and password `Pal0Alt0!`.
3. Double-click the **Google Chrome** icon located on the Desktop.



4. In the *Chromium* address field, type `https://192.168.1.254` and press **Enter**.

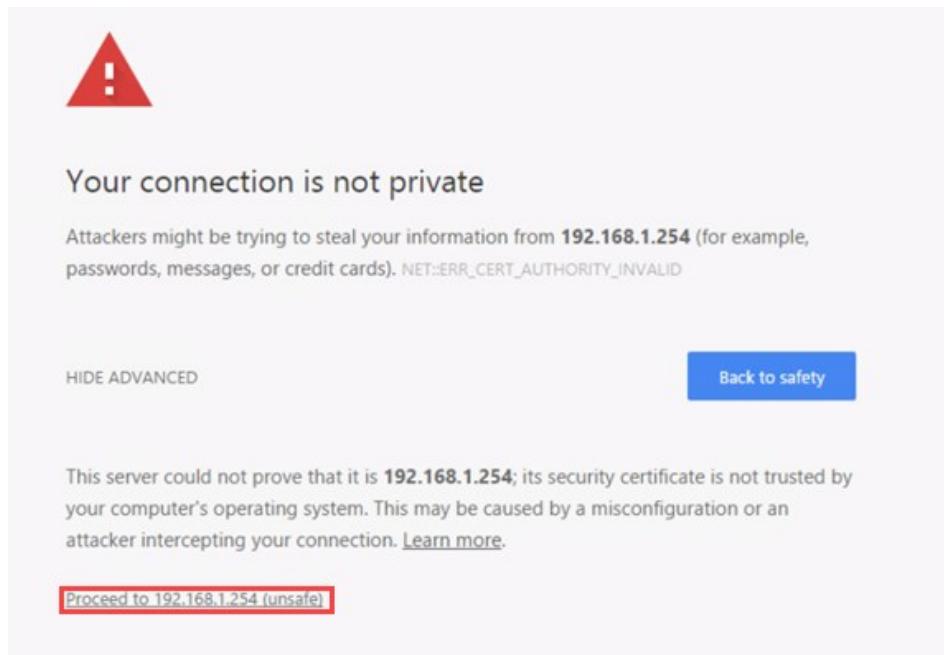


5. You will see a “*Your connection is not private*” message. Click on the **ADVANCED** link.



If you experience the “Unable to connect” or “502 Bad Gateway” message while attempting to connect to the specified IP above, please wait an additional 1-3 minutes for the Firewall to fully initialize. Refresh the page to continue.

6. Click on **Proceed to 192.168.1.254 (unsafe)**.



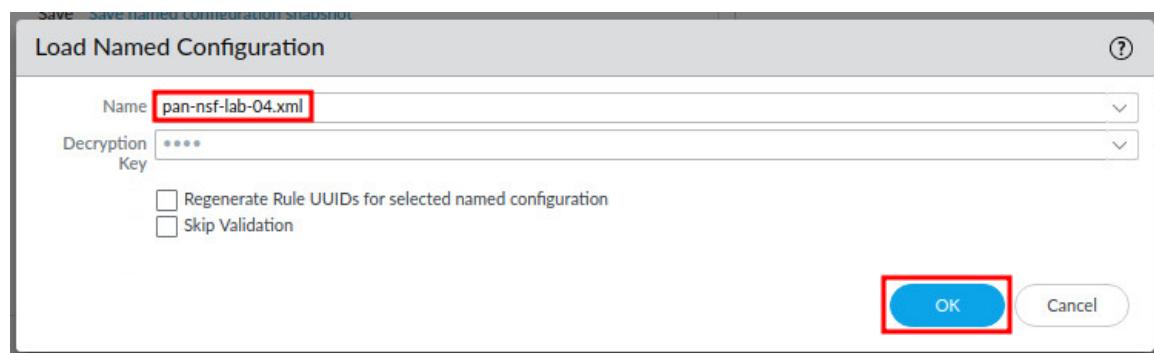
7. Log in to the Firewall web interface as username **admin**, password **Pal0Alt0!**.



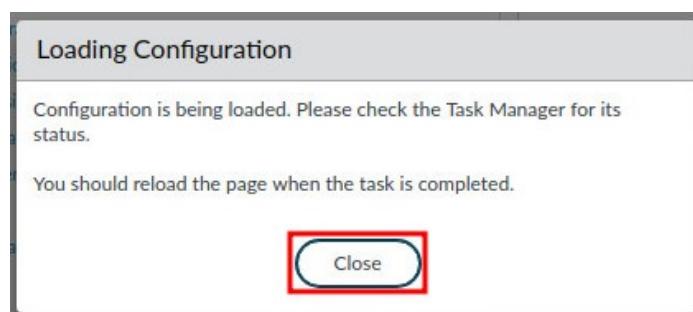
8. In the web interface, navigate to **Device > Setup > Operations** and click on **Load named configuration snapshot** underneath the *Configuration Management* section.

The screenshot shows the PA-VM web interface. The top navigation bar includes links for DASHBOARD, ACC, MONITOR, POLICIES, OBJECTS, NETWORK, and DEVICE. The DEVICE link is highlighted with a red box. The left sidebar has a 'Setup' icon highlighted with a red box, and under it, the 'Load named configuration snapshot' option is also highlighted with a red box. The main content area is titled 'Configuration Management' and contains several options: Revert, Save, Load, Export, Import, and various sub-options like 'Revert to last saved configuration', 'Save named configuration snapshot', etc. The 'Load named configuration snapshot' option is specifically highlighted with a red box.

9. In the *Load Named Configuration* window, select **pan-nsf-lab-04.xml** from the **Name** dropdown box and click **OK**.



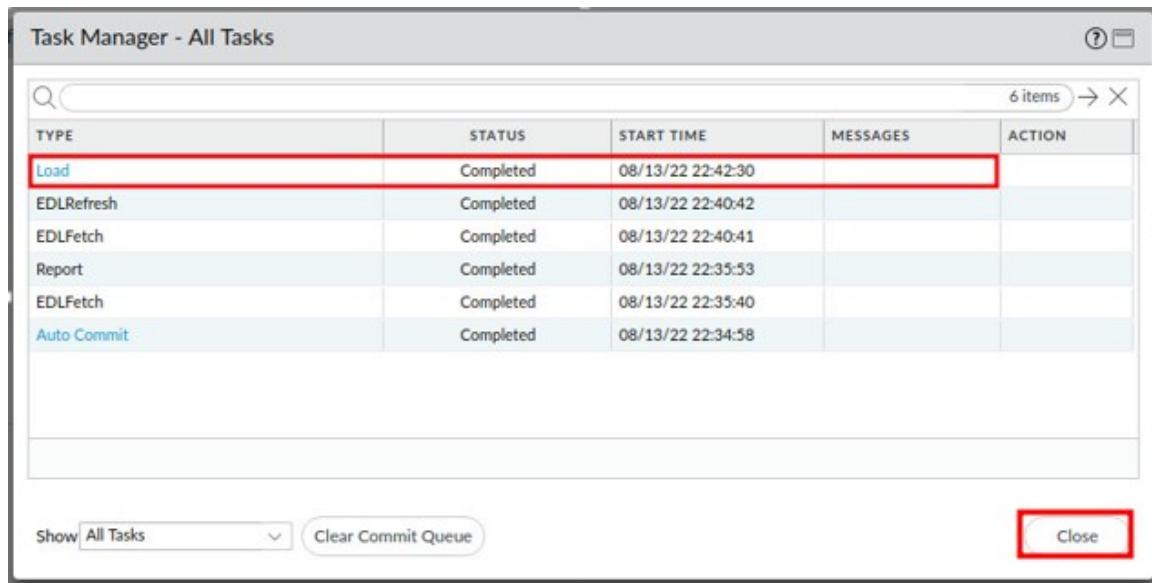
10. In the Loading Configuration window, a message will show *Configuration is being loaded. Please check the Task Manager for its status. You should reload the page when the task is completed.* Click **Close** to continue.



11. Click the **Tasks** icon located at the bottom-right of the web interface.



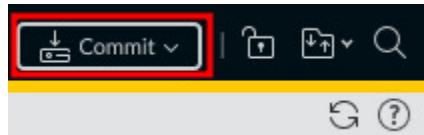
12. In the *Task Manager – All Tasks* window, verify the *Load* type has successfully completed. Click **Close**.



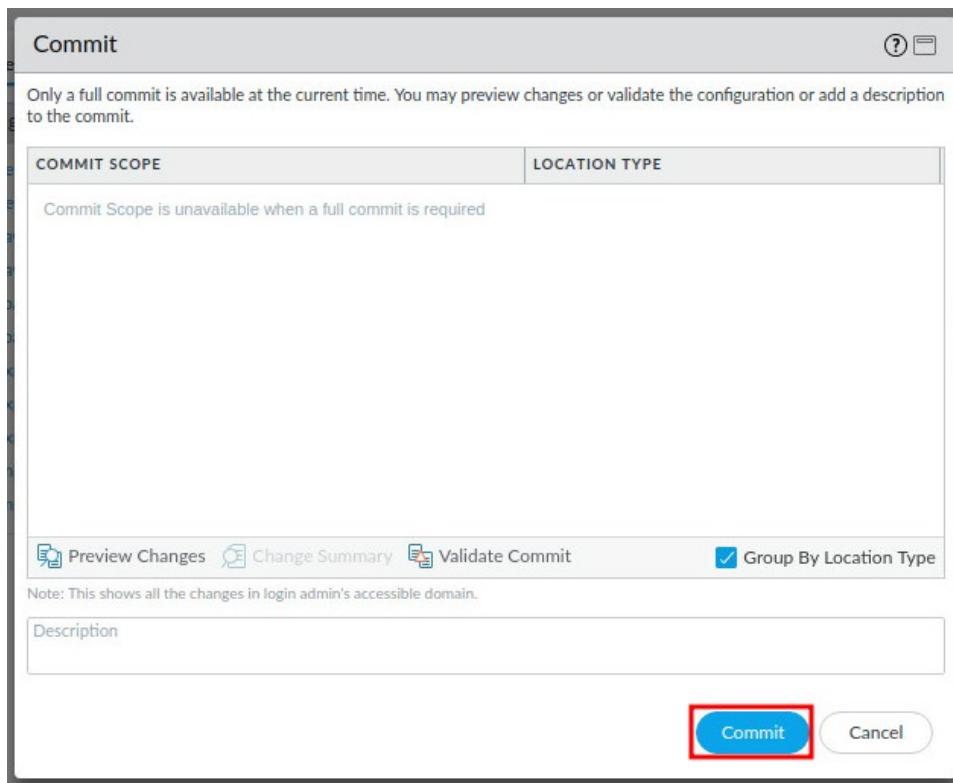
TYPE	STATUS	START TIME	MESSAGES	ACTION
Load	Completed	08/13/22 22:42:30		
EDLRefresh	Completed	08/13/22 22:40:42		
EDLFetch	Completed	08/13/22 22:40:41		
Report	Completed	08/13/22 22:35:53		
EDLFetch	Completed	08/13/22 22:35:40		
Auto Commit	Completed	08/13/22 22:34:58		

Show All Tasks

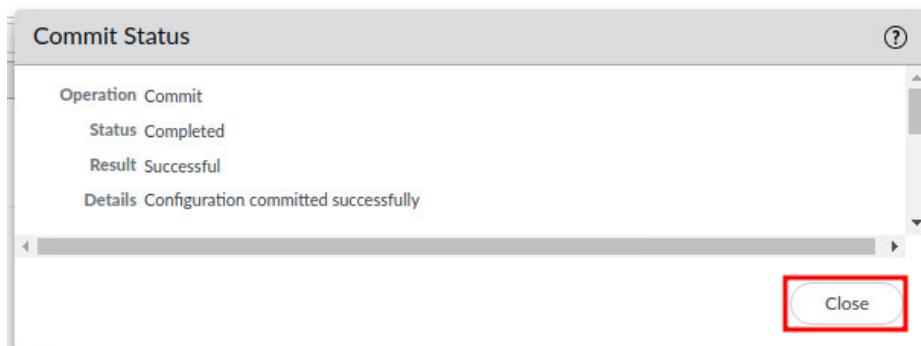
13. Click the **Commit** link located at the top-right of the web interface.



14. In the **Commit** window, click **Commit** to proceed with committing the changes.



15. When the commit operation successfully completes, click **Close** to continue.



The commit process takes changes made to the Firewall and copies them to the running configuration, which will activate all configuration changes since the last commit.

16. Close the web browser to continue on with the next task.

1.1 Configure Wireshark to Decrypt Packet Captures

In this section, you will configure Wireshark to decrypt SSL packet captures.

1. Open a new terminal by clicking on the **terminal** icon in the taskbar.



2. In the terminal, enter the command below to create a file named **.ssl-key.log**.

```
touch .ssl-key.log
```

```
C:\home\lab-user> touch .ssl-key.log  
C:\home\lab-user>
```

3. Enter the **ls** command below to verify the file you created is there.

```
ls -a
```

```
C:\home\lab-user> ls -a  
.  
..  
17717.pdf  
.aptitude  
.bash_history  
.bash_logout  
.bashrc  
.cache  
.config  
.dbus  
Desktop  
.dmrc  
C:\home\lab-user>  
docke...compose.yml  
docke...run-openvas  
Documents  
Downloads  
.gnupg  
.gtkrc-2.0  
.gtkrc-xfce  
.gvfs  
.ICEauthority  
.lessht  
.linuxmint  
.local  
mail  
minemeld  
.mozilla  
Music  
openvas-data  
Pictures  
.pki  
.profile  
Public  
.putty  
.python_history  
.recently-used  
.ssh  
.ssl-key.log  
.sudo_as_admin_successful  
Templates  
.thumbnails  
.thunderbird  
Videos  
.Xauthority  
.xsession-errors  
.xsession-errors.old
```

4. Enter the command below to open the **.bashrc** file using a text editor.

```
xed .bashrc
```

```
C:\home\lab-user> xed .bashrc
```

5. In the text editor, scroll all the way to the bottom, and at the end of the file, add the following line. Once finished, save your changes and close the text editor.

```
export SSLKEYLOGFILE=/home/lab-user/.ssl-key.log
```

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi
: undercover [[ export PS1=$C:\${PWD//\//\\} ]]
```

6. Test the *bashrc* configuration by entering the command below in your terminal.

```
source .bashrc
```

```
C:\home\lab-user> source .bashrc
```

7. Next, enter the follow-up command below. The response should be as shown in the screenshot below. If it isn't, go back and complete the steps above once more.

```
echo $SSLKEYLOGFILE
```

```
C:\home\lab-user> echo $SSLKEYLOGFILE  
/home/lab-user/.ssl-key.log  
C:\home\lab-user>
```

8. Confirm that the web browser is using the *ssl-key.log* file. In the terminal, enter the command below to open the web browser.

chromium-browser &

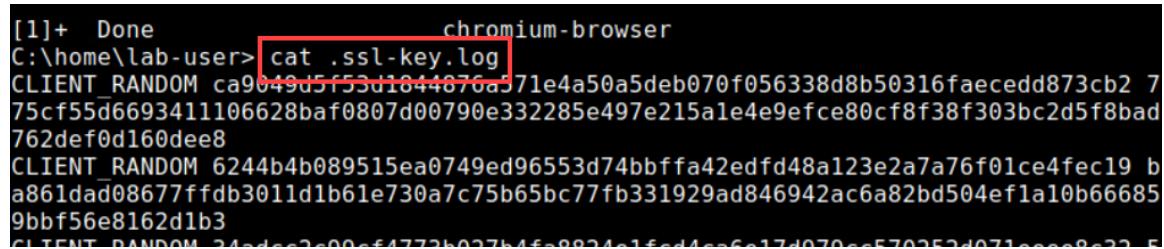
```
C:\home\lab-user> chromium-browser &
```

9. In the newly launched *Chromium* web browser, navigate to the URL: <https://www.paloaltonetworks.com/academy>. Once the website loads, wait one minute, and then close the web browser.



10. Back in the same terminal window, press the **Enter** key to return the terminal prompt and enter the command below. You should see an output similar to the screenshot below. If not, repeat the steps above.

```
cat .ssl-key.log
```

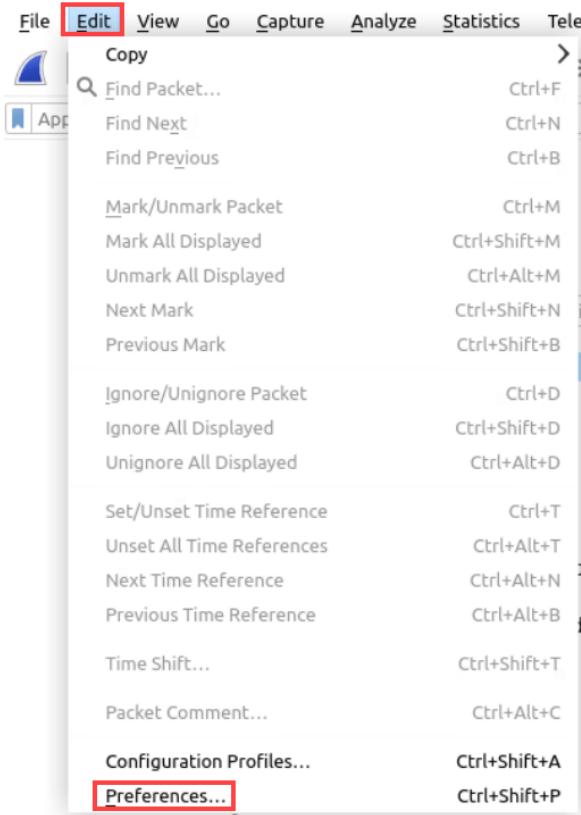


```
[1]+ Done chromium-browser
C:\home\lab-user> cat .ssl-key.log
CLIENT_RANDOM ca9049d5f53d1844876a571e4a50a5deb070f056338d8b50316faecedd873cb2 7
75cf55d6693411106628baf0807d00790e332285e497e215a1e4e9efce80cf8f38f303bc2d5f8bad
762def0d160dee8
CLIENT_RANDOM 6244b4b089515ea0749ed96553d74bbffa42edfd48a123e2a7a76f01ce4fec19 b
a861dad08677ffdb3011d1b61e730a7c75b65bc77fb331929ad846942ac6a82bd504ef1a10b66685
9bbf56e8162d1b3
CLIENT_RANDOM 24-adcc2e00cf1772b027b4fa8824e1fed4ea6a17d070cc570252d071aee8-22 f
```

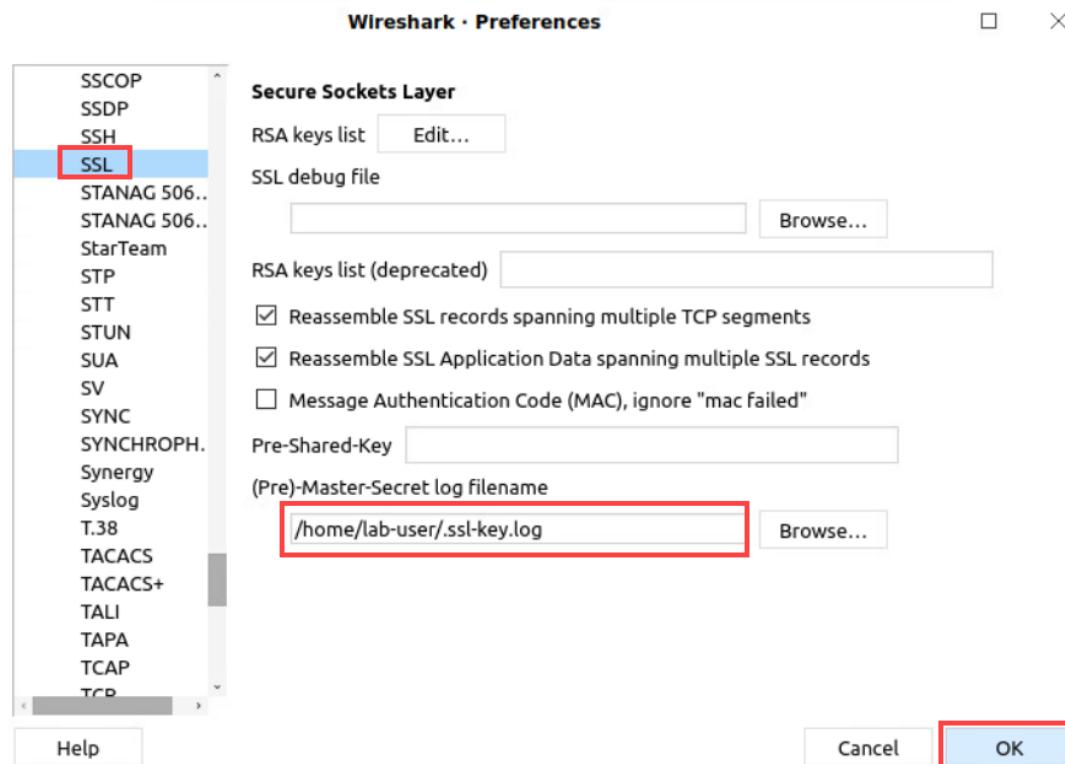
11. Minimize the terminal window and double-click the **Wireshark** icon on the desktop to open the application.



12. Once *Wireshark* launches, navigate to **Edit > Preferences**.



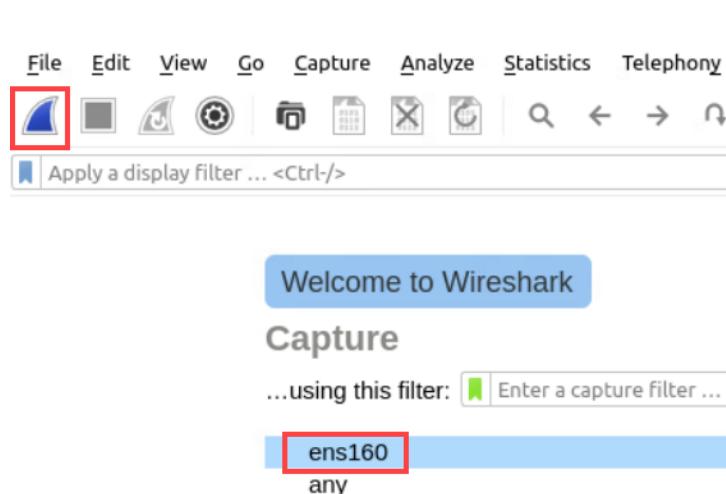
13. In the **Wireshark Preferences** window, expand **Protocols** in the left pane and then select **SSL**. Enter the full file path to the **.ssl-key.log** file in the **(Pre)-Master-Secret log filename** box: `/home/lab-user/.ssl-key.log`. Once finished, click **OK**.



14. Close the **Wireshark** application and make sure all opened web browser applications are closed.
15. Double-click on the **Wireshark** icon found on the Desktop to launch a new window.



16. In the *Wireshark* window, make sure that **ens160** is selected and click the **Start capturing packets** button.



17. Change focus back to your existing terminal window and enter the command below to open a web browser.

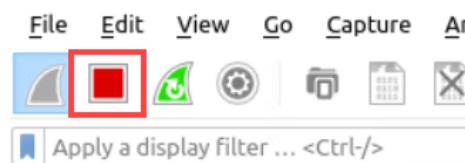
```
chromium-browser &
```

```
C:\home\lab-user> chromium-browser & [redacted]
```

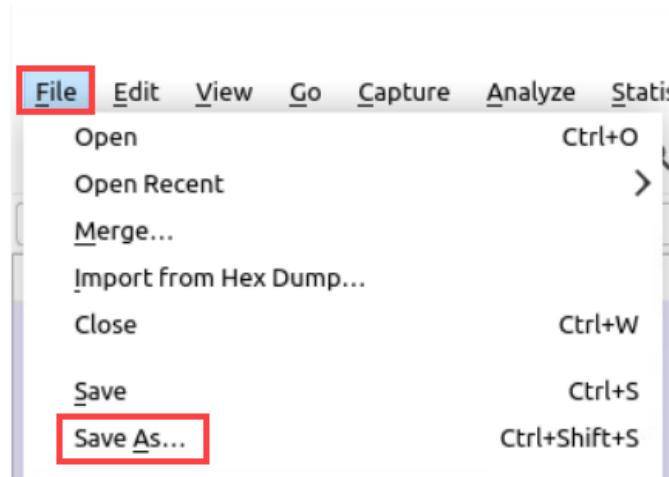
18. In the newly launched *Chromium* web browser, navigate to the URL: <https://www.paloaltonetworks.com/academy>. Once loaded, wait one minute.



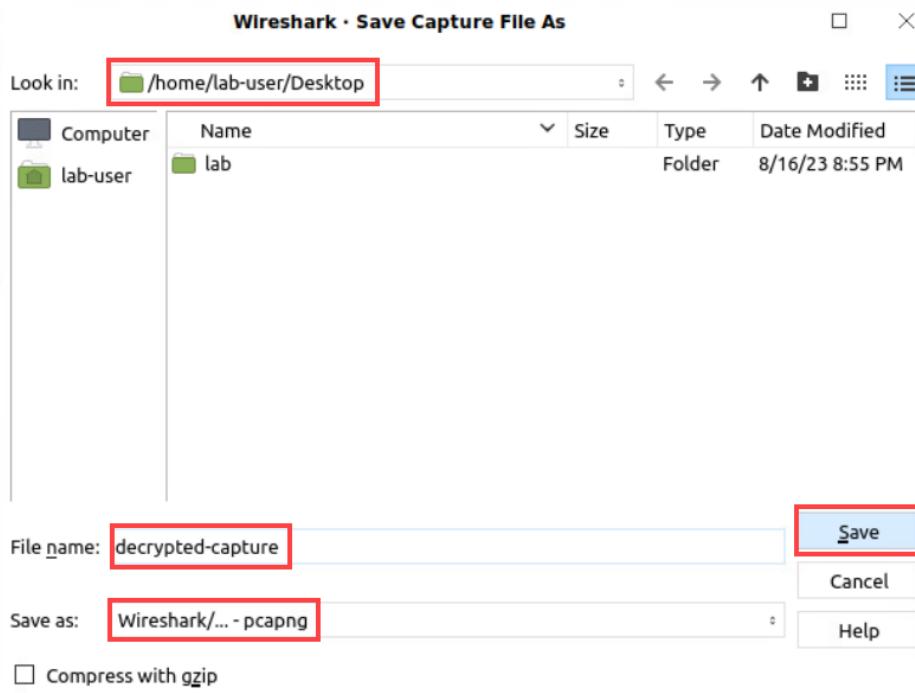
19. Change focus to the Wireshark window and click the Stop capturing packets button.



20. Save the packet capture by navigating to **File > Save As**.



21. In the Wireshark Save Capture File As window, name the file decrypted-capture using **pcapng** as the filetype. Select the **Desktop** as the save to location. Click **Save**.



22. Close the **Wireshark** window as well as any open web browser windows.

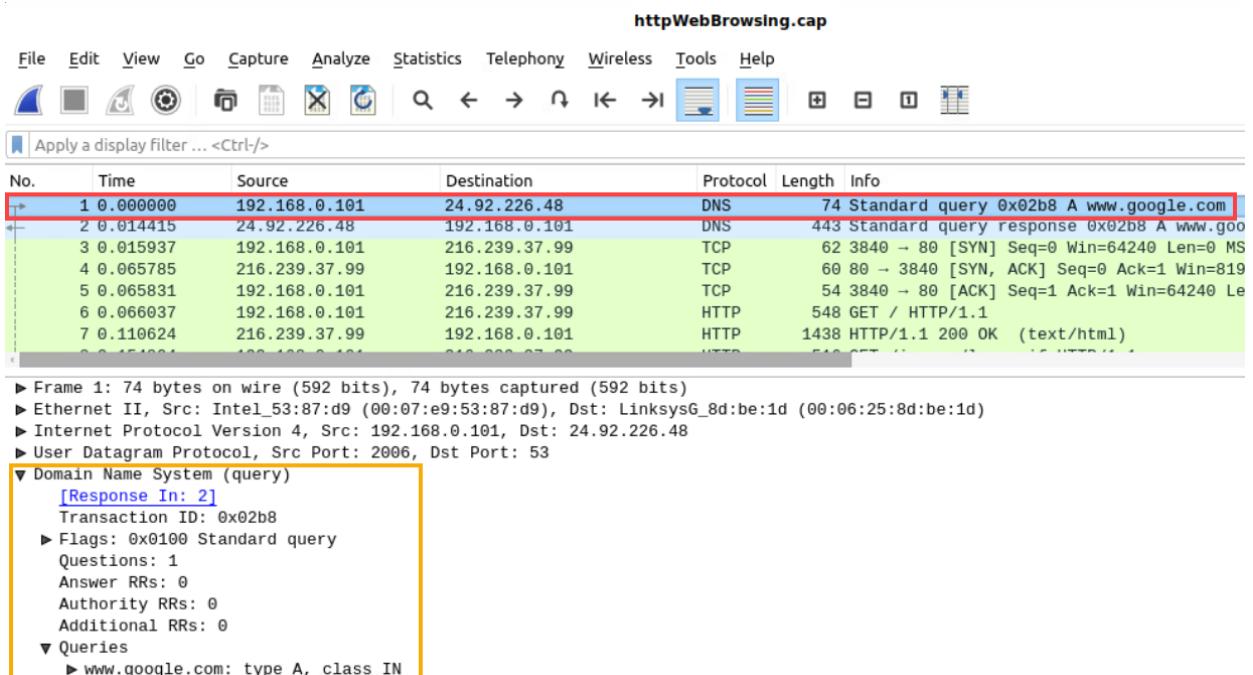
1.2 Analyze PCAP Files with Wireshark

In this section, you will analyze the traffic capture using Wireshark.

1. The first packet capture for examination is a packet capture that did not utilize an SSL/TLS encrypted connection. This will allow you to see the TCP/IP protocols without using complex Wireshark filters. You will then examine a decrypted packet capture for comparison. On the *Desktop*, double-click the **httpWebBrowsing.cap** packet capture to open it in Wireshark.



2. The first protocol you will analyze is DNS. Select the **first DNS packet** to examine it.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.101	24.92.226.48	DNS	74	Standard query 0x02b8 A www.google.com
2	0.014415	24.92.226.48	192.168.0.101	DNS	443	Standard query response 0x02b8 A www.google.com
3	0.015937	192.168.0.101	216.239.37.99	TCP	62	3840 → 80 [SYN] Seq=0 Win=64240 Len=0 MS
4	0.065785	216.239.37.99	192.168.0.101	TCP	60	80 → 3840 [SYN, ACK] Seq=0 Ack=1 Win=819
5	0.065831	192.168.0.101	216.239.37.99	TCP	54	3840 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
6	0.066037	192.168.0.101	216.239.37.99	HTTP	548	GET / HTTP/1.1
7	0.110624	216.239.37.99	192.168.0.101	HTTP	1438	HTTP/1.1 200 OK (text/html)
8	0.111004	192.168.0.101	216.239.37.99	HTTP	516	HTTP/1.1 200 OK (text/html)

► Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
 ► Ethernet II, Src: Intel_53:87:d9 (00:07:e9:53:87:d9), Dst: LinksysG_8d:be:1d (00:06:25:8d:be:1d)
 ► Internet Protocol Version 4, Src: 192.168.0.101, Dst: 24.92.226.48
 ► User Datagram Protocol, Src Port: 2006, Dst Port: 53
 ▾ Domain Name System (query)
 [Response In: 2]
 Transaction ID: 0x02b8
 ► Flags: 0x0100 Standard query
 Questions: 1
 Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 0
 ▾ Queries
 ► www.google.com: type A, class IN



In the first packet, the source is **192.168.0.101**, while the destination is **24.92.226.48**. The source is using **24.92.226.48** as its *DNS* server. The info column shows it is a standard query asking for the **A** (IPv4) record for **www.google.com**.

3. Select the second DNS packet and examine the contents.

Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.101	24.92.226.48	DNS	74	Standard query 0x02b8 A www.google.com
2	0.014415	24.92.226.48	192.168.0.101	DNS	443	Standard query response 0x02b8 A www.google.com
3	0.015937	192.168.0.101	216.239.37.99	TCP	62	3840 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=146
4	0.065785	216.239.37.99	192.168.0.101	TCP	60	80 → 3840 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len
5	0.065831	192.168.0.101	216.239.37.99	TCP	54	3840 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
6	0.066037	192.168.0.101	216.239.37.99	HTTP	548	GET / HTTP/1.1
7	0.110624	216.239.37.99	192.168.0.101	HTTP	1438	HTTP/1.1 200 OK (text/html)
8	0.151004	192.168.0.101	216.239.37.99	HTTP	540	GET / HTTP/1.1

▼ Domain Name System (response)
 [Request In: 1]
 [Time: 0.014415000 seconds]
 Transaction ID: 0x02b8
 ► Flags: 0x8180 Standard query response, No error
 Questions: 1
 Answer RRs: 4
 Authority RRs: 9
 Additional RRs: 7
 ▼ Queries
 ► www.google.com: type A, class IN
 ▼ Answers
 ► www.google.com: type CNAME, class IN, cname www.google.akadns.net
 ► www.google.akadns.net: type A, class IN, addr 216.239.37.99
 ► www.google.akadns.net: type A, class IN, addr 216.239.37.104
 ► www.google.akadns.net: type A, class IN, addr 216.239.37.147
 ► Authoritative nameservers



In the second packet, the source is the DNS server (24.92.226.48), while the destination is 192.168.0.101. The info column shows a standard query answer for A record [www.google.com](#). The answers show 3 IP addresses that could be used to connect to the [www.google.com](#) web server.

4. Next, examine the 3-way handshake between 192.168.0.101 and the Google Server (216.239.37.99). Select **packet #3**.

Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.101	24.92.226.48	DNS	74	Standard query 0x02b8 A www.google.com
2	0.014415	24.92.226.48	192.168.0.101	DNS	443	Standard query response 0x02b8 A www.google.com
3	0.015937	192.168.0.101	216.239.37.99	TCP	62	3840 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=146
4	0.065785	216.239.37.99	192.168.0.101	TCP	60	80 → 3840 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len
5	0.065831	192.168.0.101	216.239.37.99	TCP	54	3840 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
6	0.066037	192.168.0.101	216.239.37.99	HTTP	548	GET / HTTP/1.1
7	0.110624	216.239.37.99	192.168.0.101	HTTP	1438	HTTP/1.1 200 OK (text/html)
8	0.151004	192.168.0.101	216.239.37.99	HTTP	540	GET / HTTP/1.1

► Frame 3: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
 ► Ethernet II, Src: Intel_53:87:d9 (00:07:e9:53:87:d9), Dst: LinksysG_8d:be:1d (00:06:25:8d:be:1d)
 ► Internet Protocol Version 4, Src: 192.168.0.101, Dst: 216.239.37.99
 ▼ Transmission Control Protocol, Src Port: 3840, Dst Port: 80, Seq: 0, Len: 0
 Source Port: 3840
 Destination Port: 80
 [Stream index: 0]
 [TCP Segment Len: 0]
 Sequence number: 0 (relative sequence number)
 Acknowledgment number: 0
 0111 = Header Length: 28 bytes (7)
 ► Class: Ax002 (SYN)



In the first part of the 3-way handshake, the source (192.168.0.101), sends a packet requesting a new session which is indicated by the TCP SYN flag being set in the header. To the destination *Google* web server (216.239.37.99).

5. Select **packet #4** and examine the packet information.

Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.101	24.92.226.48	DNS	74	Standard query 0x02b8
2	0.014415	24.92.226.48	192.168.0.101	DNS	443	Standard query response
3	0.015937	192.168.0.101	216.239.37.99	TCP	62	3840 → 80 [SYN] Seq=0
4	0.065785	216.239.37.99	192.168.0.101	TCP	60	80 → 3840 [SYN, ACK]
5	0.065831	192.168.0.101	216.239.37.99	TCP	54	3840 → 80 [ACK] Seq=1
6	0.066037	192.168.0.101	216.239.37.99	HTTP	548	GET / HTTP/1.1
7	0.110624	216.239.37.99	192.168.0.101	HTTP	1438	HTTP/1.1 200 OK (text/html)
8	0.151024	192.168.0.101	216.239.37.99	HTTP	510	GET / HTTP/1.1 200 OK

```

▶ Frame 4: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
▶ Ethernet II, Src: LinksysG_8d:be:1d (00:06:25:8d:be:1d), Dst: Intel_53:87:d9 (00:07:e9:53:87:d9)
▶ Internet Protocol Version 4, Src: 216.239.37.99, Dst: 192.168.0.101
▼ Transmission Control Protocol, Src Port: 80, Dst Port: 3840, Seq: 0, Ack: 1, Len: 0
  Source Port: 80
  Destination Port: 3840
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 0      (relative sequence number)
  Acknowledgment number: 1      (relative ack number)
  0110 .... = Header Length: 24 bytes (6)
  ▶ Flags: 0x012 (SYN, ACK)
  Window size value: 8190

```



In the second part of the 3-way handshake, the source (216.239.37.99) sends a TCP packet with the flags *SYN* and *ACK* set in the header, to the destination (192.168.0.101). The server is acknowledging the original *SYN* packet it received and is confirming that it is ready to establish a new session with 192.168.0.101.

6. Select **packet #5** and examine the packet information.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.101	24.92.226.48	DNS	74	Standard query 0x02b8 A www
2	0.014415	24.92.226.48	192.168.0.101	DNS	443	Standard query response 0x02b8
3	0.015937	192.168.0.101	216.239.37.99	TCP	62	3840 → 80 [SYN] Seq=0 Win=64
4	0.065785	216.239.37.99	192.168.0.101	TCP	60	80 → 3840 [SYN, ACK] Seq=0 Ack=1
5	0.065831	192.168.0.101	216.239.37.99	TCP	54	3840 → 80 [ACK] Seq=1 Ack=1
6	0.066037	192.168.0.101	216.239.37.99	HTTP	548	GET / HTTP/1.1
7	0.110624	216.239.37.99	192.168.0.101	HTTP	1438	HTTP/1.1 200 OK (text/html)
8	0.151024	192.168.0.101	216.239.37.99	HTTP	540	GET / HTTP/1.1

```

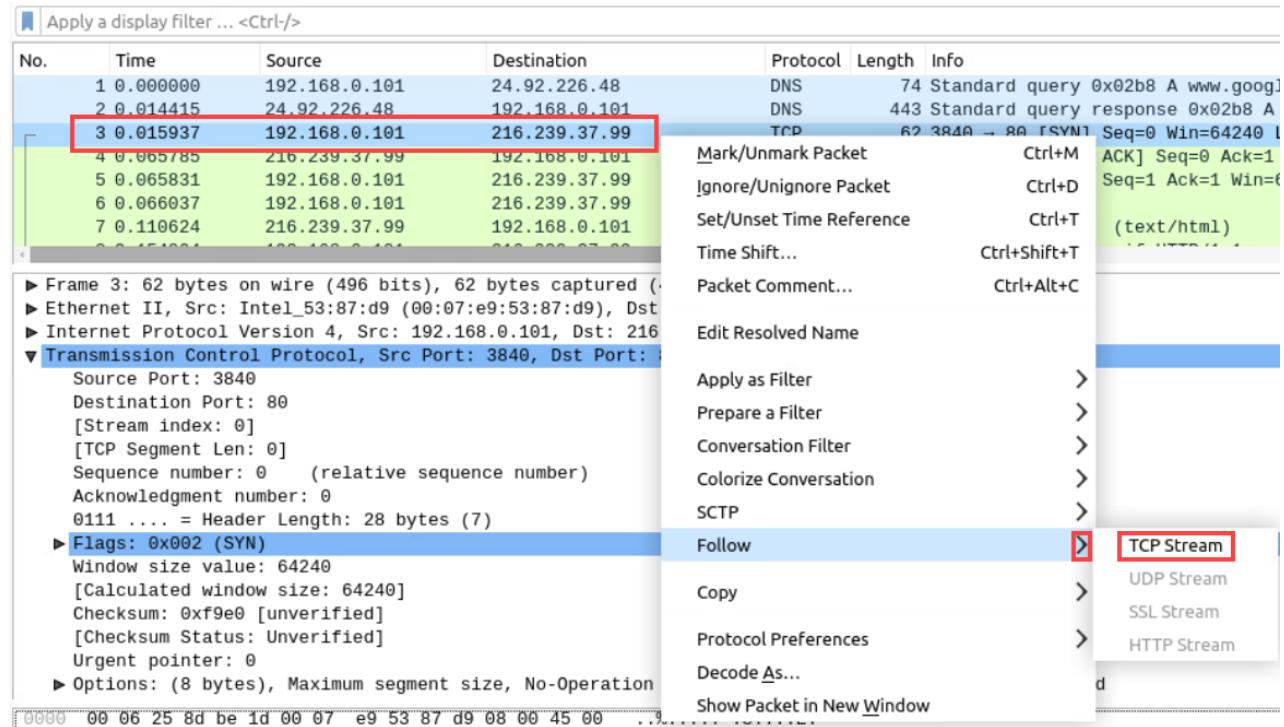
▶ Frame 5: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
▶ Ethernet II, Src: Intel_53:87:d9 (00:07:e9:53:87:d9), Dst: LinksysG_8d:be:1d (00:06:25:8d:be:1d)
▶ Internet Protocol Version 4, Src: 192.168.0.101, Dst: 216.239.37.99
▼ Transmission Control Protocol, Src Port: 3840, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
  Source Port: 3840
  Destination Port: 80
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 1      (relative sequence number)
  Acknowledgment number: 1    (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  ▶ Flags: 0x010 (ACK)
  Window size value: 64240

```



In the third part of the 3-way handshake, the source (**192.168.0.101**) sends a TCP packet with the flag **ACK** set in the header to the destination (**216.239.37.99**). The session between the two systems is now established for data transmission.

7. All the packets ranging from #3 - #20 represent the *TCP Stream* that makes up the communication between the two systems for the *TCP Established* connection. View all the packets comprising this session by right-clicking **packet #3** and selecting **Follow > TCP Stream**.



8. Observe the TCP Stream information presented in the new window. Cycle through the other TCP Streams in this capture by clicking the **top arrow** next to *Stream* on the bottom right of the window.

Wireshark · Follow TCP Stream (tcp.stream eq 0) · httpWebBrowsing

```

GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.5) Gecko/20031007
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PREF=ID=2922596a77b005c7:TM=1073520455:LM=1073520455:S=Ycw7Yx3HeW-X0ndK

HTTP/1.1 200 OK
Cache-control: private
Content-Type: text/html
Content-Encoding: gzip
Server: GWS/2.1
Content-length: 1216
Date: Thu, 08 Jan 2004 00:11:54 GMT

.....mo.6....Wp*`..d.N....a..6C6.h.},(...J.BRv...}G.~.+R. ....W....
$0..j.*....v1..LpM.....r+.i.M.F1BYE....y..{I...i.A...qhW...
+..K..."_...:.....AA.V....d....*PT."..~.P.?z.f.....D.... .dYd. r. I4...q.)..a.
$.qh.?..U&Y...8Rt<3.H....\d].....:....Pv...0Gw."Q....+..a1&V10@..J....y.
\]#.V;....F..p..a....F&..byN9...X...-H.Q....G^EU1f.S....k....0....^....8#<..e..
5.....i.ZR.I.
R+.V..!....5..V.I+J.>/..M....'i2A5....Z..|>%.)"vU.....%3.C%...WQ*dN%. ..V-.../..
%y..@...0.)....B.....STQVV.O..p.J.V..rhL..V...1J....q.M.....e.g9.1.....`Qrl.v.o.P.KI.
$6....l....^./M.0....X.9....ia~_<.R..f.W.....r\....h.tY.C..dC.....
(.P..p.I....."<.....D)|.*I.>q`.....<....]3....-Gs.....
;.*~.....u.....r?~0....).G....l~9....|....v.....
{....-g.fZ.."<....RY...0.!...."!|.....jwoh../q....c.E..M....&N.....%2.<....u+{}.
8{....'..U....a..Lz*824.g..].....A..`C.]S^..:?:....Z..v,...c..m....T.../\.
...R.$t.e_W[..m%,,:...-.V....9..NHG..J.I.0cS..&..t.....J
..Y.g....5.e....P....D7f.^..a.^..Lm"1....).D.B.DI.f.Z'.LW..s..
.V.f... @..S.S...Qw.N<V.m.....4...{...bHR..q...."U....j..c.#..]"..r..v..&..../
n hi ? n n 17 nth a ? Ra

```

4 client pkts, 10 server pkts, 5 turns.

Entire conversation (11 kB) Show and save data as ASCII Stream

Find: Find Next

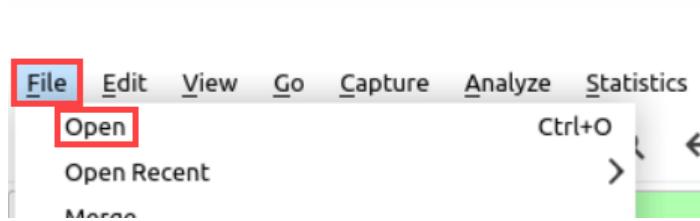
Help Filter Out This Stream Print Save as... Back Close



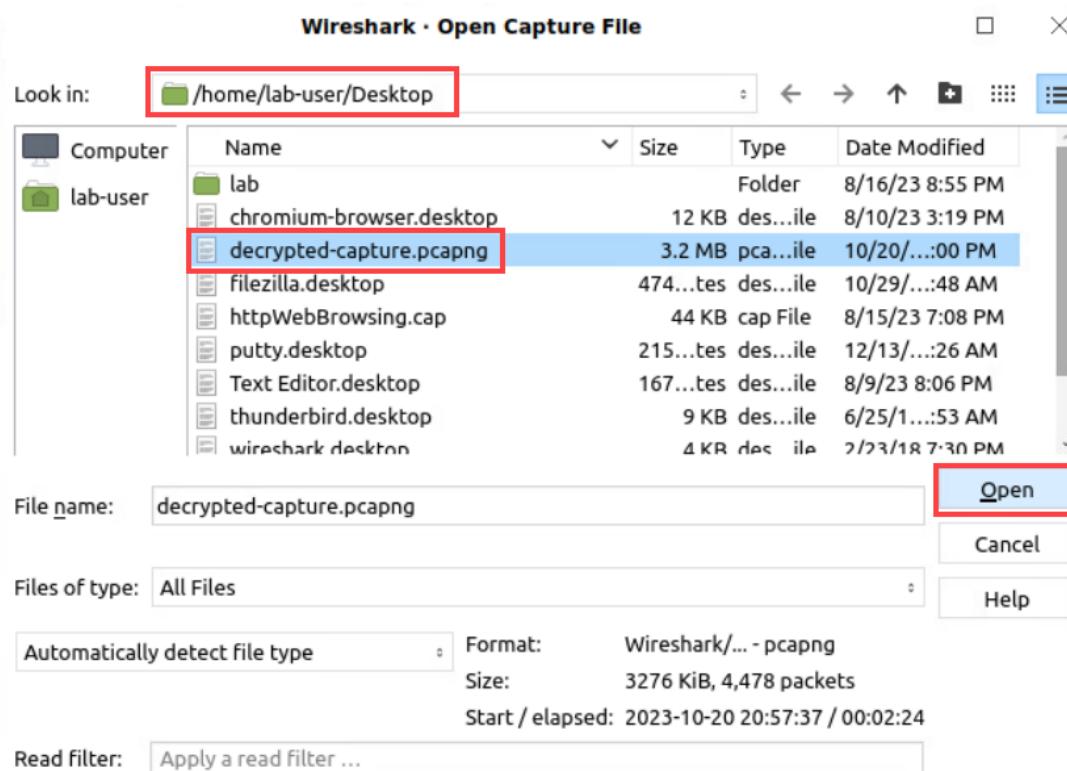
Notice the assembled packets represent the HTML website visited, with the red representing requests from the original source **192.168.0.101** and the blue representing the server destination **216.239.37.99**.

9. Close the **Follow TCP Stream** window.

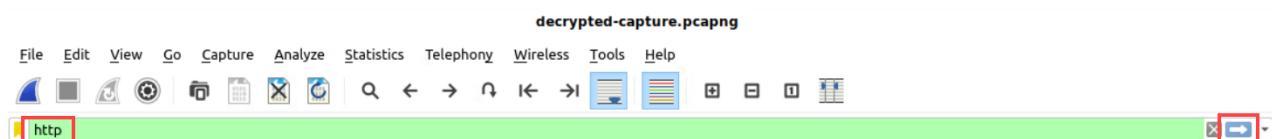
10. Almost all web traffic is encrypted by SSL/TLS and so to view all the protocols in a packet capture, you first need to decrypt it. You did this in a previous task and saved the packet capture to your *Desktop*. In the *Wireshark* window, navigate to **File > Open**.



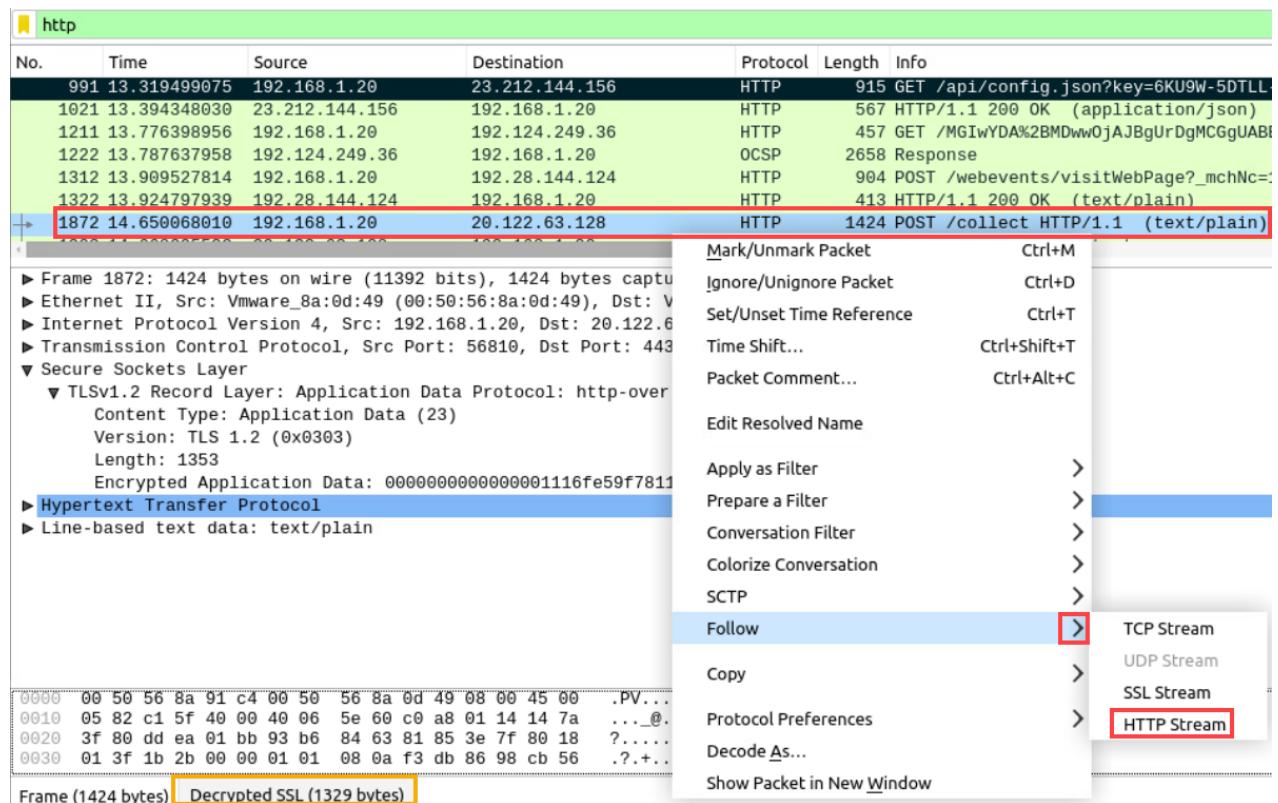
11. In the *Wireshark Open Capture File* window, view the **Desktop** directory, select **decrypted-capture.pcapng**, and click **Open**.



12. Type **http** into the *Wireshark* filter box and apply the filter.



13. Identify the first HTTP POST /collect packet. Right-click the packet and select **Follow > HTTP Stream** to view the decrypted HTTP stream.



14. Observe the HTTP Stream information that is provided in the stream. Once finished, close the HTTP Stream window.

Wireshark · Follow HTTP Stream (tcp.stream eq 35) · decrypted-capture

```

POST /collect HTTP/1.1
Host: p.clarity.ms
Connection: keep-alive
Content-Length: 855
Origin: https://www.paloaltonetworks.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/65.0.3325.181 Chrome/65.0.3325.181 Safari/537.36
Content-Type: text/plain; charset=UTF-8
Accept: /*
Referer: https://www.paloaltonetworks.com/services/education/academy
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

{"e": ["0.7.13", 1, 0, 229, "gvruf573nv", "1jrskvc", "aookus", 1, 0, 0], "a": [[8, 1265, 6967], [40, 11, 1265, 630], [40, 28, "visible"], [41, 33, 3, null, null, null, 0], [229, 34, "_uetvid"], ["4bf5f1a06eb811ee87a0f13a830cde77"], "_s", ["uet"], "_u", ["211019041"], "C_IS", ["0"], "_uemid", ["3bdb8f54-411e-4a1c-8ecb-73c0d9051949", "3bdb8f54-411e-4a1c-8ecb-73c0d9051949"]], [229, 1, 0, ["Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/65.0.3325.181 Chrome/65.0.3325.181 Safari/537.36"], 1, ["https://www.paloaltonetworks.com/services/education/academy"], 3, ["Academy Page - Palo Alto Networks"], 9, ["en-US"], 15, ["1vbbdio"], 16, ["en-US"], 19, ["website"], 20, ["Academy Page"], 22, ["Linux x86_64"], 26, ["1"], 27, ["4g"], 28, ["1387"], 29, ["2"]], [229, 0, 0, 1697835471719, 1, 0, 3, 30, 4, 42, 5, 2, 6, 33, 7, 1, 14, 1280, 15, 768, 16, 24, 25, 6, 26, 0, 31, 1, 32, 0, 33, 1, 34, 2, 35, 0], [229, 36, 6, [164, 60]]]}HTTP/1.1 204 No Content
Server: nginx/1.18.0 (Ubuntu)
Date: Fri, 20 Oct 2023 20:57:52 GMT
Connection: keep-alive
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://www.paloaltonetworks.com
Vary: Origin
Request-Context: appId=cid-v1:2f7711a9-b21e-4abe-a9d6-5b0ce5d18b64

POST /collect HTTP/1.1
Host: p.clarity.ms
Connection: keep-alive
Content-Length: 599742
Origin: https://www.paloaltonetworks.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/65.0.3325.181 Chrome/65.0.3325.181 Safari/537.36
5 client pkts, 5 server pkts, 9 turns.

Entire conversation (623 kB) Show and save data as ASCII
Find: Find Next
Help Filter Out This Stream Print Save as... Back Close

```

15. Notice when the window is exited, a filter is automatically pre-applied. Leave this filter applied as it will be different from each lab. Examine the SSL/TLS handshake between the Client and the server. Scroll up and identify the first **Client Hello** packet with the source *192.168.1.20* and select this packet. The source of this packet is the Client (*192.168.1.20*), and the destination is the server (*20.122.63.128*). This packet under the SSL layer tells the server that the Client is ready to establish an encrypted session.

tcp.stream eq 35						
No.	Time	Source	Destination	Protocol	Length	Info
1812	14.610380445	192.168.1.20	20.122.63.128	TCP	74	56810 → 443 [SYN] S
1817	14.626759315	20.122.63.128	192.168.1.20	TCP	74	443 → 56810 [SYN, A]
1818	14.626785968	192.168.1.20	20.122.63.128	TCP	66	56810 → 443 [ACK] S
1824	14.627903161	192.168.1.20	20.122.63.128	TLSv1.2	266	Client Hello
1855	14.644269173	20.122.63.128	192.168.1.20	TCP	66	443 → 56810 [ACK] S
1856	14.644276431	20.122.63.128	192.168.1.20	TLSv1.2	2962	Server Hello
1857	14.644282696	192.168.1.20	20.122.63.128	TCP	66	56810 → 443 [ACK] S
1858	14.644282740	20.122.63.128	192.168.1.20	TCP	1000	443 → 56810 [ACK] S

► Frame 1824: 266 bytes on wire (2128 bits), 266 bytes captured (2128 bits) on interface 0
► Ethernet II, Src: VMware_8a:0d:49 (00:50:56:8a:0d:49), Dst: VMware_8a:91:c4 (00:50:56:8a:91:c4)
► Internet Protocol Version 4, Src: 192.168.1.20, Dst: 20.122.63.128
► Transmission Control Protocol, Src Port: 56810, Dst Port: 443, Seq: 1, Ack: 1, Len: 200
▼ Secure Sockets Layer
 ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
 Content Type: Handshake (22)
 Version: TLS 1.0 (0x0301)
 Length: 195
 ► Handshake Protocol: Client Hello



The destination IP addresses and packet capture numbers in your packet capture may differ from these examples due to the nature of the lab environment.

16. Identify the corresponding **Server Hello** packet that shares the same IP address from the destination now set as the source. The source of this packet is the server (20.122.63.128), and the destination is the Client (192.168.1.20). In this packet, the server acknowledges the Client's request.

tcp.stream eq 35							
No.	Time	Source	Destination	Protocol	Length	Info	
1812	14.610380445	192.168.1.20	20.122.63.128	TCP	74	56810 → 443 [SYN]	
1817	14.626759315	20.122.63.128	192.168.1.20	TCP	74	443 → 56810 [SYN]	
1818	14.626785968	192.168.1.20	20.122.63.128	TCP	66	56810 → 443 [ACK]	
1824	14.627903161	192.168.1.20	20.122.63.128	TLSv1.2	266	Client Hello	
1855	14.644269173	20.122.63.128	192.168.1.20	TCP	66	443 → 56810 [ACK]	
1856	14.644276431	20.122.63.128	192.168.1.20	TLSv1.2	2962	Server Hello	
1857	14.644282696	192.168.1.20	20.122.63.128	TCP	66	56810 → 443 [ACK]	
1858	14.644299486	192.168.1.20	20.122.63.128	TCP	1266	443 → 56810 [PSH, ACK] Seq=2897 Ack=201 Win=65024 Len=0	

```

▶ Frame 1856: 2962 bytes on wire (23696 bits), 2962 bytes captured (23696 bits) on interface 0
▶ Ethernet II, Src: Vmware_8a:91:c4 (00:50:56:8a:91:c4), Dst: Vmware_8a:0d:49 (00:50:56:8a:0d:49)
▶ Internet Protocol Version 4, Src: 20.122.63.128, Dst: 192.168.1.20
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 56810, Seq: 1, Ack: 201, Len: 2896
▼ Secure Sockets Layer
  ▶ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 84
  ▶ Handshake Protocol: Server Hello

```

17. Next, identify the follow-up packet from the server, **Server Hello Done**. In this packet, the server tells the Client what version of TLS to use and then sends its public certificate, including its public encryption key, to the Client.

tcp.stream eq 35							
No.	Time	Source	Destination	Protocol	Length	Info	
1856	14.644276431	20.122.63.128	192.168.1.20	TLSv1.2	2962	Server Hello	
1857	14.644282696	192.168.1.20	20.122.63.128	TCP	66	56810 → 443 [ACK] Seq=201 Ack=2897 Win=35072 Len=0	
1858	14.644291742	20.122.63.128	192.168.1.20	TCP	1266	443 → 56810 [PSH, ACK] Seq=2897 Ack=201 Win=65024 Len=0	
1859	14.644299486	192.168.1.20	20.122.63.128	TCP	66	56810 → 443 [ACK] Seq=201 Ack=4097 Win=37888 Len=0	
1860	14.644884148	20.122.63.128	192.168.1.20	TLSv1.2	1303	Certificate, Server Key Exchange, Server Hello Done	
1861	14.644889644	192.168.1.20	20.122.63.128	TCP	66	56810 → 443 [ACK] Seq=201 Ack=5334 Win=40832 Len=0	
1867	14.647960467	192.168.1.20	20.122.63.128	TLSv1.2	159	Client Key Exchange, Change Cipher Spec, Finished	

```

▶ Frame 1860: 1303 bytes on wire (10424 bits), 1303 bytes captured (10424 bits) on interface 0
▶ Ethernet II, Src: Vmware_8a:91:c4 (00:50:56:8a:91:c4), Dst: Vmware_8a:0d:49 (00:50:56:8a:0d:49)
▶ Internet Protocol Version 4, Src: 20.122.63.128, Dst: 192.168.1.20
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 56810, Seq: 4097, Ack: 201, Len: 1237
  [3 Reassembled TCP Segments (4930 bytes): #1856(2807), #1858(1200), #1860(923)]
▼ Secure Sockets Layer
  ▶ TLSv1.2 Record Layer: Handshake Protocol: Certificate
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 4925
  ▶ Handshake Protocol: Certificate
▼ Secure Sockets Layer
  ▶ TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 300
  ▶ Handshake Protocol: Server Key Exchange
  ▶ TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done

```

18. Examine the corresponding **Client Key Exchange** packet. The source is the Client (192.168.1.20), and the destination is the server (20.122.63.128). When the Client receives the *Server Hello* packet, it first verifies that the server certificate is trusted. Then, the Client generates a session key using the Diffe-Hellman protocol. This session key will be used to encrypt and decrypt traffic from the server to the Client for the duration of the session. The Client securely sends this session key in this packet by encrypting the session key using the server's public key. When the server receives this packet, it will use its private key to decrypt and obtain the session key.

tcp.stream eq 35						
No.	Time	Source	Destination	Protocol	Length	Info
1856	14.644276431	20.122.63.128	192.168.1.20	TLSv1.2	2962	Server Hello
1857	14.644282696	192.168.1.20	20.122.63.128	TCP	66	56810 → 443 [ACK] Seq=201 Ack=2897 Win=35072 Len=0
1858	14.644291742	20.122.63.128	192.168.1.20	TCP	1266	443 → 56810 [PSH, ACK] Seq=2897 Ack=201 Win=65024
1859	14.644299486	192.168.1.20	20.122.63.128	TCP	66	56810 → 443 [ACK] Seq=201 Ack=4097 Win=37888 Len=0
1860	14.644884148	20.122.63.128	192.168.1.20	TLSv1.2	1303	Certificate, Server Key Exchange, Server Hello Done
1861	14.644889644	192.168.1.20	20.122.63.128	TCP	66	56810 → 443 [ACK] Seq=201 Ack=5334 Win=40832 Len=0
1867	14.647960467	192.168.1.20	20.122.63.128	TLSv1.2	159	Client Key Exchange, Change Cipher Spec, Finished

► Frame 1867: 159 bytes on wire (1272 bits), 159 bytes captured (1272 bits) on interface 0
 ► Ethernet II, Src: VMware_8a:0d:49 (00:50:56:8a:0d:49), Dst: VMware_8a:91:c4 (00:50:56:8a:91:c4)
 ► Internet Protocol Version 4, Src: 192.168.1.20, Dst: 20.122.63.128
 ► Transmission Control Protocol, Src Port: 56810, Dst Port: 443, Seq: 201, Ack: 5334, Len: 93

▼ Secure Sockets Layer
 ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
 Content Type: Handshake (22)
 Version: TLS 1.2 (0x0303)
 Length: 37
 ▶ Handshake Protocol: Client Key Exchange
 ▼ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
 Content Type: Change Cipher Spec (20)
 Version: TLS 1.2 (0x0303)
 Length: 1
 Change Cipher Spec Message
 ▼ TLSv1.2 Record Layer: Handshake Protocol: Finished
 Content Type: Handshake (22)
 Version: TLS 1.2 (0x0303)

19. The lab is now complete; you may end the reservation.