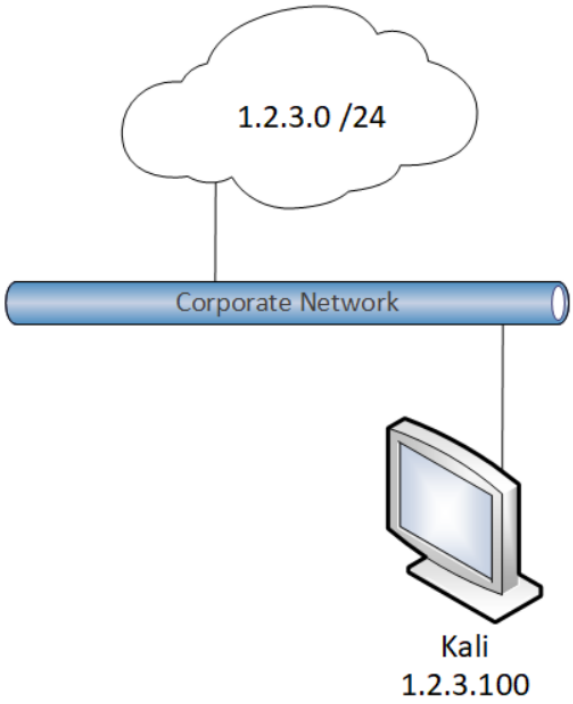
Pod Topology



Lab Settings

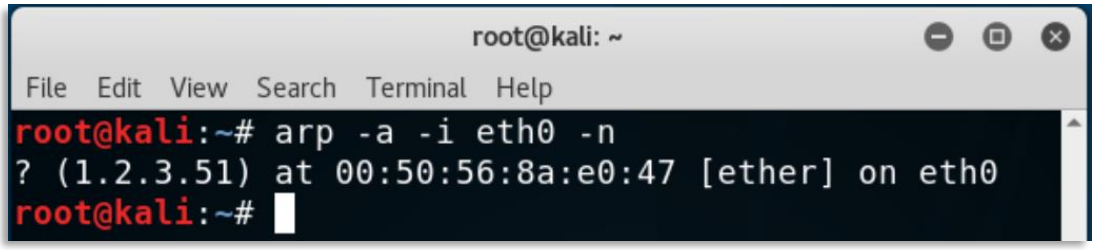
1. Click on the Kali tab or computer icon (see above).
2. Open a terminal window from the menu listed at the bottom of your screen.

ARP

1. One of the simplest ways to identify local network hosts is to review the ARP cache. This can be done by using the **`arp`** command:

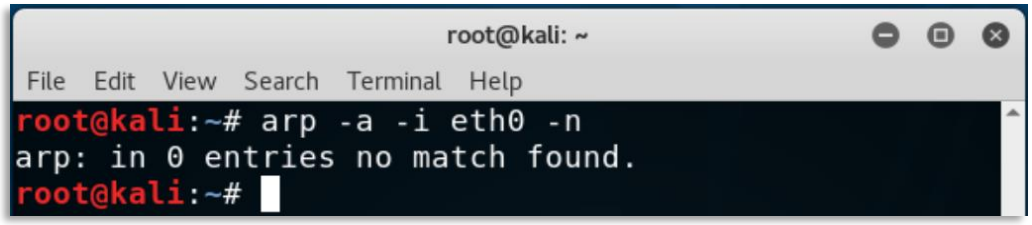
```
arp -a -i eth0 -n
```

The results of the command should look similar to the figure below.



Screenshot of the arp command showing one entry.

If you do not see any information displayed, then the ARP cache is simply not yet populated. If you do have entries, that is good too. To try again, just press the up arrow or retype the command, and press Enter. Either result is informative.



Screenshot of the arp command showing no entries.

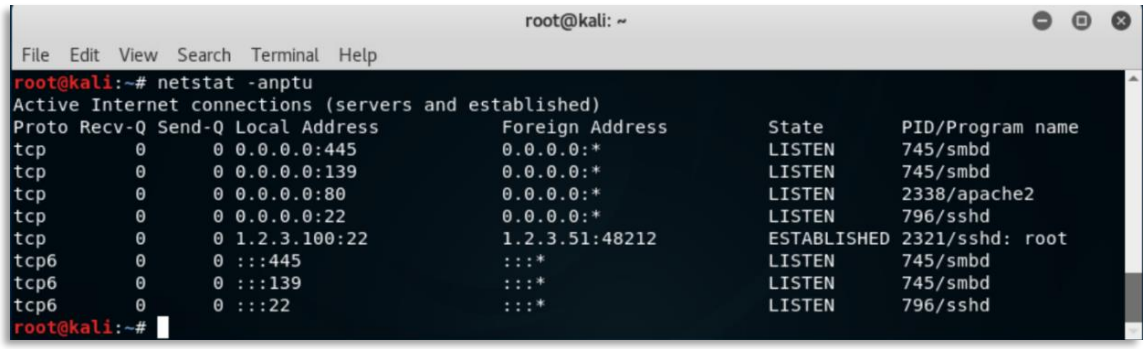
Netstat

1. The next reconnaissance method is used to identify both local and remote hosts, by looking at active network sessions using the **`netstat`** command. Netstat also shows the processes listening on a given port that help identify the network services running at the local system. The command options are described as follows:

   `-p` owning process ID, **`-a`** all sockets, **`-n`** no name resolution, **`-t`** tcp, **`-u`** udp.

   ```
   netstat –pantu
   ```

   The host you are exploring is your Kali Desktop virtual machine. It may be running services, which would appear in a LISTEN state as shown below. The results of this command should look similar.



Screenshot of current network connections using netstat –pantu command

   The example shows our localhost with a listening Web server on Port 80, SSH server on Port 22, and File Sharing services on Port 139 and 445. Some use IPv4 and some use both IPv4 and IPv6. Also take note of any ESTABLISHED connections that already exist. These are active connections between your local host and a remote host.

2. Try using different **`netstat`** command options to see how it affects the output.

(e.g., `netstat -pant`, `netstat -antu`).

.bash_history

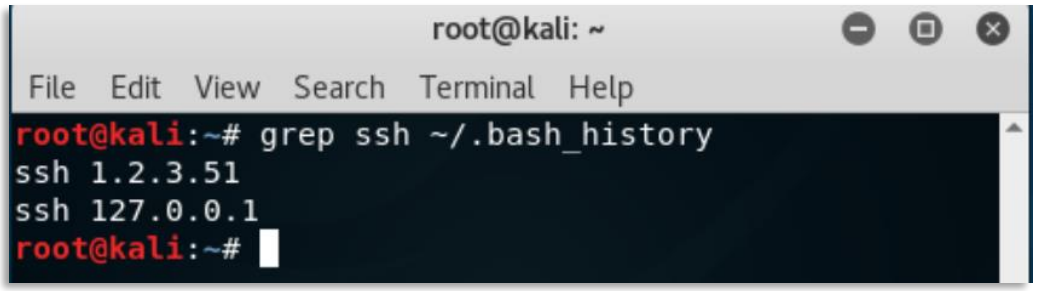
History and log files may contain information that can be used to identify additional hosts and networks accessible from the current system. This next example looks at the **bash** command history file .bash_history. This file contains a list of all the commands that have been executed in the **bash** command shell. By identifying commands associated with IP and hostname information, we learn about new hosts and networks that could be accessible from this system.

Hence, we are passively discovering and mapping the network as we search through this and other files on the system. The .bash_history file is located in each user's home directory, which is represented by the ~/.

1. To more efficiently search the .bash_history file use the **grep** command to search for keywords such as ssh, ftp, telnet, etc.

```
grep ssh ~/.bash_history
```

The results of this command should look similar to the figure below. By using the **grep** command, only the lines that contain the keyword ssh are shown, making it easy to identify the hosts likely to be running Linux with the SSH service running.



Search for ssh in the .bash_history file using grep

2. Try using the same command with a different keyword such as **ping**.

```
grep ping ~/.bash_history
```

If someone previously used a ping to look for another host, then you can discover what someone else once attempted to discover. You do not have to run the ping command. Stay passive by not sending pings, but you can at least investigate what has been looked for in the past.

3. Telnet is often considered questionable when in use since it is arguably insecure for using clear text transmissions. See if anyone has been using **telnet**.

```
grep telnet ~/.bash_history
```

If a result comes up more than once, it just means that the command was used in the past more than one time.

4. FTP connections are another example of past activity that we could look for.

```
grep ftp ~/.bash_history
```

Commands can also contain interesting information such as usernames. Did you find any? If so, list them below as part of your passive collection of information!

5. At this point, you may have taken an interest in the **1.2.3.0/24** IP network. Perhaps you would like to see all commands that included the text "**1.2.3.**" to attempt searching for related IP addresses. These are the first 3 octets matching that IP network. Find any command that may have used them and list them below.

```
grep 1.2.3 ~/.bash_history
```


6. The **.bash_history** file is not written with recent commands until the terminal (command shell) has been exited. Upon exit, the buffered commands are written to the **.bash_history** file. You can test this by exiting your current command shell.

7. Then start a new command shell and use the **history** command. You will now see all the commands you entered in the command shell before you exited.

```
history
```

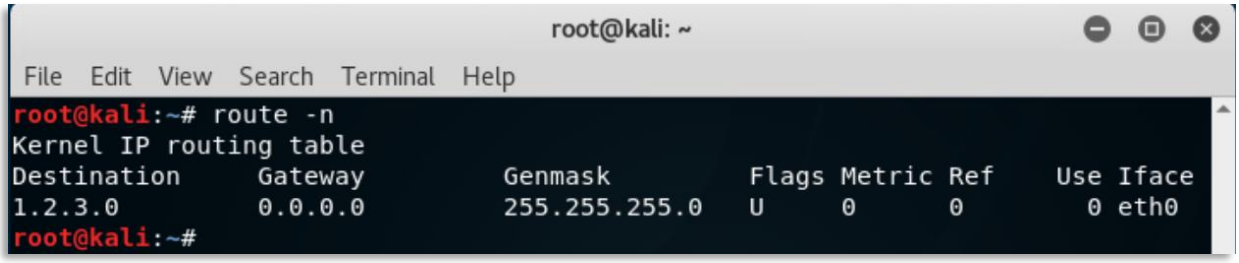
Routing Tables

Routing tables are another source for learning about the hosts and networks that are accessible from a system. This information can be used to identify new targets for attack and establish a map of the network. There are four main types of entries in the routing table, host routes, local and remote network routes, and the default route or gateway.

1. To view the routing table, use the **route** or **netstat** commands.

```
route –n
```

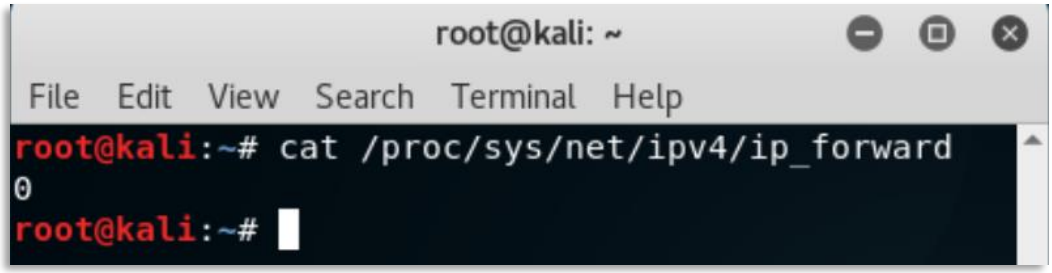
The results of this command should look similar to the figure below. The host you are using may not have a very interesting routing table. However, that in itself is useful information. If there are no gateways listed, then you just learned that this host is not setup to communicate with any outside networks.



Routing table shown by using the route –n command

2. If this system had two network interfaces, it may be possible to route traffic between the two networks. This can be determined by looking at the /proc/sys/net/ipv4/ip_forward configuration file with a text viewer or editor, where 1 means it is forwarding and 0 means it is not. You can also simply "**cat**" the contents to the terminal instead of opening it with a text editor. The file contains only a single character.

```
cat /proc/sys/net/ipv4/ip_forward
```



This information is useful for identifying rogue network forwarding or for launching a potential man-in-the-middle attack.

tcpdump

tcpdump is a tool that allows you to capture all traffic on the network interface, whether it is destined for your computer or not. However, because of switched network environments, normally you will only see traffic that was generated from your host or destined to your host.

1. You can watch packets as they arrive live, without saving them. Try watching for a few minutes and see if any traffic shows up. Packets will print to your terminal as they arrive.

   ```
   tcpdump -i eth0 -n
   ```

2. When you are done, press Ctrl-C to end tcpdump.
3. We can practice capturing some network traffic and save the captured packets to a file. Open a terminal window and ensure that you are in the Desktop directory. This will make it easier to find the file later.

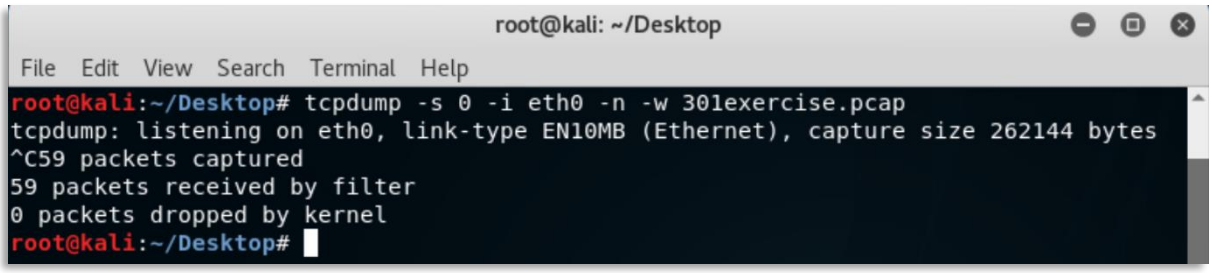
   ```
   cd ~/Desktop
   ```

4. To start the network traffic capture, type the following.

   ```
   tcpdump -i eth0 -n -w 301exercise.pcap
   ```

5. As before, let this capture run for a few minutes. You will not see anything printed, because the packets are being written to the file. Press Ctrl-C when you are done.

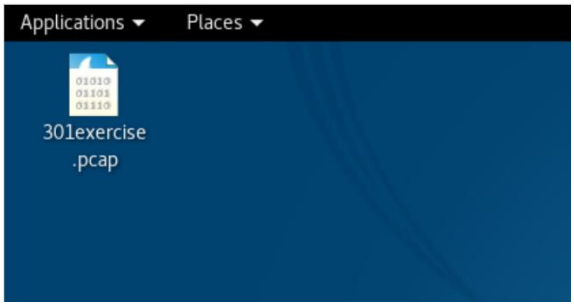
   When done, notice the capture size and number of packets captured.



This captured file will appear on your Desktop. The result is a simple packet capture with no specific filtering. The file can be stored anywhere for later review. The next section covers Wireshark. We can review this capture file, along with more interesting capture files, in the next section.

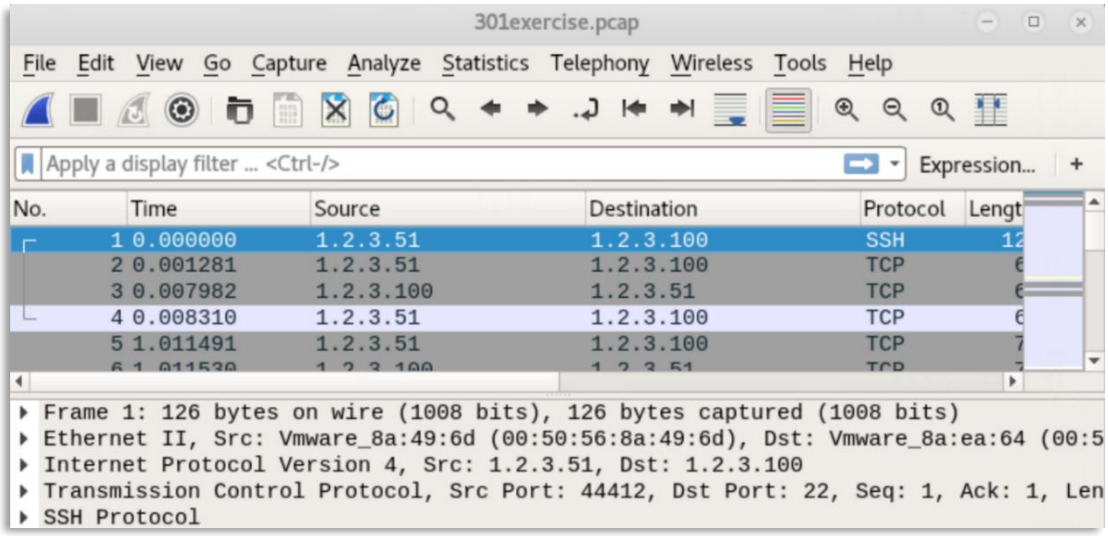
Wireshark

1. Use Wireshark to open and analyze the network traffic captured during the tcpdump. This can be done by double-clicking the file on the Desktop. This will automatically launch Wireshark with the selected pcap file.



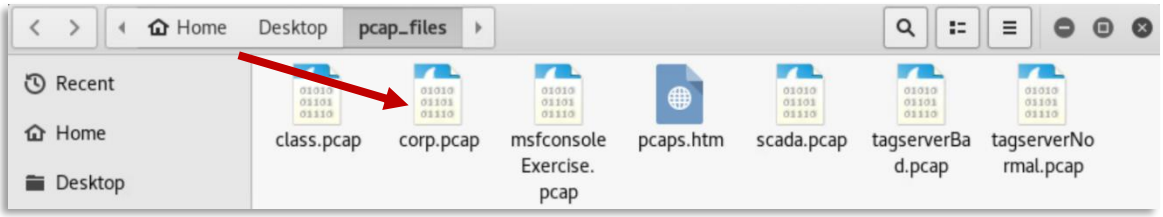
2. Because this lab is using the root user, Wireshark will give you a warning that it has full permission when run as root, or the superuser. For this lab, this is acceptable, and you can click OK. However, understand that in other situations, this may not be desired.

3. As the file loads, you will see Wireshark's main presentation. It shows you the packets found in the **capture that you made** and provides a way to explore it. You will only see minor traffic exchanged between yourself and other components of this lab. Scroll to explore the results.

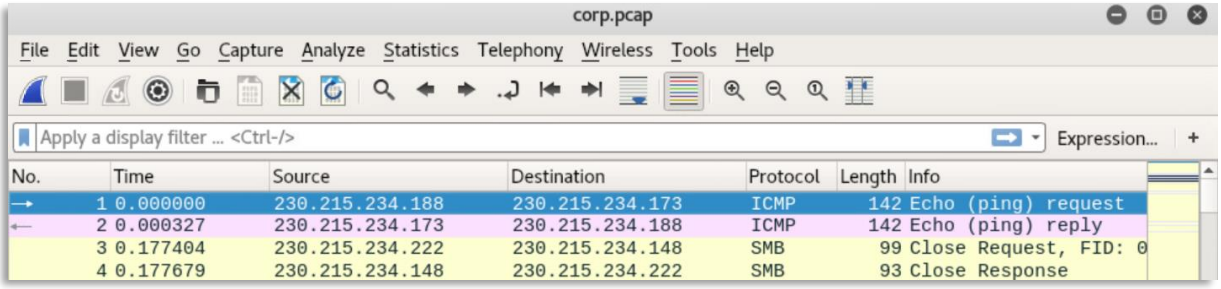


4. When you are finished analyzing your traffic, **close Wireshark**. Next, we will examine additional traffic capture files we have made for you.

5. Now that you have closed Wireshark, we will start over with a different .pcap file. On the Desktop, open the folder named **pcap_files**. Open **corp.pcap** by double-clicking it.

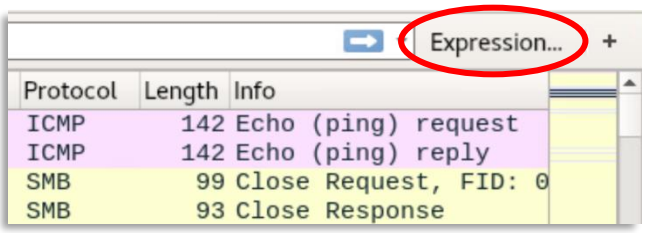


Wireshark will automatically load the file.



To help in the analysis of the traffic capture, we can generate display filters to show you the packets of interest. This is done by entering the display filter syntax directly into the filter box if you know the syntax, or by using the GUI-based expression builder that will generate the appropriate display filter syntax for you.

6. Next is an example of how to create a display filter for TCP Port 80 using the expression builder. First, click the ***Expression*** icon that is next to the Filter box as shown below.

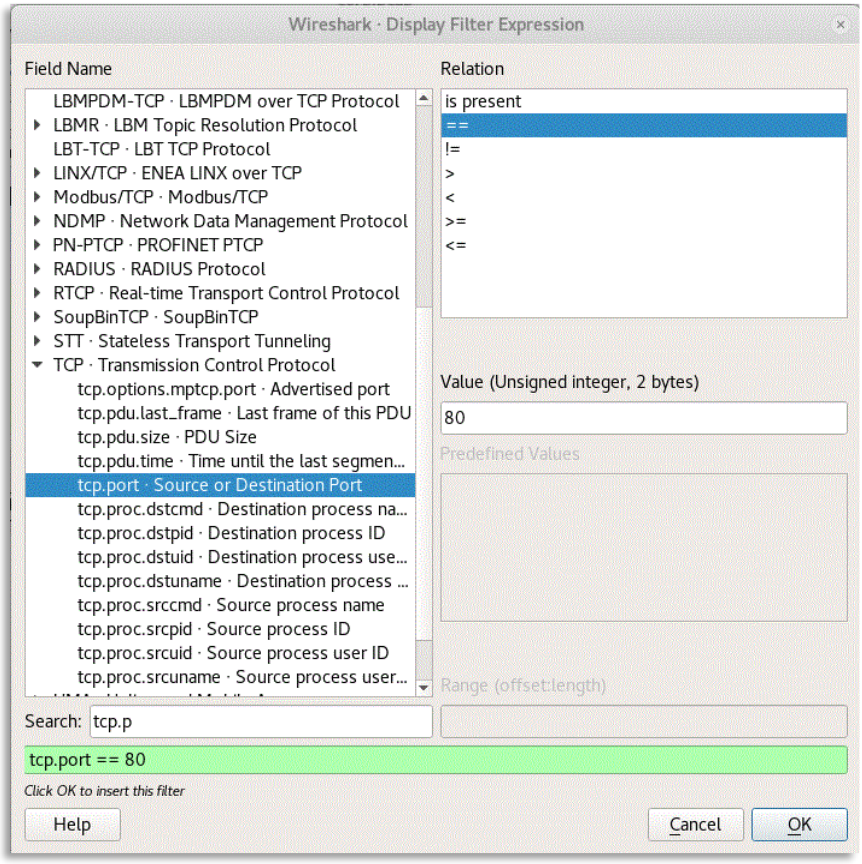


7. When the expression builder window appears, scroll down to the **TCP-Transmission Control Protocol** drop-down menu.

   NOTE: *This will be near the bottom of the selection window.*

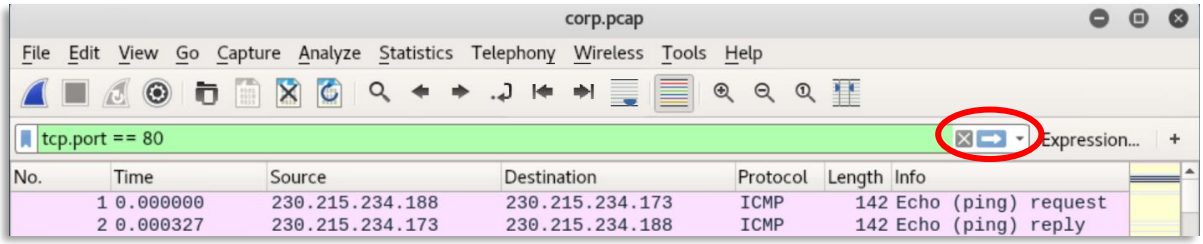
   Open the TCP drop-down menu and select "tcp.port" then select "==" from the Relation column, and finally type "80" in the Value box.

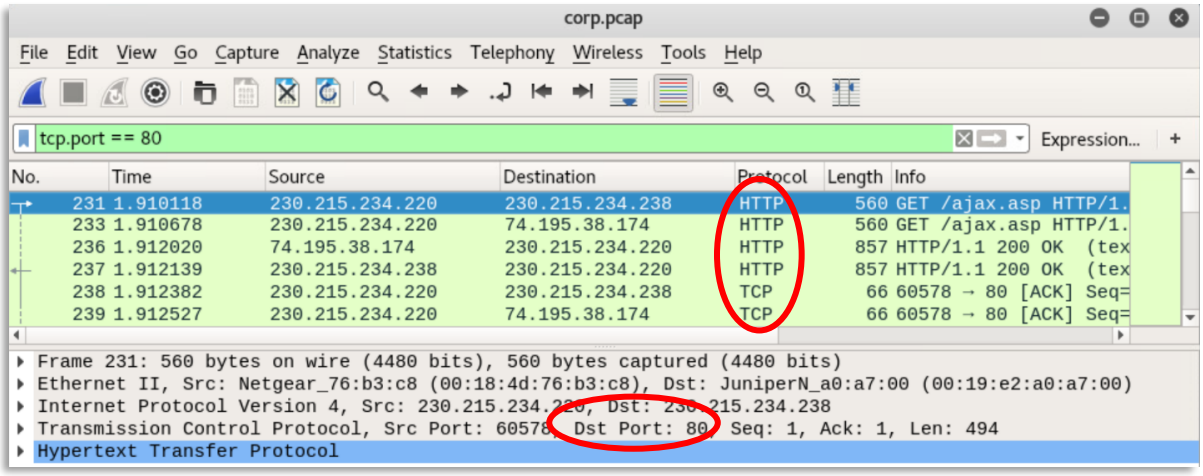
   When the field turns green, it means that the syntax is valid. Note that is now says "tcp.port == 80". This builder has now made an expression for you to use.



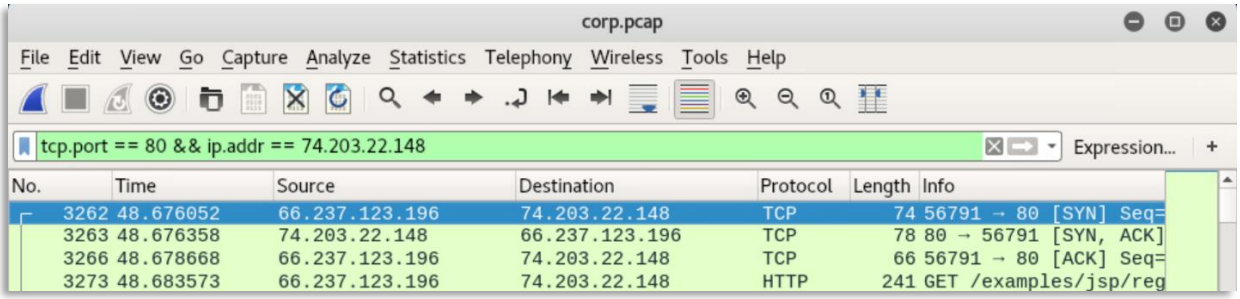
8. Finish by clicking OK. The builder utility will close. Observe that the expression has now been automatically written into the expression bar of the main window.

9. To activate the filter, press the Apply icon. Once the filter is applied, the only packets shown will have a source or destination of TCP port 80.



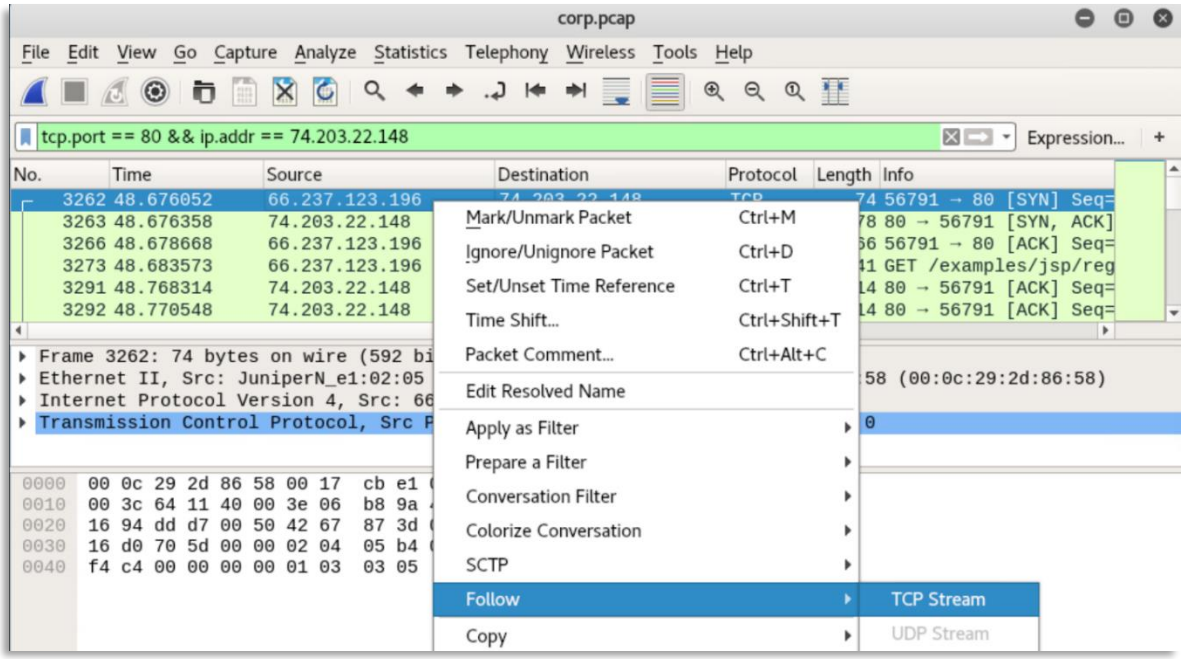


10. Continue by creating your own display filter combinations. You can use familiar boolean operators. Examples would be && and || to combine filters. For example, tcp.port == 80 && ip.addr == 74.203.22.148 is a combination of two filters.



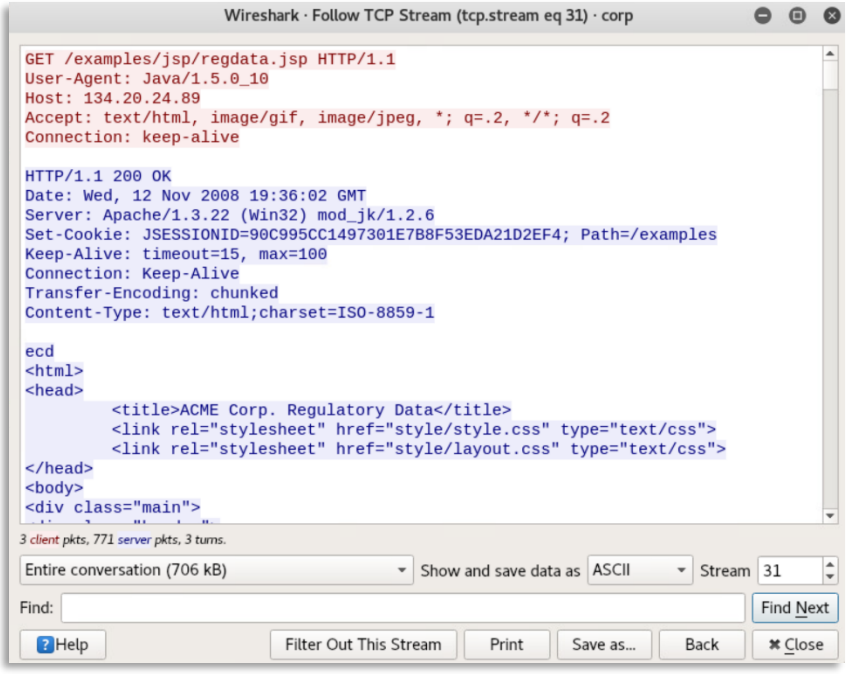
The new combined expression filters packets that match IP address 74.203.22.148 that communicate over TCP port 80.

11. These results show multiple TCP conversations from the same IP address. If you want to follow a single TCP session, right-click one of the packets and select ***Follow TCP Stream***. This will generate a display filter that will show you only that TCP stream.



This window shows the single TCP stream. All the HTTP messages within are now shown.

The cheat sheet on the next page is a reference for some of the basic Wireshark display filter syntax. Use this table or the expression builder to help you create your own display filter combinations.



Ethernet		
eth.addr	eth.len	eth.src
eth.dst	eth.lg	eth.trailer
eth.ig	eth.multicast	eth.type

IEEE 802.1Q		
vlan.cfi	vlan.id	vlan.priority
vlan.etype	vlan.len	vlan.trailer

IPv4	
ip.addr	ip.fragment.overlap.conflict
ip.checksum	ip.fragment.toolongfragment
ip.checksum_bad	ip.fragments
ip.checksum_good	ip.hdr_len
ip.dsfield	ip.host
ip.dsfield.ce	ip.id
ip.dsfield.dscp	ip.len
ip.dsfield.ect	ip.proto
ip.dst	ip.reassembled_in
ip.dst_host	ip.src
ip.flags	ip.src_host
ip.flags.df	ip.tos
ip.flags.mf	ip.tos.cost
ip.flags.rb	ip.tos.delay
ip.frag_offset	ip.tos.precedence
ip.fragment	ip.tos.reliability
ip.fragment.error	ip.tos.throughput
ip.fragment.multipletails	ip.ttl
ip.fragment.overlap	ip.version

IPv6	
ipv6.addr	ipv6.hop_opt
ipv6.class	ipv6.host
ipv6.dst	ipv6.mipv6_home_address
ipv6.dst_host	ipv6.mipv6_length
ipv6.dst_opt	ipv6.mipv6_type
ipv6.flow	ipv6.nxt
ipv6.fragment	ipv6.opt.pad1
ipv6.fragment.error	ipv6.opt.padn
ipv6.fragment.more	ipv6.plen
ipv6.fragment.multipletails	ipv6.reassembled_in
ipv6.fragment.offset	ipv6.routing_hdr
ipv6.fragment.overlap	ipv6.routing_hdr.addr
ipv6.fragment.overlap.conflict	ipv6.routing_hdr.left
ipv6.fragment.toolongfragment	ipv6.routing_hdr.type
ipv6.fragments	ipv6.src
ipv6.fragment.id	ipv6.src_host
ipv6.hlim	ipv6.version

ARP	
arp.dst.hw_mac	arp.proto.size
arp.dst.proto_ipv4	arp.proto.type
arp.hw.size	arp.src.hw_mac
arp.hw.type	arp.src.proto_ipv4
arp.opcode	

TCP	
tcp.ack	tcp.options.qs
tcp.checksum	tcp.options.sack
tcp.checksum_bad	tcp.options.sack_le
tcp.checksum_good	tcp.options.sack_perm
tcp.continuation_to	tcp.options.sack_re
tcp.dstport	tcp.options.time_stamp
tcp.flags	tcp.options.wscale
tcp.flags.ack	tcp.options.wscale_val
tcp.flags.cwr	tcp.pdu.last_frame
tcp.flags.ecn	tcp.pdu.size
tcp.flags.fin	tcp.pdu.time
tcp.flags.push	tcp.port
tcp.flags.reset	tcp.reassembled_in
tcp.flags.syn	tcp.segment
tcp.flags.urg	tcp.segment.error
tcp.hdr_len	tcp.segment.multipletails
tcp.len	tcp.segment.overlap
tcp.nxtseq	tcp.segment.overlap.conflict
tcp.options	tcp.segment.toolongfragment
tcp.options.cc	tcp.segments
tcp.options.ccecho	tcp.seq
tcp.options.ccnew	tcp.srcport
tcp.options.echo	tcp.time_delta
tcp.options.echo_reply	tcp.time_relative
tcp.options.md5	tcp.urgent_pointer
tcp.options.mss	tcp.window_size
tcp.options.mss_val	

UDP		
udp.checksum	udp.dstport	udp.srcport
udp.checksum_bad	udp.length	
udp.checksum_good	udp.port	

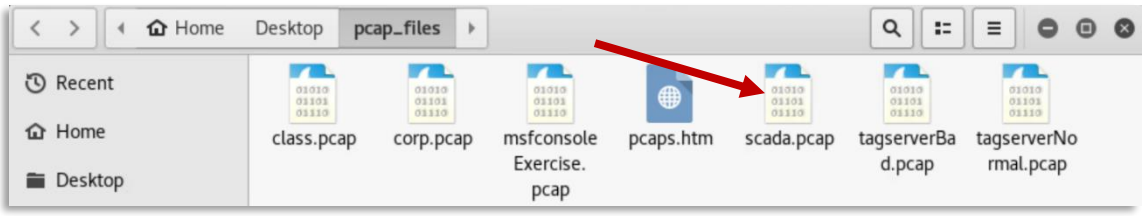
Operators		Logic	
eq	or ==	and	or && Logical AND
ne	or !=	or	or || Logical OR
gt	or >	xor	or ^^ Logical XOR
lt	or <	not	or ! Logical NOT
ge	or >=	[n] […]	Substring operator
le	or <=		

Ref: http://packetlife.net/library/cheat-sheets/

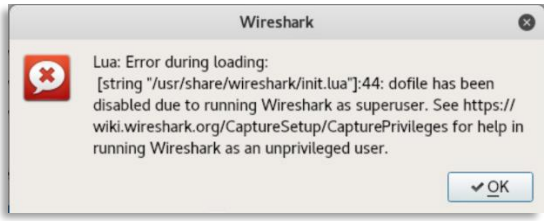
Wireshark Continued - Displaying Packets with DNP3

Wireshark also has protocol support for a number of ICS related traffic.

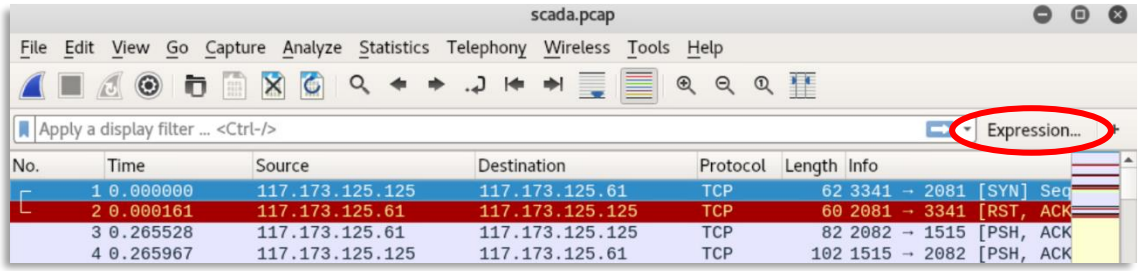
1. Close the corp.pcap file and go back to open the **scada.pcap** file. The same approach can be taken. However, this time we can build an expression to look specifically for the ICS protocol, DNP3.



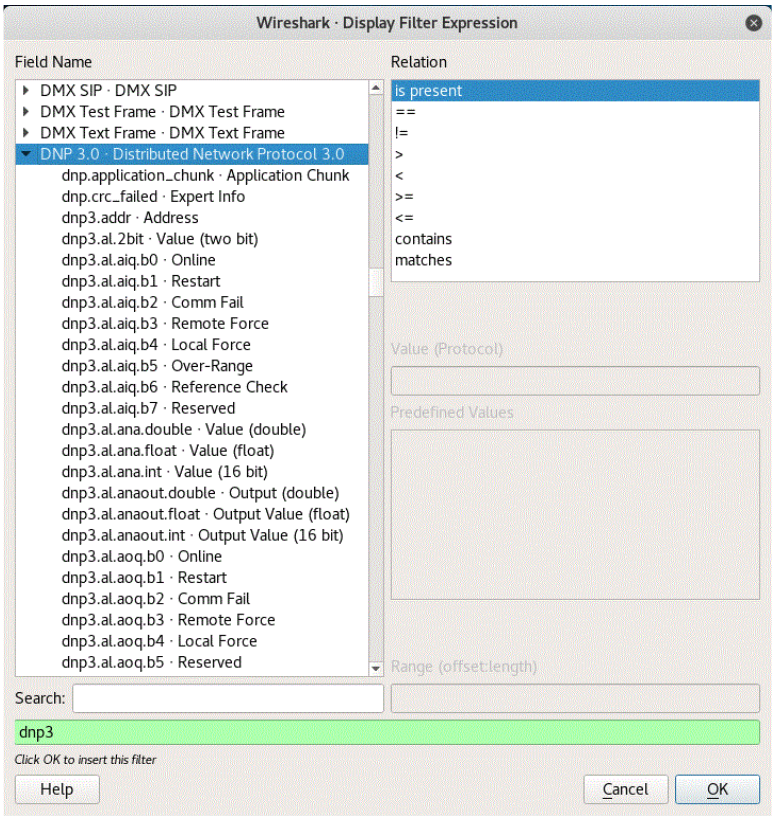
2. Click OK for the superuser privileges warning.



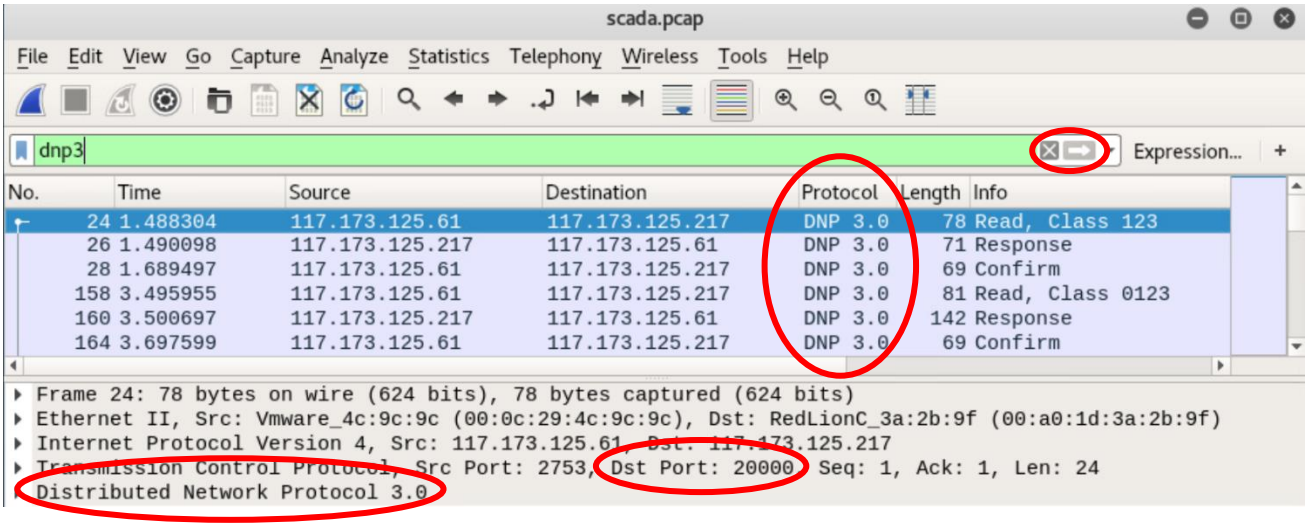
3. Wireshark will automatically load the file. Proceed by clicking on the Expression… icon.



4. A new pop-up window will appear. Scroll down and expand the DNP 3.0 option. A list of DNP3 protocol fields will appear.
5. For this example, we do not need to select any of these. We can just use the overall DNP3 selection.
6. In the Relation window, choose “is present”.
7. There is no value to fill this time. Your expression is now ready. Click **OK** to finish using the builder. You will be returned to the main Wireshark display.
8. Click on the ***Apply*** icon.



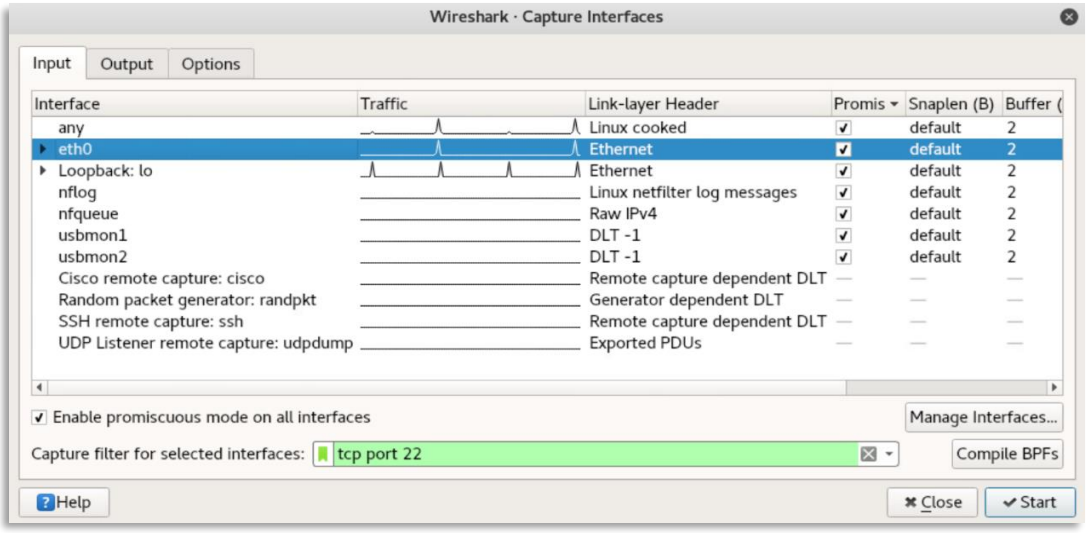
Wireshark will refresh the packets to match your filter expression. The results are all DNP 3 traffic that resides in this capture.



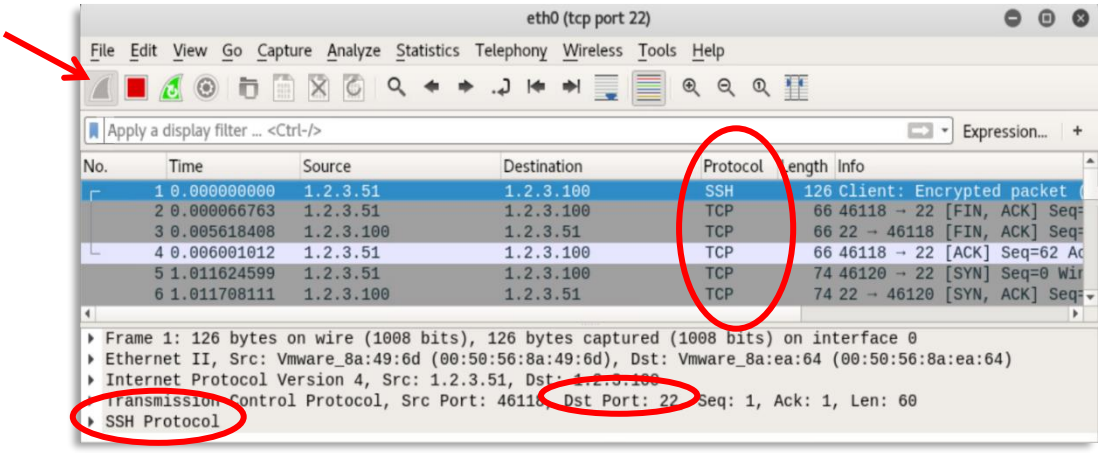
Wireshark Continued - Using a Capture Filter

The final exercise for Wireshark is to create a network traffic capture (pcap) using BPFs to filter the traffic. This is much like the concept of using tcpdump.

1. In a new Wireshark instance, select ***Capture > Options*** on the Wireshark menu bar. This will bring up a Capture Interfaces menu. Wireshark has listed available interfaces to capture from.
2. To create a simple capture filter for SSH traffic, select ***eth0*** from the list.



3. Type **tcp port 22** into the ***Capture Filter*** box as shown above. Instead of capturing every possible packet, this will only capture traffic using TCP port 22 which would commonly be SSH traffic.
4. Cancel the capture anytime, by clicking the **red** stop button.



5. Review the captured results. You will see only packets that are TCP 22, even though no filter is in your expression bar. The Capture itself, already filtered the desired packet type.

Post-exercise analysis

- What network protocols did you find?

_____
_____

- What ICS-specific protocols did you find?

_____
_____

- Were there plain text protocols?

_____
_____

Post-exercise analysis

- What network protocols did you find?

_____
_____

- What ICS-specific protocols did you find?

_____
_____

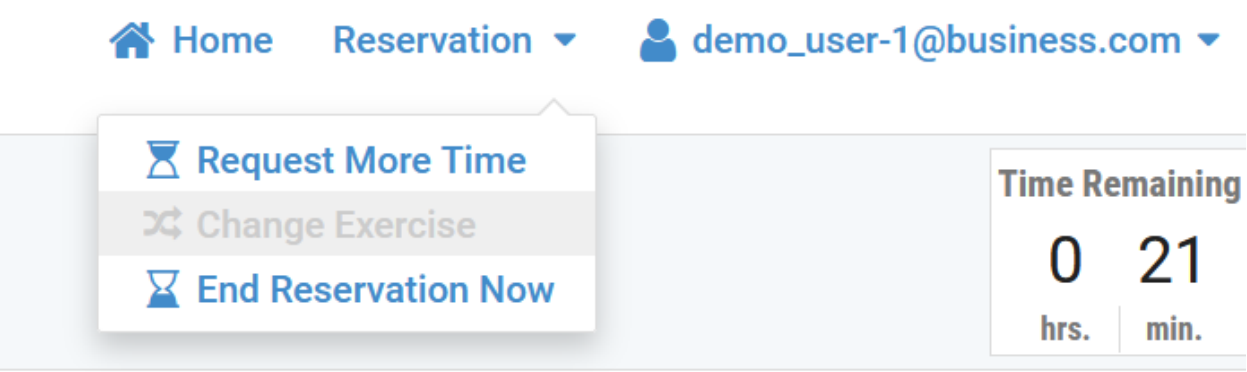
- Were there plain text protocols?

_____
_____

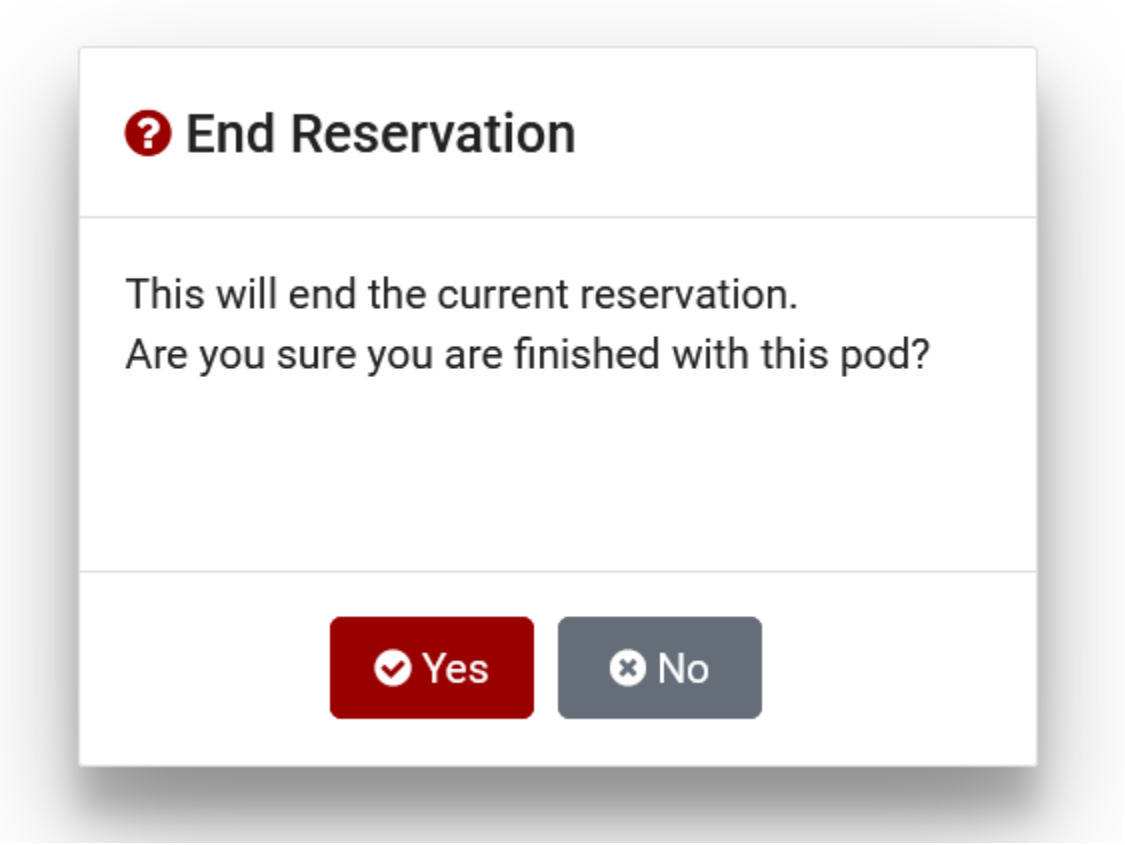
When you have finished recording the required data in your student guide, to close Netlab, go to the upper right corner of the browser window.

Click the Reservation drop-down.

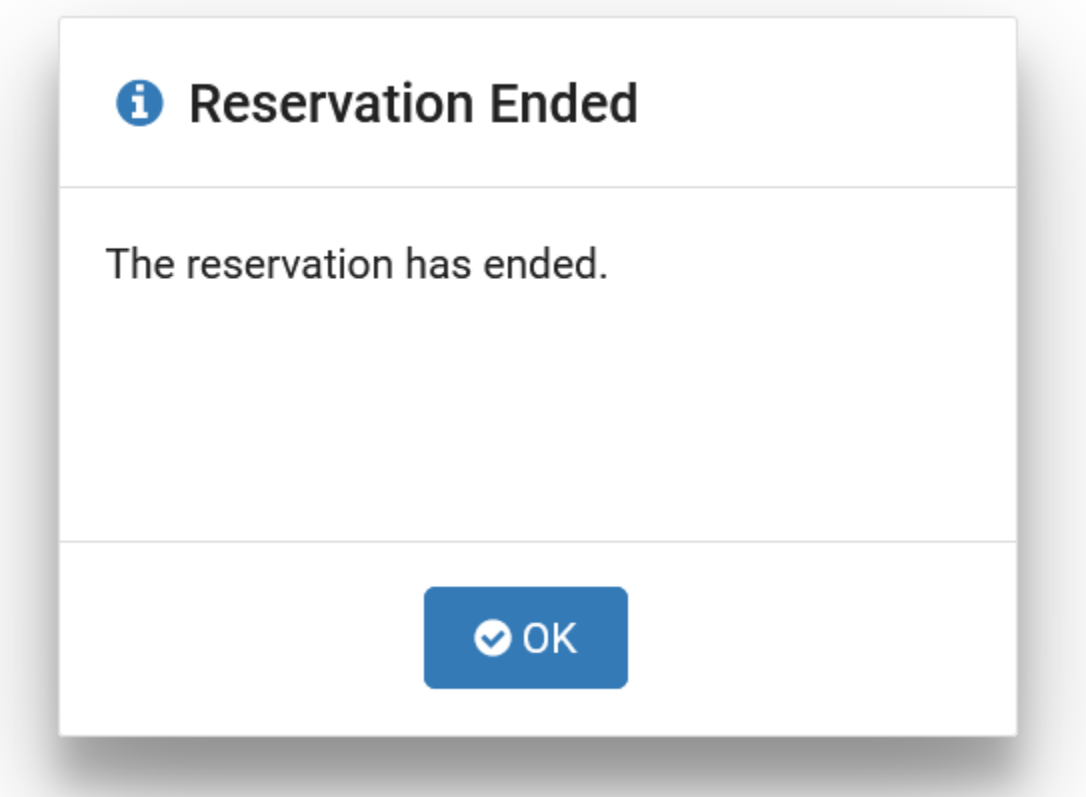
_____
_____
_____
_____



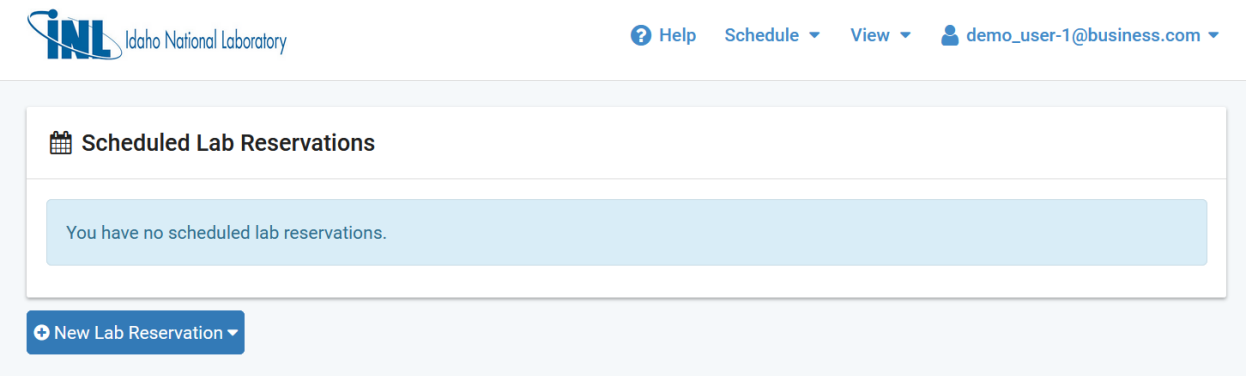
Click “End Reservation Now”



A warning will display. Click “Yes”.



A notification that the Reservation has ended will appear. Click “OK”.



You will be returned to the Netlab homepage. You can now exit from this tab or window. Be careful not to close your 301V training tab in the CISA VLP.