Introduction

Network forensics refers to the investigation of network traffic patterns and data captured in transit between computing devices. Network forensic techniques can provide insight into the source and scope of an attack, as well as supplement investigations of information left behind by intruders. By investigating the causes of a breach, and using that knowledge to build stronger security controls, organizations can work to prevent or mitigate future breaches. With the recent changes in the world, and the massive shift to remote working, the importance of network forensics has increased dramatically, as the rise of breaches, ransomware, and other incidents has grown exponentially.

Network forensics has two general purposes. The first consists of monitoring a network for anomalous traffic and identifying intrusions, typically as part of a broader security program. The second consists of analyzing captured network traffic with the objective of reassembling files and parsing communication streams, typically as part of a law enforcement investigation. For both purposes, one of the principal techniques is packet analysis, which involves using a packet capture utility (also called a sniffer) to intercept and record live network traffic, then reviewing the captured packets for evidence. This form of analysis is similar to conducting investigations on drive images and other forms of acquired evidence that have been preserved in a static format. Network forensics is also commonly conducted directly on live networking devices, including routers, switches, and firewalls, due to the fact that shutting down these devices and attempting to image them can potentially result in lost evidence.

In this lab, you will learn to capture and analyze network traffic using Wireshark, a popular sniffer and packet analysis utility. You will also learn to access and retrieve evidence directly from both a live router and a live firewall.

Lab Overview

SECTION 1 of this lab has two parts, which should be completed in the order specified.

1. In the first part of the lab, you will use Wireshark to capture and analyze network traffic.

2. In the second part of the lab, you will use basic router commands to gather information about the device itself and the broader network.

SECTION 2 of this lab allows you to apply what you learned in **SECTION 1** with less guidance and different deliverables, as well as some expanded tasks and alternative methods. You will take a deeper dive into packet analysis and examine firewall logs.

Finally, you will explore the virtual environment on your own in **SECTION 3** of this lab to answer a set of questions and challenges that allow you to use the skills you learned in the lab to conduct independent, unguided work - similar to what you will encounter in a real-world situation.

Learning Objectives

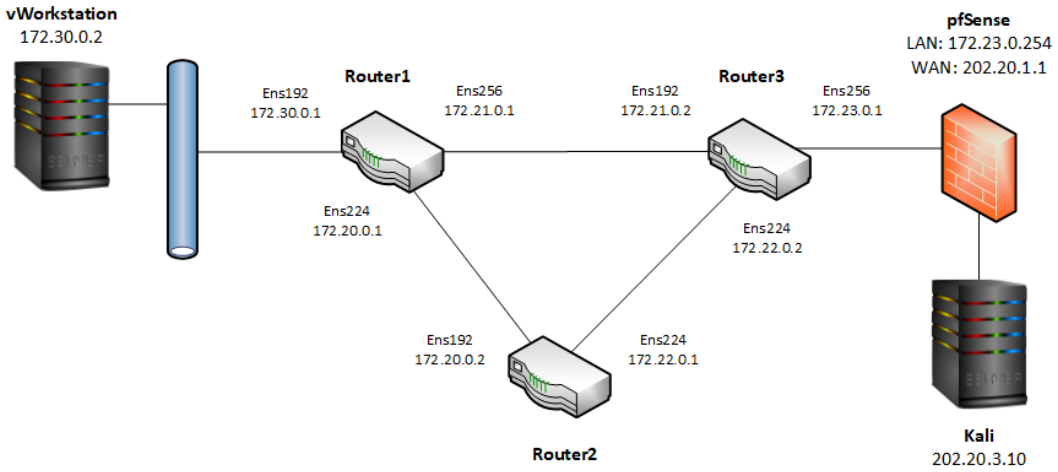
Upon completing this lab, you will be able to:

1. Perform packet capture using Wireshark.
2. Perform packet analysis using Wireshark.
3. Analyze routers for forensic evidence.
4. Examine firewall logs for forensic evidence.
5. Identify suspicious network traffic.

Topology

This lab contains the following virtual machines. Please refer to the network topology diagram below.

- vWorkstation (Windows: Server 2019)
- router1 (Linux: Ubuntu 16)
- router2 (Linux: Ubuntu 16)
- router3 (Linux: Ubuntu 16)
- pfSense (FreeBSD: pfSense 2.4)
- AttackLinux01 (Linux: Kali)



Tools and Software

The following software and/or utilities are required to complete this lab. Students are encouraged to explore the Internet to learn more about the products and tools used in this lab.

- Wireshark
- PuTTY
- Quagga
- pfSense

Deliverables

Upon completion of this lab, you are required to provide the following deliverables to your instructor:

SECTION 1

1. Lab Report file, including screen captures of the following:

- Timestamp-sorted traffic
- IP-filtered traffic

- Port-filtered traffic
- TCP push flag-filtered traffic
- Http-filtered traffic
- Router’s version output
- Router’s interface details
- Router's ARP table
- IP routing table
- Currently running configuration

2. Any additional information as directed by the lab:

- None

SECTION 2

1. Lab Report file, including screen captures of the following:

- Successful transfer of the secureTopo.png file
- Passive port specified by the FTP server in the Packet Details pane
- Time to live field in the Packet Details pane
- Follow TCP stream window
- Reconstituted PNG file
- Entries in the firewall log
- Resolved entries in the firewall log

2. Any additional information as directed by the lab:

- None

SECTION 3

1. Lab Report file, including screen captures of the following:

- Non-RIP route that you discovered on the target router.

2. Any additional information as directed by the lab:

- Record the destination IP address and Port number of the outgoing connection attempt.

Section 1: Hands-On Demonstration

Note: In this section of the lab, you will follow a step-by-step walk-through of the objectives for this lab to produce the expected deliverables.

1. **Review** the **Tutorial**.

   Frequently performed tasks, such as making screen captures and downloading your Lab Report, are explained in the Cloud Lab Tutorial. The Cloud Lab Tutorial is available from the User menu in the upper-right corner of the Student Dashboard. You should review these tasks before starting the lab.

2. **Proceed** with **Part 1**.

Part 1: Perform Packet Capture and Analysis

Note: Wireshark is a popular, free, and open-source packet collector and analyzer used for network troubleshooting and analysis. As packets are captured, the Wireshark interface allows users to inspect traffic at each layer of the TCP/IP stack. TCP, UDP, IP and Ethernet header information can be filtered and explored, providing an in-depth view of how protocols function and interact with each other. Since it is open-source, it is not only free to use but also can be further developed and improved for use with new network applications. This makes Wireshark very versatile and, therefore, has led to its adoption by many different IT professionals across a variety of IT sectors. Network administrators, software developers, penetration testers, and even hackers use Wireshark to inspect and analyze network traffic. With practice, Wireshark can be an invaluable tool for forensic investigations.

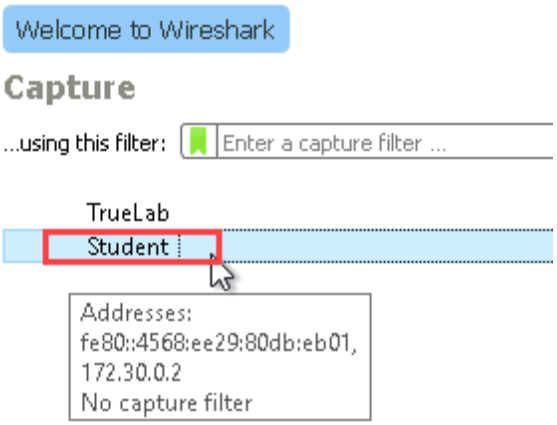
In this part of the lab, you will use Wireshark to capture live network traffic, then review an existing packet capture file and perform basic packet analysis. In the next steps, you will launch the Wireshark application and generate live traffic.

1. On the vWorkstation desktop, **double-click** the **Wireshark icon** (a blue shark fin) to launch the Wireshark application.

Wireshark icon

Note: The main screen of Wireshark includes details about the current capture configuration. From this screen, analysts can select common filters from the drop-down menu, or type a custom filter command to quickly sort the captured data.

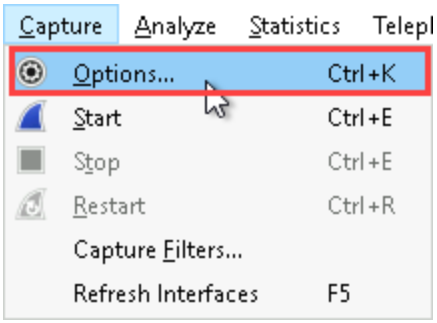
2. In the Capture section of the Wireshark window, **click** the **Student interface** to select it as your capture interface.



Student interface

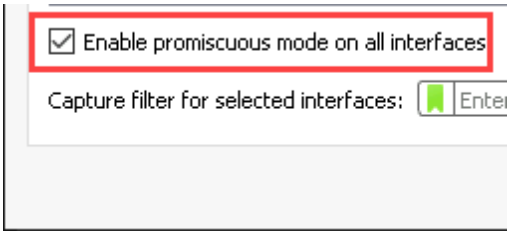
Note: If you were running Wireshark on your local computer, it is possible that you will see many more interfaces. It is also possible that some interfaces you expect to see might not appear on the list at all. If you know that a logical or physical interface exists, but it does not show up on the list, check the installation of winpcap and troubleshoot accordingly. Often, it is necessary to reinstall or update the Network Interface Card (NIC) drivers.

3. From the Wireshark menu bar, **click** the **Capture menu** and **select Options** to open the Capture Interfaces window.



Capture menu

4. In the Capture Interfaces window, **confirm** that the ***Enable promiscuous mode on all interfaces*** **checkbox** is **checked**.

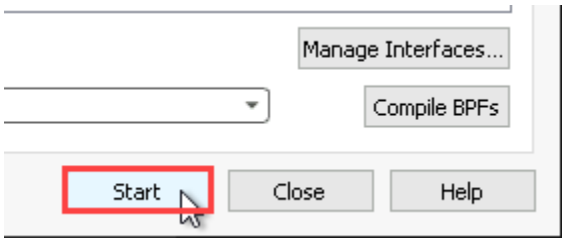


Enable promiscuous mode

Note: Promiscuous mode enables Wireshark to capture packets destined to any host on the same subnet or virtual LAN (VLAN) within this lab environment. Without this option selected, Wireshark would capture only packets to and from vWorkstation.

In a real wired network environment using a switch, enabling promiscuous mode in Wireshark would only enable you to capture broadcast and multicast packets on the network segment, in addition to traffic directed to and from the system running the packet collector. In this lab environment, promiscuous mode has been enabled on the underlying virtual switch, which broadly enables you to capture all traffic on the network segment, regardless of its destination.

5. In the Capture Interfaces window, **click** the **Start button** to begin capturing network traffic from the selected interface.



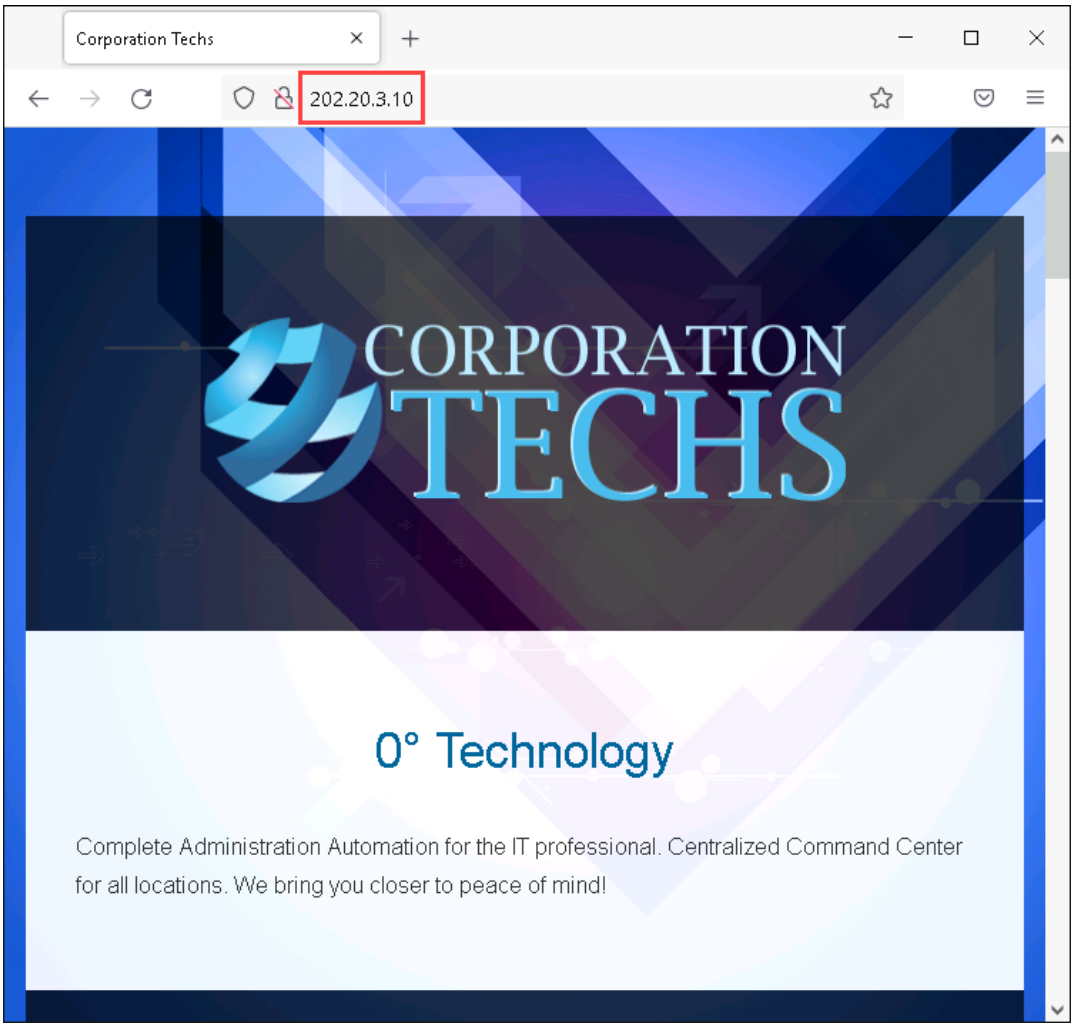
Start button

Note: Wireshark will close the Capture Interfaces window and display the main capture interface, which will begin to populate with captured packets.

6. On the vWorkstation taskbar, **click** the **Firefox icon** to open a new browser window.

Firefox icon

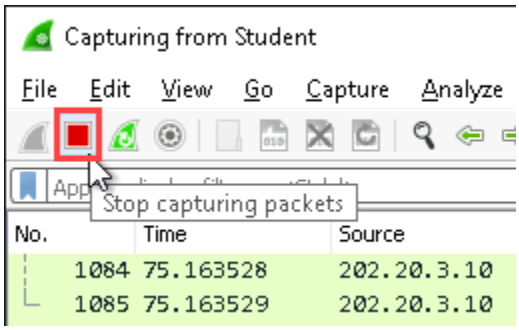
7. In the Firefox navigation bar, **type** `202.20.3.10` and **press Enter** to open a sample webpage on the Kali system located on the external network.



Sample webpage

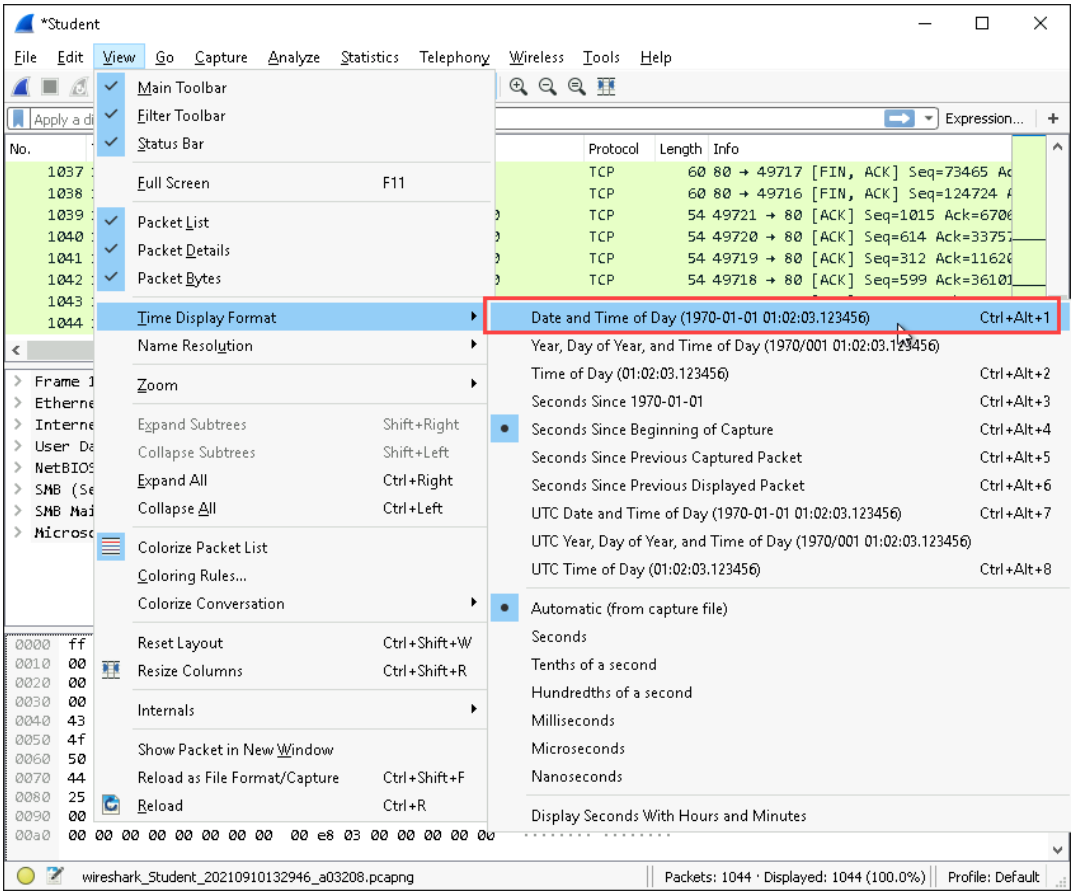
8. **Close** the **Firefox window**.

9. On the Wireshark toolbar, **click** the **Stop capture button** to stop the packet capture process.



Stop capture button

10. On the Wireshark menu bar, **click** the **View menu**, then **select Time Display Format > Date and Time of Day** to sort the captured packets according to their timestamp.



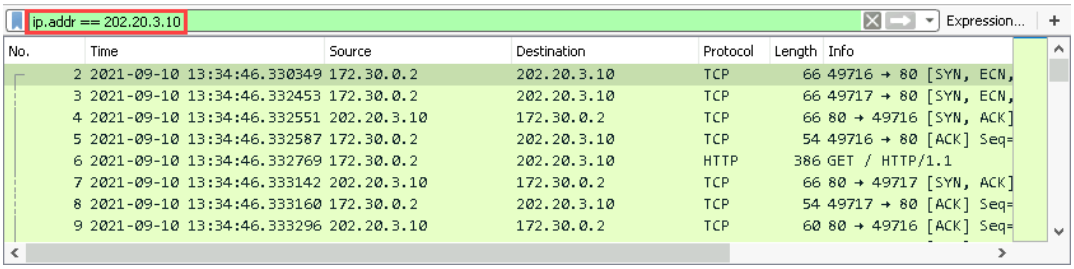
Time Display Format

Note: As a forensic investigator, you can use this feature to quickly establish the timeline for the captured packets. In some cases, traditional date and time will suffice, but for other investigations it may be more appropriate to select Seconds Since Beginning of Capture or another choice when a very specific time threshold is necessary.

11. **Make a screen capture** showing the **timestamp-sorted traffic**.

Note: Wireshark also offers a wide array of useful filters that allow you to drill down to specific information you are looking for. In the next step, you will apply the IP address filter which, as the name suggests, allows you to filter by specific IP address. Alongside MAC addresses, IP addresses are the principal identifiers for hosts in the TCP/IP networking model. When conducting a forensic investigation, isolating traffic to and from specific hosts is often a critical first step towards locating relevant evidence.

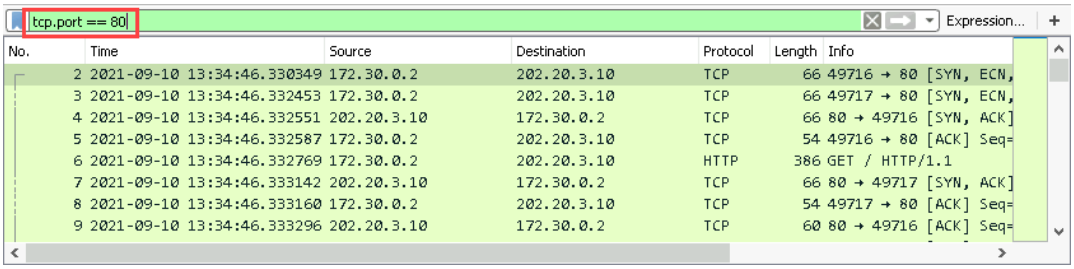
12. On the Wireshark toolbar, **type** `ip.addr == 202.20.3.10` in the Display Filter field and **press Enter** to display only traffic to and from the web server you connected to earlier.



IP address filter

13. **Make a screen capture** showing the **IP-filtered traffic**.

14. On the Wireshark toolbar, **highlight** the **existing text** in the Display Filter field, then **type** `tcp.port == 80` and **press Enter** to display only traffic to and from port 80.



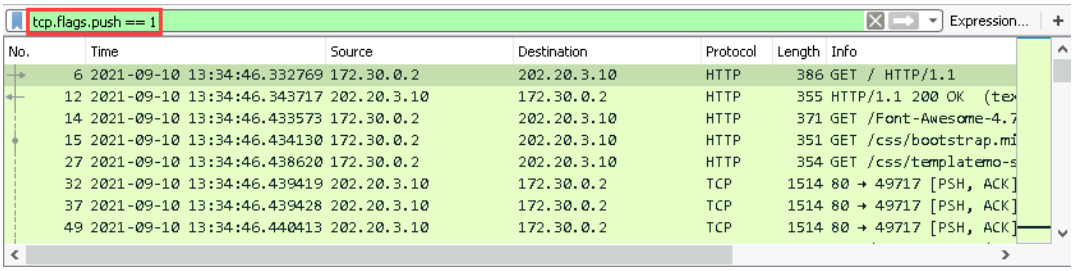
TCP port filter

Note: As a forensic investigator, searching by specific ports will also be a regular occurrence. Specific port numbers are typically associated with types of traffic (web, email, chat, etc.), and consequently provide an efficient means of isolating the specific types of traffic that you are investigating. Similarly, if you are investigating an intrusion, the application used by the intruder will likely have a specific port associated with it – for example, Metasploit defaults to port 4444 for its listeners.

In this case, you are effectively filtering for HTTP traffic, which defaults to port 80.

15. **Make a screen capture** showing the **port-filtered traffic**.

16. On the Wireshark toolbar, **highlight** the **existing text** in the Display Filter field, then **type** `tcp.flags.push == 1` and **press Enter** to display only traffic with the TCP push flag.

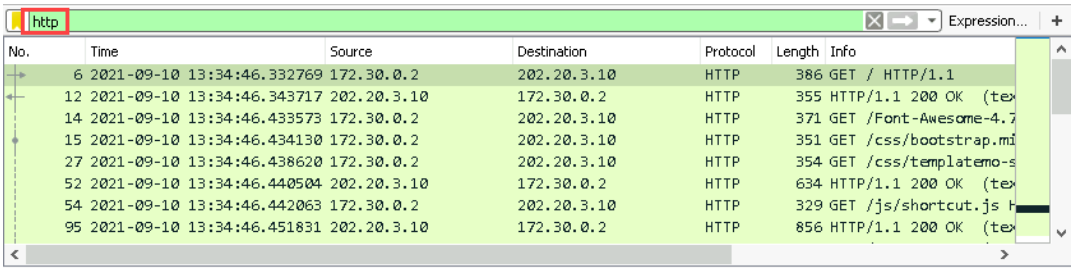


TCP push filter

Note: TCP flags are control bits that determine how TCP packets should be handled by the client. TCP Flags are some combination of ACK, SYN, FIN, PSH, RST, URG, ECE, CWR, or NS. Within your filtered traffic, you should notice the SYN (synchronization) flag, which is used as a first step in establishing a three-way handshake between two hosts, and the ACK (acknowledge) flag, which is used to confirm the connection, and PSH, which means that data has been sent.

17. **Make a screen capture** showing the **TCP push flag-filtered traffic**.

18. On the Wireshark toolbar, **highlight** the **existing text** in the Display Filter field, then **type** `http` and **press Enter** to display only traffic associated with the HTTP protocol.



HTTP filter

Note: Similarly, you can search for other protocols by name, such as ftp or tcp, to display only packets that use that protocol. In this case, the filter is functionally equivalent to your previous port 80 filter.

19. **Make a screen capture** showing the **http-filtered traffic**.

20. **Close** the **Wireshark window**.

21. When prompted, **click Quit without saving** to discard your current capture file and exit the Wireshark application.

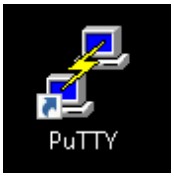
Part 2: Analyze a Router for Forensic Evidence

Note: In this part of the lab, you will access one of the live routers in the lab environment. Routers are specialized computer systems that are responsible for directing the flow of traffic within and across networks. Alongside switches and (less commonly) hubs, routers are the fundamental building blocks of modern computer networks, which makes them a valuable source of forensic evidence, especially when investigating a network-based attack.

When analyzing a router as part of a forensic investigation, you will typically need to connect directly to the router using a terminal utility. Network operating systems like Cisco IOS and other dedicated routing programs offer multiple commands for retrieving and manipulating information on the router. Although these commands are typically used for configuring and troubleshooting the router, they can also be used to retrieve evidence for forensic investigations.

For the purposes of this lab, you will be interacting with Quagga, an open-source Cisco-like routing suite. While the commands used in this lab will be specific to Quagga, they all have near-identical functional equivalents in Cisco IOS and other popular routing programs. In the next steps, you will open a terminal connection to a Linux system running the Quagga routing suite.

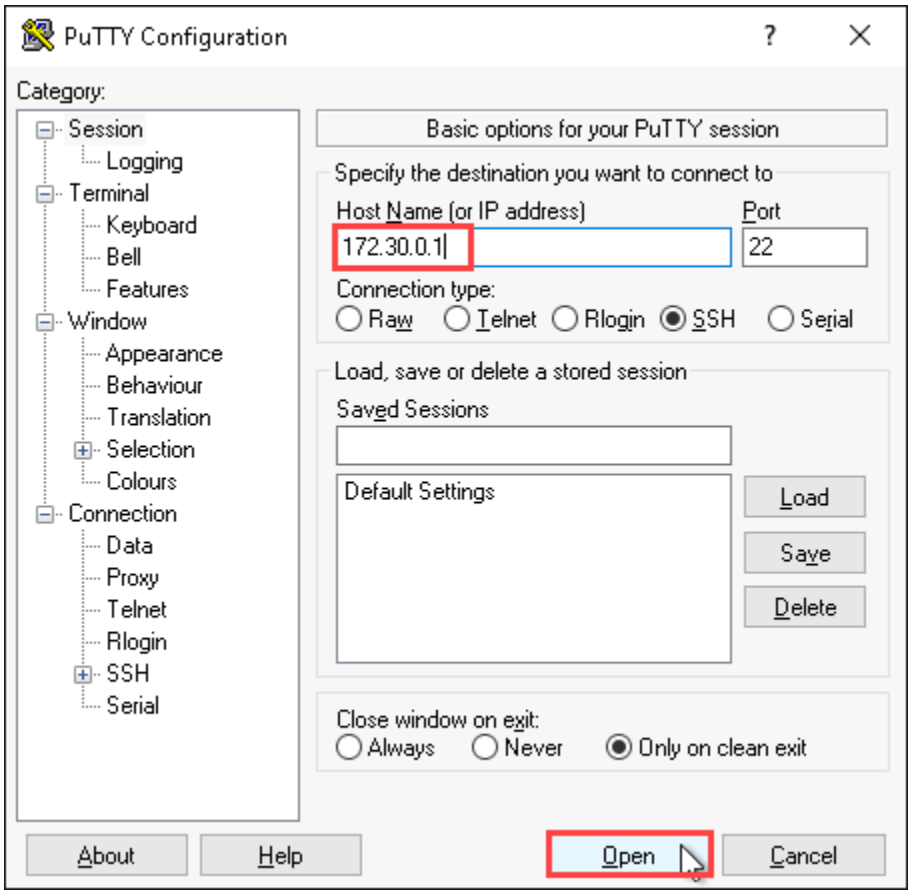
1. On the vWorkstation desktop, **double-click** the **PuTTY icon** to open the PuTTY application.



PuTTY icon

Note: PuTTY is an open-source application used for enabling a console connection or serial port connection, or for performing a file transfer via SFTP and FTP file transfer applications and protocols.

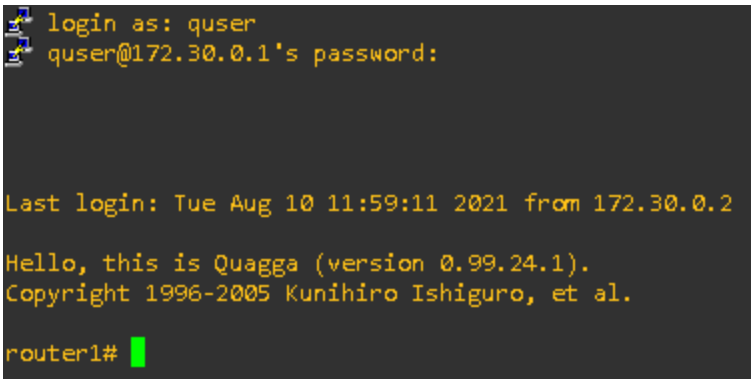
2. In the PuTTY configuration window, **type** `172.30.0.1` in the Host Name field, then **click Open** to open an SSH connection (port 22) to the default gateway router.



PuTTY configuration window

3. At the router1 login prompt, **type** the following credentials and **press Enter** to log in to the router.

   Username: **`quser`**
   Password: **`password`**



Router1 login prompt

Note: PuTTY will not display your password inputs on-screen for security reasons, but rest assured your inputs are being logged.

If you do not enter your credentials within 30 seconds of opening the session, PuTTY will automatically close the session.

4. At the command prompt, **type** `show version` and **press Enter** to display hardware and software information about the router.

```
router1# show version
Quagga 0.99.24.1 ().
Copyright 1996-2005 Kunihiro Ishiguro, et al.
configured with:
    --build=x86_64-linux-gnu --prefix=/usr --includedir=${prefix}/include --mand
ir=${prefix}/share/man --infodir=${prefix}/share/info --sysconfdir=/etc --locals
tatedir=/var --disable-silent-rules --libexecdir=${prefix}/lib/quagga --disable-
maintainer-mode --disable-dependency-tracking --enable-exampledir=/usr/share/doc
/quagga/examples/ --localstatedir=/var/run/quagga --sbindir=/usr/lib/quagga --sy
sconfdir=/etc/quagga --enable-vtysh --enable-isisd --enable-watchquagga --enable
-ospf-te --enable-opaque-lsa --enable-ipv6 --enable-ospfclient=yes --enable-ospf
api=yes --enable-multipath=64 --enable-user=quagga --enable-group=quagga --enabl
e-vty-group=quaggavty --enable-configfile-mask=0640 --enable-logfile-mask=0640 -
-enable-rtadv --enable-gcc-rdynamic --enable-pimd --with-libpam CFLAGS=-g -O2 -f
PIE -fstack-protector-strong -Wformat -Werror=format-security CPPFLAGS=-Wdate-ti
me -D_FORTIFY_SOURCE=2 CXXFLAGS=-g -O2 -fPIE -fstack-protector-strong -Wformat -
Werror=format-security FCFLAGS=-g -O2 -fPIE -fstack-protector-strong FFLAGS=-g -
O2 -fPIE -fstack-protector-strong GCJFLAGS=-g -O2 -fPIE -fstack-protector-strong
 LDFLAGS=-Wl,-Bsymbolic-functions -fPIE -pie -Wl,-z,relro -Wl,-z,now OBJCFLAGS=-
g -O2 -fPIE -fstack-protector-strong -Wformat -Werror=format-security OBJCXXFLAG
S=-g -O2 -fPIE -fstack-protector-strong -Wformat -Werror=format-security
router1#
```

Show version command

Note: The **`show version`** command provides valuable information about the router's internal components that may be useful in a forensics investigation. The information provided will vary depending on the software or hardware implementation, but typically includes such details as the operating system, boot image, system uptime, and available ports. Depending on the context of your investigation, running this command is usually a natural place to start.

Since Quagga is a software routing suite, the **`show version`** command primarily provides information about the software configuration, including Quagga user and group permissions, as well as paths to key executables, libraries, and configuration and log files. For example, in your output you can see that the location of the Quagga configuration files is /etc/quagga, which you could use to determine the startup configuration for Quagga. When compared with the currently running configuration, you are able to determine what, if any, changes were made since the last write or system reboot.

5. **Make a screen capture** showing the **router's version output**.

6. At the command prompt, **type** **`show interface`** and **press Enter** to list details about the router's available network interfaces.

```
router1# show interface
Interface ens192 is up, line protocol detection is disabled
  index 2 metric 0 mtu 1500
  flags: <UP,BROADCAST,RUNNING,MULTICAST>
  HWaddr: 00:50:56:ab:6f:6a
  inet 172.30.0.1/24 broadcast 172.30.0.255
Interface ens224 is up, line protocol detection is disabled
  index 3 metric 0 mtu 1500
  flags: <UP,BROADCAST,RUNNING,MULTICAST>
  HWaddr: 00:50:56:ab:22:c6
  inet 172.20.0.1/24 broadcast 172.20.0.255
Interface ens256 is up, line protocol detection is disabled
  index 4 metric 0 mtu 1500
  flags: <UP,BROADCAST,RUNNING,MULTICAST>
  HWaddr: 00:50:56:ab:08:3f
  inet 172.21.0.1/24 broadcast 172.21.0.255
Interface lo is up, line protocol detection is disabled
  index 1 metric 0 mtu 65536
  flags: <UP,LOOPBACK,RUNNING>
  inet 127.0.0.1/8
router1#
```

Show interface command

Note: The **`show interface`** command can provide important details about the router's network interfaces. These details can include the current status of the interfaces, perhaps revealing interfaces that have been taken offline to prevent communications that may reveal a hacker's presence, or to ensure that traffic is forced to exit via a specific interface. Additionally, the IP and MAC address provide you with information that you can compare against the ARP table to ensure there is fidelity between the output here and the reported mappings in the ARP table.

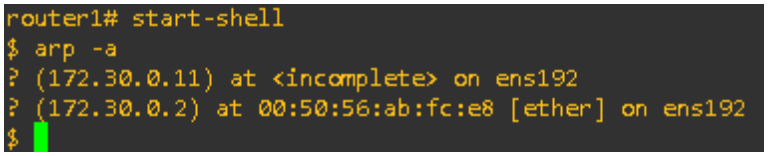
7. **Make a screen capture** showing the **router's interface details**.

Note: In the next steps, you will access the Linux system directly to view the ARP, or Address Resolution Protocol, table. While Quagga manages network layer protocol communications, datalink layer (MAC address) protocol queries require access to the underlying Linux shell.

8. At the command prompt, **type** **`start-shell`** and **press Enter** to gain shell access to the

underlying Linux operating system.

9. At the command prompt, **type** `arp -a` and **press Enter** to display the router's current ARP entries.



ARP table

Note: The Address Resolution Protocol (ARP) is a communications protocol that network devices use to associate a Data Link Layer address (MAC address) with a Network Layer address (IP address). The mapping between IP addresses and their corresponding MAC addresses is what enables a router to transmit IP packets using Ethernet-based Data Link Layer frames across a network.

These mappings are stored in the ARP table or ARP cache, which provides a common vector for bad actors to place themselves in line with network traffic. When the router attempts to communicate with another device via their IP, they will first broadcast "who has this IP," to which the holder will respond "I do, and this is my Layer 2 address, use that to reach me." You can easily imagine that bad actors are eager to answer these requests, providing their own MAC address so that they may intercept traffic destined for the device to which the IP actually belongs. This is known as ARP poisoning, and is a common vector that attackers use to compromise hosts on a network they have gained access to.

10. **Make a screen capture** showing the **router1 ARP table**.

11. At the command prompt, **type** `exit` and **press Enter** to return to the Quagga shell.

12. At the command prompt, **type** `show ip route` and **press Enter** to display all active IP routes and reachable IP network numbers on the Router1 device.

```
router1# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel,
       > - selected route, * - FIB route

C>* 127.0.0.0/8 is directly connected, lo
C>* 172.20.0.0/24 is directly connected, ens224
C>* 172.21.0.0/24 is directly connected, ens256
R>* 172.22.0.0/24 [120/2] via 172.20.0.2, ens224, 00:30:49
R>* 172.23.0.0/24 [120/2] via 172.21.0.2, ens256, 00:30:27
C>* 172.30.0.0/24 is directly connected, ens192
R>* 202.20.3.10/32 [120/2] via 172.21.0.2, ens256, 00:30:27
router1#
```

Show ip route

Note: As a forensic investigator, this command will be helpful for mapping out the network topology by showing you the established routing table. Additionally, manipulating routing tables is one of the primary ways intruders infiltrate routers. This command can provide information to help you determine if a routing table has been altered. For example, you will notice these routes are prefixed with an R code; this indicates that the Routing Information Protocol is being used, which dynamically routes traffic based on some simple criteria. Since such dynamic protocols are typically used in lieu of adding static routes for each destination, the presence of any static routes may indicate malicious activity. Static routes are automatically prioritized over any routes added by a dynamic routing protocol, and therefore provide a way to 'beat' dynamic entries. Additionally, the more specific a route, the higher its precedence, so an attacker may attempt to beat existing static entries by specifying their route with a larger subnet mask. For example, if you were trying to access 10.0.0.1, a route to 10.0.0.1/32 would be preferred over a route to 10.0.0.0/24, despite both routes matching traffic destined for 10.0.0.1.

13. **Make a screen capture** showing the **IP routing table**.

14. At the command prompt, **type** `show running-config` and **press Enter** to display the currently running configuration.

    As needed, **press Enter** to scroll through the output until the command prompt returns.

```
router1# show running-config
Building configuration...

Current configuration:
!
log stdout
!
password zebra
enable password zebra
!
interface ens192
 ip rip authentication mode text
 ip rip authentication string P@ssw0rd!
 ipv6 nd suppress-ra
 no link-detect
!
interface ens224
 ip rip authentication mode text
 ip rip authentication string P@ssw0rd!
 ipv6 nd suppress-ra
 no link-detect
!
interface ens256
 ip rip authentication mode text
 ip rip authentication string P@ssw0rd!
 ipv6 nd suppress-ra
 no link-detect
!
interface lo
 no link-detect
!
router rip
 version 2
 network 172.20.0.0/24
 network 172.21.0.0/24
 network 172.30.0.0/24
 passive-interface ens192
!
ip forwarding
!
line vty
!
end
router1#
```

Show running-config

Note: Recall that network forensics is a careful practice, as analysis is commonly conducted on live devices for which evidence can be lost if the device is power cycled. This is highlighted in the utility of the **`show running-config`** command, which displays the router's current configuration file and may include additional specifications not yet saved to the startup configuration. If this were true, a restart would remove this discrepancy, as any unsaved changes to the running configuration would be blown away and the running configuration would now match the startup configurations (in /etc/quagga,

as discovered earlier) exactly.

In this case, some points of forensic interest include the interface details, which contain authentication details used by RIP-enabled routers to update their neighbor routers, and the router RIP configuration near the bottom, which list the attached networks this device is currently performing routing for.

15. **Make a screen capture** showing the **currently running configuration.**

16. **Close** the **PuTTY window**.

Note: This concludes Section 1 of the lab.

Section 2: Applied Learning

Note: SECTION 2 of this lab allows you to apply what you learned in **SECTION 1** with less guidance and different deliverables, as well as some expanded tasks and alternative methods. You will take a deeper dive into packet analysis and examine firewall logs.

Part 1: Perform Advanced Packet Capture and Analysis

Note: Network forensics often involves sorting through a great deal of noise in search of relevant evidence. It is important for an investigator to be capable of distinguishing between the two – knowing what a regular cog in the machinery looks like, as well as a wrench. Of course, identifying a wrench is just the beginning of a forensic investigation, as you also need to be equipped to explore leads after you have identified them. This may involve digging into packets to uncover details about the host, highlighting prolonged conversations between two entities, and even pulling raw data off the wire to reconstruct files sent across it. This is the distinction between simply knowing what a footprint looks like and knowing how to track a person.

In the next steps, you will begin a Wireshark capture, and then generate FTP (File Transfer Protocol) traffic from the vWorkstation to a remote server located beyond the pfSense perimeter firewall in this topology. You will then inspect your capture, identifying relevant details to your investigation by tracking the details of a packet flow between two entities, and then using the data discovered therein to reconstitute a file transferred across the wire.

1. From the vWorkstation desktop, **launch Wireshark**.

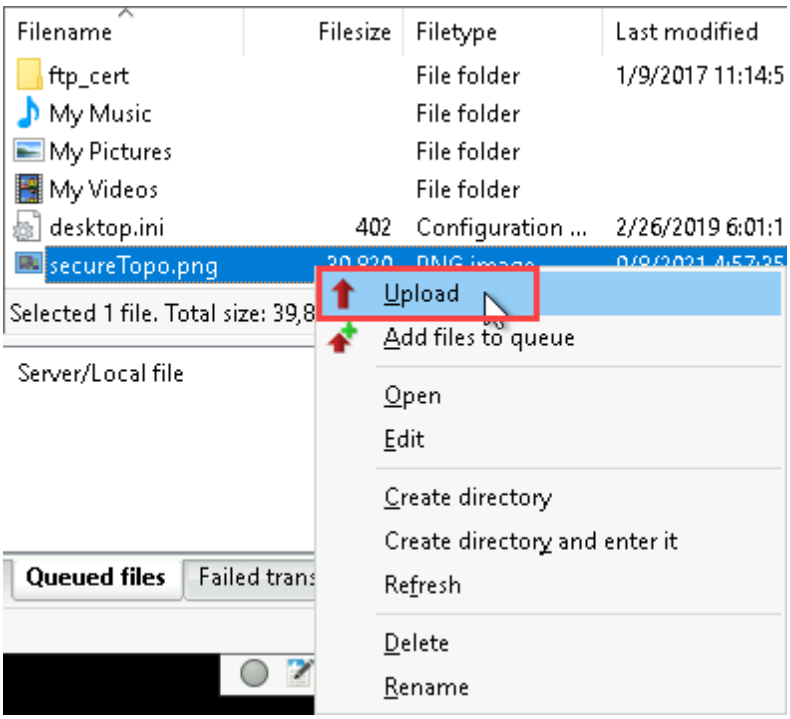
2. **Start** a **packet capture session** on the **Student interface**, then **minimize** the **Wireshark window**.

3. On the vWorkstation desktop, **double-click** the **FileZilla icon** to open the FileZilla Client application.

4. In the FileZilla Client window, **type** the following information, then **click Quickconnect** to open an FTP connection with the Kali system.

   Host: **`202.20.3.10`**
   Username: **`ftpuser01`**
   Password: **`P@ssw0rd!`**
   Port: **`21`**

5. In the Local site pane, **right-click** the **secureTopo.png file** and **select Upload** from the context menu to send the secureTopo.png file to the remote file server.



Upload

6. In the bottom pane, **select** the **Successful transfers tab** to view a log of all successful transfers.

7. **Make a screen capture** showing the **successful transfer of the secureTopo.png file**.

8. **Close** the **FileZilla Client window**.

9. From the vWorkstation taskbar, **restore** the **Wireshark window**.

10. **Stop** your **packet capture session**.

11. **Apply a display filter** to show only traffic using the **FTP protocol**.

Note: File Transfer Protocol (FTP) uses TCP (Transmission Control Protocol) as a connection-oriented Transport Layer protocol. FTP uses one channel for control messages (such as *put* and *get* commands), which is typically the well-known port 21, and another channel for conveying the actual data, which is typically port 20.

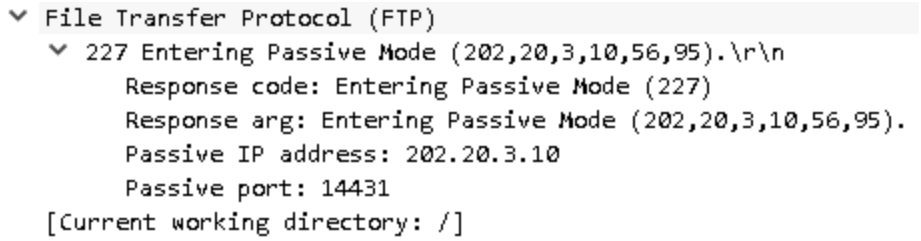
Pay attention to the first few packets in this capture. First, the server reports its type and version, providing some useful identifying information about its setup. It appears to be using the VSFTPD (Very Secure FTP Dameon), which is exclusive to Unix-based machines, so this file server is likely to be running Linux. Following it are several AUTH TLS commands, which is a reference to yet another file transfer protocol, known as FTPS, or FTP Secure, which implements a secure connection over the Transport Layer Security protocol. You may recognize TLS from its common implementation in the HTTPS (HTTP Secure) protocol. The FTP client will attempt to connect via TLS by default (AUTH TLS), and if the FTP server does not support it, it will return an error code. The error will vary depending on the file server distribution and configuration. In this case it is a clean 530, indicating an authentication failure.

The next few packets show some auth prompts and the login credentials you used to establish the connection. Because FTP does not provide encryption, these details are easily gleaned from the capture. Following authentication, a SYST request is sent from the vWorkstation, which is a request for information about the OS. Most servers will report 215 UNIX Type: L8, regardless of their OS, so the response provides little help in identifying the host.

12. **Locate** and **select** the packet with **Response: 227 Entering Passive Mode** in the Info column.

13. In the Packet Details pane, **expand** the **File Transfer Protocol row** to view the FTP details.

14. In the Packet Details pane, **expand** the **227 Entering Passive Mode row** to view the Passive Mode details.



Passive Mode details

Note: FTP connections operate in two modes: Active and Passive. In Active mode, the client first sends a PORT command to the server, typically using the control port 21. In this message, the client specifies a random port number it will be listening on for the data connection (typically port 20). Upon receipt, the server initiates a data connection back to the client through the specified port. This is the mode of operation used when FTP first came about, though it is used much less commonly than Passive mode, because nowadays most clients are behind firewalls, and therefore will not permit the incoming connection to their machine. However, externally-exposed servers, such as a public file server, do accept incoming connections, and so Passive mode was created to change the connection operations to act in only one direction: client > server.

In Passive mode, the client first sends a PASV command to the server (which you can see in the packet immediately preceding this one), requesting the server to instead specify the ports the client should connect to – also typically on port 21. The server will then respond back with a port to connect to, and the client will then establish a data connection on that specified port. You can see those details in this packet – the server responds with a comma-separated string of numbers, where the first four represent the IP address, and the last two provide the port to connect to. If you take the second-to-last value, multiply it by 256, and add the last value, you should get the value specified in the *Passive port:* field shown in the Packet Details. For example, a string of 202,20,3,10,141,125 means the client should connect to the IP 202.20.30.10 on port 36221 (141*256 + 125).

Note that while the ports are negotiated in this transaction, ports are actualized at the Transport Layer (the details of which can be found in the Transmission Control Protocol row immediately above the FTP row).

15. **Make a screen capture** showing the **passive port specified by the FTP server in the Packet Details pane**.

16. In the Packet Details pane, **expand** the **Internet Protocol Version 4 row** to display the network data details.

Note: While the FTP row provided information about the Application Layer protocol information, the IPv4 row provides information about the Network Layer. This layer consists of details contained specifically within the packet (the Network Layer Data Unit), such as the source and destination IP addresses, Quality of Service/Traffic Control flags, VLAN IDs, as well as important packet profile details like the Time-to-Live, or TTL, which represents the number of hops (routers) that can be traversed before the packet expires. TTL can be used to infer the distance between the two conversating parties.

17. In the Packet Details pane, **select** the **Time to live: 61 field** to highlight it.

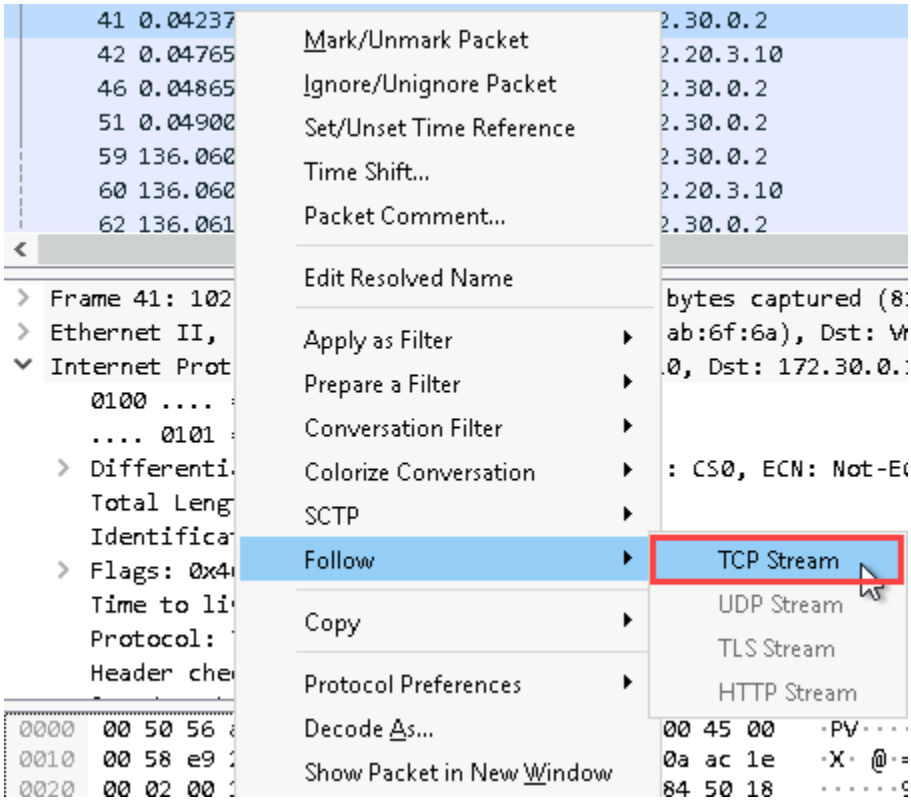
Note: While you may be inclined to interpret this as “distance of 61 hops/routers,” know that this packet was sent out with a TTL of 64, and this value was decremented by one at each router it traversed to get here. See, Linux machines typically set an initial TTL value of 64, while Windows machines use 128. You know this packet was sent from the FTP server, which looks to be a Linux machine, so you may infer that this packet traversed three routers to get here (64-61=3).

18. **Make a screen capture** showing the **Time to live field in the Packet Details pane**.

Note: Wireshark provides another useful tool for forensic investigator in its Follow Stream feature. Streams represent a packet flow between two network endpoints over a specific protocol, providing a way to focus in on a particular exchange. For example, you may want to isolate a TCP stream over which an FTP exchange occurred, in order to discover the username and password used for the connection, or the files (if any) that were uploaded or downloaded. Wireshark streams are primarily characterized by a profile of five attributes, also referred to as a five-tuple, which consists of a source IP, source port, destination IP, destination port, and protocol (TCP, UDP, TLS, or HTTP).

In the final steps of this part, you will take advantage of the Follow Stream tool to track and recover the secureTopo.png file you uploaded to the remote file server.

19. In the Packet List pane, **right-click** the **currently selected packet** and **select Follow > TCP Stream** to open the Follow TCP Stream window.



Follow > TCP Stream

Note: Wireshark color-codes data in the stream, with blue representing transmissions from one endpoint (in this case, the server), and red representing the other endpoint (here, the client). This lean view allows a forensic investigator to easily parse the exchange between two entities, without having to scroll through any extraneous packets. As you can see, your full transaction with the remote file server is neatly chronicled, including the credentials you used to log in, as well as the file you uploaded.

20. **Make a screen capture** showing the **Follow TCP stream window**.

Note: In the next steps, you will locate the raw data for the file you uploaded by searching Wireshark for the relevant file signature, a special string of characters that is unique to a particular file type. File signatures are often used to verify the contents of a file, a way of ensuring that the file extension is an accurate indicator of what you can expect to be contained therein. Also known as magic numbers or bytes, a file signature is a string of hexadecimal characters that is unique to a particular file-type encoding. For example, the file signature for a PDF document is *25 50 44 46 2d,* which is to say that

all PDF-encoded files, when converted to hexadecimal data, will contain this string. It is important to note that this identifier is only present in a genuine PDF-encoded file – that is, simply changing the file extension of a document to .PDF does not make it a PDF file, and therefore it will not contain this string.

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

22. On the Wireshark toolbar, **clear the display filter** so that all packets in the capture are displayed in the Packet List view.

23. **Execute** the **Ctrl-F hotkey combination** to summon the Wireshark Find bar.

24. On the Wireshark Find bar, **click** the **dropdown menu** (currently showing Display filter) and **select Hex value**.

25. In the Find bar, **type** `89 50 4E 47 0D 0A 1A 0A` and **press Enter** to locate a packet containing the PNG (Portable Network Graphics) file signature.

Note: The resulting packet should be an ftp-data packet that shows the *STOR secureTopo.png* command in the Info column. This packet contains the first fragment of the secureTopo.png image that you transferred to the remote file server. Notice that there are additional, seemingly identical packets that follow this one. This is because the image was too large to be sent in a single ftp-data packet, and so was split across numerous transmissions.

As you can see from this example, following streams is not just useful for control connections, but also for gathering the entirety of a data payload that was fragmented into several packets. This is particularly useful in scenarios where you are attempting to reconstruct the file transferred over the connection through assembly of its constituent bytes.

In the final steps, you will follow the TCP stream that this packet is a part of. This will provide you will the complete set of bits that constitute the PNG file. You will use this set to reconstruct the file on your desktop.

26. **Follow** the **TCP stream** for this packet.

Note: The TCP Stream window contains each byte that makes up the secureTopo.png image, displayed by default in ASCII format. Because you know this is PNG-encoded, and have each component byte of data, you should be able to save the raw data of this image and thereby reconstitute the original picture.

27. At the bottom of this window, **select Raw** from the *Show and save data as* menu.

Note: While Raw refers to binary format (base 2), the data currently in view is actually in a hexadecimal format. This is because binary is not human-readable, and so the common convention is to display it in hexadecimal instead (which converts cleanly, because hexadecimal is base 16, which is a power of 2). However, saving this data in Raw format will ensure the resulting file is in binary format.

28. At the bottom of this window, **click** the **Save as… button**, then **save** your file to the Desktop as ***yourname*_secureTopo.png**, where *yourname* is your own name.

29. **Close** the **TCP Stream window**.

30. **Close** the **Wireshark window**.

    When prompted, quit without saving.

31. From the vWorkstation desktop, **open** the ***yourname*_secureTopo.png file**.

32. **Make a screen capture** showing the **reconstituted PNG file**.

33. **Close** any **open windows**.

Part 2: Analyze a Firewall for Forensic Evidence

Note: Firewalls are responsible for permitting or denying all incoming and outgoing traffic in a network.

For this reason, they occupy a critical junction in a network infrastructure. Provided they are configured to do so, firewalls will maintain a log of details on important events that can assist an investigator in tracing an attacker and determining the vector of a network breach. For this reason, firewall forensics is among the most important branches of digital forensics.

The pfSense firewall at the top of this network infrastructure serves as the first layer of security for potential attackers seeking to breach the network. The perimeter firewall straddles the line between the internal infrastructure and the public Internet, with at least one interface attached to an internal network, and another (commonly referred to as the WAN, or Wide Area Network interface) attached to an external network, the first exit for any internet-bound traffic.

In the next steps, you will access the pfSense webGUI, a graphical interface for interacting with the firewall. You will then inspect several log entries and learn how the information contained therein may be used to advance an investigation.

1. On the vWorkstation taskbar, **click** the **Chrome icon** to open a new browser window.

2. In the browser's address box, **type** **`172.23.0.254`** and **press Enter** to open the pfSense web interface.

3. At the pfSense log-in page, **use the following credentials** to sign in to the pfSense webGUI.

   Username: **`admin`**
   Password: **`pfsense`**

Note: You may see a WARNING message about the admin account being set to the default value. Please disregard this message and continue to the next step. While it is a security best practice to change the administrator password to something other than its default value, you do not need to do so in order to complete this lab.

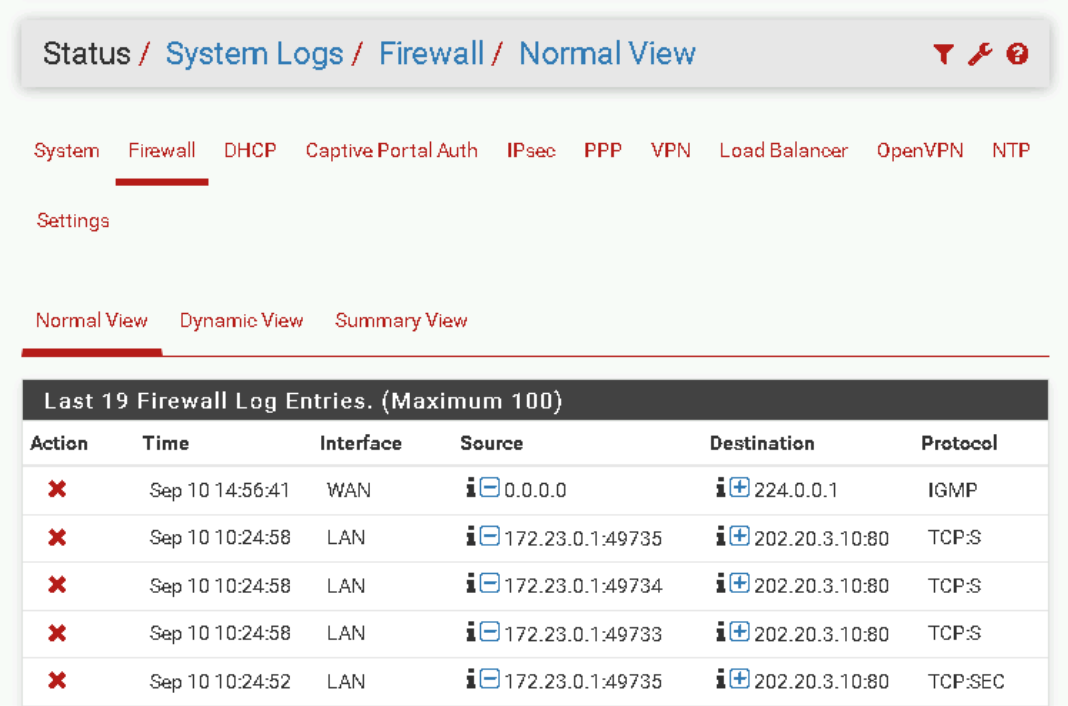
4. On the pfSense menu bar, **click Status** and **select System Logs** to access your firewall's log management console.

5. On the Status / System Logs / System / General page, **click** the **Settings tab** to access the pfSense logging settings.

6. In the General Logging Options module, **click** the **Forward/Reverse Display checkbox** to display the newest log entries at the top of the log.

7. At the bottom of the page, **click** the **Save button** to save your changes.

8. At the top of the page, **click** the **Firewall tab** to view the firewall logs.



Firewall logs

Note: By default, pfSense only logs blocked connections, although you can also configure the firewall to log permitted connections, or finely tune it to only log events that match particular rules. In this instance, it appears an SSH rule on the WAN interface has logging enabled, as indicated by the green checkmark entry in the middle of the list. Presumably this rule was created by the system administrator to remotely manage the firewall, for which they enabled logging so that they may track access events.

Preceding the successful SSH hit, you will find several blocked incoming connections from source IP 202.20.3.10 to several common TCP ports. Patterns like this may indicate that an attacker conducted a port scan on the firewall in search of open ports to squeeze through. Following it are failed HTTP outgoing connections to 202.20.3.10, possibly indicating an attacker has successfully breached the network and then pivoted to another internal machine, or implemented some nefarious configuration to

redirect traffic from authorized users.

9. **Make a screen capture** showing the **entries in the firewall log**.

10. In the firewall log, **click** the **i icon beside the IP address 202.20.3.10** on any entry in the file to attempt to resolve the IP address to its domain name.

Resolve the IP address

Note: It appears this IP address resolves to corporationtechs.com. If this were a familiar website to this company, but the IP address was not, you may have reason to suspect a DNS spoofing attack. In such a scenario, the attacker may 'poison' the DNS cache, such that any traffic sent to corporationtechs.com is now sent to 202.20.3.10, instead of its regular address. Alternatively, the attacker may hijack the DNS server altogether, taking full control of the corporationtechs.com domain. In either case, the risk is that internal users may now be at risk of communicating sensitive information to the nefarious party while being entirely unaware they are not connecting to the genuine site.

11. **Make a screen capture** showing the **resolved entries in the firewall log**.

12. **Close** the **Chrome browser window**.

Note: This concludes Section 2 of the lab.

Section 3: Challenge and Analysis

Note: The following exercises are provided to allow independent, unguided work using the skills you learned earlier in this lab - similar to what you would encounter in a real-world situation.

Part 1: Identify the Source of a Suspicious Route

For this section of the lab, assume that the work you completed in Sections 1 and 2 was performed within the context of a forensic investigation being conducted with the objective of identifying suspicious activity on your employer's network. You have already spent some time reviewing the default gateway router in the initial network segment (where the vWorkstation is located), and then moved to the firewall to search for suspicious activity in the logs. After discovering some concerning connections from a host at 202.20.3.10, you decide that the initial routing table that you reviewed in Section 1, Part 2 deserves closer inspection. You know that your employer has recently implemented dynamic routing in anticipation of growing their network, and has made it clear that no static/manual routes should be used to direct traffic in the infrastructure. Although you do not recall seeing any static routes on your first inspection of the routing table, perhaps one of the routes it has acquired was manually added to a neighboring router, and then advertised to its peers. In this part of the lab, you will take a closer look to determine if there is any merit to this hypothesis.

From the vWorkstation, establish an SSH connection to 172.30.0.1. Once connected, execute the command to show all network routes currently being served. Identify the route with the highest subnet mask, and then determine where traffic matching this route would be passed next.

Next, use the same credentials to establish an SSH connection to the next hop you identified (a neighboring router), and see if you can identify a route that is neither directly connected (prefixed with a C code) nor provided as part of the RIP configuration file (prefixed with an R code), indicating that it was added independent of the dynamic routing configuration.

Make a screen capture showing the **non-RIP route that you discovered on the target router**.

Part 2: Identify Suspicious Outgoing Connections

As a seasoned security analyst, you know humans tend to be the most vulnerable component of an organization. A firewall can be set to deny all incoming connections, but rulesets for outgoing connections are typically much more liberal, with many set to “allow all traffic by default.” By targeting a user in the organization, the attacker does not need to ever breach the firewall, compromise a machine, and call back to their C2. They simply need to guide the target user to make that callback for them and boom, they’re in.

In this part of the lab, you will assume the role of the target user and download a malicious file from an external website. This outgoing attempt will be stopped by the firewall, as this organization has a more restrictive outgoing access ruleset. You will then put your forensic investigator hat back on and inspect the firewall logs in search of this malicious egress attempt.

From the vWorkstation, launch Google Chrome and navigate to corporationtechs.com/fixIt.exe (*Note: fixIt.exe is case-sensitive*), then download and run the executable. Next, review the pfSense firewall logs to identify the connection attempt made after downloading the malicious file.

Record the destination IP address and Port number of the outgoing connection attempt.

Note: This concludes Section 3 of the lab.