Lab – Encrypting and Decrypting Data Using OpenSSL

- Part 1: Encrypting Messages with OpenSSL
- Part 2: Decrypting Messages with OpenSSL

Background / Scenario

OpenSSL is an open source project that provides a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a general-purpose cryptography library. In this lab, you will use OpenSSL to encrypt and decrypt text messages.

Note: While OpenSSL is the de facto cryptography library today, the use presented in this lab is NOT recommended for robust protection. Below are two security problems with this lab:

1. The method described in this lab uses a weak key derivation function. The ONLY security is introduced by a very strong password.
2. The method described in this lab does not guarantee the integrity of the text file.

This lab should be used for instructional purposes only. The methods presented here should NOT be used to secure truly sensitive data.

Required Resources

- CyberOps Workstation Virtual Machine
- Internet access

Part 1: Encrypting Messages with OpenSSL

OpenSSL can be used as a standalone tool for encryption. While many encryption algorithms can be used, this lab focuses on AES. To use AES to encrypt a text file directly from the command line using OpenSSL, follow the steps below:

Step 1: Encrypting a Text File

a. Log into CyberOPS Workstation VM.

b. Open a terminal window.

c. Because the text file to be encrypted is in the /home/analyst/lab.support.files/ directory, change to that directory:

```
[analyst@secOps ~]$ cd ./lab.support.files/
```

d. Type the command below to list the contents of the encrypted **letter_to_grandma.txt** text file on the screen:

```
[analyst@secOps lab.support.files]$ cat letter_to_grandma.txt
```

```
Hi Grandma,

I am writing this letter to thank you for the chocolate chip cookies you sent

me. I got them this morning and I have already eaten half of the box! They

are absolutely delicious!
```

e. From the same terminal window, issue the command below to encrypt the text file. The command will use AES-256 to encrypt the text file and save the encrypted version as **message.enc.** OpenSSL will ask for a password and for password confirmation. Provide the password as requested and be sure to remember the password.

```
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -in

letter_to_grandma.txt -out message.enc

enter aes-256-cbc encryption password:
```

Document the password.
Student choice of password
f. When the process is finished, use the **cat** command again to display the contents of the **message.enc** file.

```
[analyst@secOps lab.support.files]$ cat message.enc
```

Did the contents of the **message.enc** file display correctly? What does it look like? Explain.
No. The file seems broken as just symbols are displayed. The symbols are shown because OpenSSL has generated a binary file.
g. To make the file readable, run the OpenSSL command again, but this time add the **-a** option. The **-a** option tells OpenSSL to encode the encrypted message using a different encoding method of Base64 before storing the results in a file.
Note: Base64 is a group of similar binary-to-text encoding schemes used to represent binary data in an ASCII string format.

```
[analyst@secOps lab.support.files]$ openssl aes-256-cbc -a -in
```

```
enter aes-256-cbc encryption password:
```

h. Once again, use the **cat** command to display the contents of the, now re-generated, **message.enc** file:
Note: The contents of **message.enc** will vary.

```
[analyst@secOps lab.support.files]$ cat message.enc

U2FsdGVkX19ApWyrn8RD5zNp0RPCuMGZ98wDc26u/vmj1zyDXobGQhm/dDRZasG7

rfnth5Q8NHValEw8vipKGM66dNFyyr9/hJUzCoqhFpRHgNn+Xs5+TOtz/QCPN1bi

08LGTSzOpfkg76XDCk8uPy1hl/+Ng92sM5rgMzLXfEXtaYe5UgwOD42U/U6q73pj

a1ksQrTWsv5mtN7y6mh02Wobo3A1ooHrM7niOwK1a3YKrSp+ZhYzVTrtksWDl6Ci
```

Is **message.enc** displayed correctly now? Explain.
Yes. While message.enc is encrypted, it is now correctly displayed because it has been converted from binary to text and encoded with Base64.
Can you think of a benefit of having **message.enc** Base64-encoded?
The encrypted message can now be copied and pasted in an email message, for example.

Part 2: Decrypting Messages with OpenSSL

With a similar OpenSSL command, it is possible to decrypt **message.enc.**
a. Use the command below to decrypt message.enc:

```
[analyst@secOps lab.support.files]$ openssl aes-256-cbc –a -d -in message.enc -out decrypted_letter.txt
```

b. OpenSSL will ask for the password used to encrypt the file. Enter the same password again.

c. When OpenSSL finishes decrypting the **message.enc** file, it saves the decrypted message in a text file called **decrypted_letter.txt.** Use the **cat** display the contents of **decrypted_letter.txt:**

```
[analyst@secOps lab.support.files]$ cat decrypted_letter.txt
```

Lab – Snort and Firewall Rules

- **Part 1: Preparing the Virtual Environment**
- **Part 2: Firewall and IDS Logs**
- **Part 3: Terminate and Clear Mininet Process**

Background / Scenario

In a secure production network, network alerts are generated by various types of devices such as security appliances, firewalls, IPS devices, routers, switches, servers, and more. The problem is that not all alerts are created equally. For example, alerts generated by a server and alerts generated by a firewall will be different and vary in content and format.

In this lab, to get familiar with firewall rules and IDS signatures.

Required Resources

- CyberOps Workstation virtual machine
- Internet connection

Note: In this lab, the CyberOps Workstation VM is a container for holding the Mininet environment shown in the Topology. If a memory error is received in an attempt to run any command, quit out of the step, go to the VM settings, and increase the memory. The default is 1 GB; try 2GB.

Instructions

Part 1: Preparing the Virtual Environment

a. Launch **Oracle VirtualBox** and change the **CyberOps Workstation** for Bridged mode, if necessary. Select **Machine > Settings > Network**. Under **Attached To**, select **Bridged Adapter** (or if you are using WiFi with a proxy, you may need **NAT adapter**) and click **OK**.
b. Launch the **CyberOps Workstation VM**, open a terminal and configure its network by executing the **sh** script.
Because the script requires super-user privileges, provide the password for the user **analyst**.

```
[analyst@secOps ~]$ sudo ./lab.support.files/scripts/configure_as_dhcp.sh

[sudo] password for analyst:

[analyst@secOps ~]$
```

c. Use the ifconfig command to verify **CyberOps Workstation VM** now has an IP address on your local network. You can also test connectivity to a public webserver by pinging www.cisco.com. Use **Ctrl+C** to stop the pings.

```
[analyst@secOps ~]$ ping www.cisco.com
PING e2867.dsca.akamaiedge.net (23.204.15.199) 56(84) bytes of data.
64 bytes from a23-204-15-199.deploy.static.akamaitechnologies.com (23.204.15.199):
icmp_seq=1 ttl=54 time=28.4 ms
64 bytes from a23-204-15-199.deploy.static.akamaitechnologies.com (23.204.15.199):
icmp_seq=2 ttl=54 time=35.5 ms
^C
--- e2867.dsca.akamaiedge.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 28.446/32.020/35.595/3.578 ms
```

Part 2: Firewall and IDS Logs

Firewalls and Intrusion Detection Systems (IDS) are often deployed to partially automate the traffic monitoring task. Both firewalls and IDSs match incoming traffic against administrative rules. Firewalls usually compare the packet header against a rule set while IDSs often use the packet payload for rule set comparison. Because firewalls and IDSs apply the pre-defined rules to different portions of the IP packet, IDS and firewall rules have different structures.

While there is a difference in rule structure, some similarities between the components of the rules remain. For example, both firewall and IDS rules contain matching components and action components. Actions are taken after a match is found.

- **Matching component** – specifies the packet elements of interest, such as: packet source; the packet destination; transport layer protocols and ports; and data included in the packet payload.
- **Action component** – specifies what should be done with that packet that matches a component, such as: accept and forward the packet; drop the packet; or send the packet to a secondary rule set for further inspection.

A common firewall design is to drop packets by default while manually specifying what traffic should be allowed. Known as dropping-by-default, this design has the advantage protecting the network from unknown protocols and attacks. As part of this design, it is common to log the events of dropped packets since these are packets that were not explicitly allowed and therefore, infringe on the organization's policies. Such events should be recorded for future analysis.

Step 1: Real-Time IDS Log Monitoring

a. From the **CyberOps Workstation VM**, run the script to start **mininet**.

```
[analyst@secOps ~]$ sudo
./lab.support.files/scripts/cyberops_extended_topo_no_fw.py

[sudo] password for analyst:

*** Adding controller

*** Add switches

*** Add hosts

*** Add links

*** Starting network

*** Configuring hosts

R1 R4 H1 H2 H3 H4 H5 H6 H7 H8 H9 H10 H11

*** Starting controllers

*** Starting switches
```

*** Add routes

*** Post configure switches and hosts

*** Starting CLI:

mininet>

The **mininet** prompt should be displayed, indicating **mininet** is ready for commands.
b. From the **mininet** prompt, open a shell on **R1** using the command below:

mininet> xterm R1

mininet>

The **R1** shell opens in a terminal window with black text and white background. What user is logged into that shell? What is the indicator of this?
The root user. This is indicated by the # sign after the prompt.
c. From **R1's** shell, start the Linux-based IDS, Snort.
[root@secOps analyst]# **./lab.support.files/scripts/start_snort.sh**
Running in IDS mode
--== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
<output omitted>
Note: You will not see a prompt as Snort is now running in this window. If for any reason, Snort stops running and the **[root@secOps analysts]#** prompt is displayed, rerun the script to launch Snort. Snort must be running to capture alerts later in the lab.
d. From the **CyberOps Workstation VM mininet** prompt, open shells for hosts **H5** and **H10**.

mininet> xterm H5

mininet> xterm H10

mininet>

e. **H10** will simulate a server on the Internet that is hosting malware. On **H10**, run the **mal_server_start.sh** script to start the server.

[root@secOps analyst]# ./lab.support.files/scripts/mal_server_start.sh

[root@secOps analyst]#

f. On **H10**, use `netstat` with the `-tunpa` options to verify that the web server is running. When used as shown below, `netstat` lists all ports currently assigned to services:

[root@secOps analyst]# netstat -tunpa

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 0.0.0.0:6666 0.0.0.0:* LISTEN 1839/nginx: master
[root@secOps analyst]#
As seen by the output above, the lightweight webserver **nginx** is running and listening to connections on port TCP 6666.
g. In the **R1** terminal window, an instance of Snort is running. To enter more commands on **R1**, open another **R1** terminal by entering the **xterm R1** again in the **CyberOps Workstation VM** terminal window. You may also want to arrange the terminal windows so that you can see and interact with each device.
h. In the new **R1** terminal tab, run the **tail** command with the **-f** option to monitor the **/var/log/snort/alert** file in real-time. This file is where snort is configured to record alerts.

[root@sec0ps analyst]# tail -f /var/log/snort/alert

Because no alerts were yet recorded, the log should be empty. However, if you have run this lab before, old alert entries may be shown. In either case, you will not receive a prompt after typing this command. This window will display alerts as they happen.

i. From **H5**, use the wget command to download a file named **Nimda.Amm.exe**. Designed to download content via HTTP, **wget** is a great tool for downloading files from web servers directly from the command line.
[root@secOps analyst]# **wget 209.165.202.133:6666/W32.Nimda.Amm.exe**
--2017-04-28 17:00:04-- http://209.165.202.133:6666/W32.Nimda.Amm.exe
Connecting to 209.165.202.133:6666... connected.
HTTP request sent, awaiting response... 200 OK
Length: 345088 (337K) [application/octet-stream]
Saving to: 'W32.Nimda.Amm.exe'

W32.Nimda.Amm.exe
100%[======================================>] 337.00K --.-KB/s in 0.02s

2017-04-28 17:00:04 (16.4 MB/s) - 'W32.Nimda.Amm.exe' saved [345088/345088]

[root@secOps analyst]#
What port is used when communicating with the malware web server? What is the indicator?

Port 6666. The port was specified in the URL, after the : separator.
Was the file completely downloaded?

Yes
Did the IDS generate any alerts related to the file download?

Yes
j. As the malicious file was transiting **R1**, the IDS, Snort, was able to inspect its payload. The payload matched at least one of the signatures configured in Snort and triggered an alert on the second **R1** terminal window (the tab where **tail -f** is running). The alert entry is show below. Your timestamp will be different:

04/28-17:00:04.092153 [**] [1:1000003:0] Malicious Server Hit! [**] [Priority: 0] {TCP} 209.165.200.235:34484 -> 209.165.202.133:6666

Based on the alert shown above, what was the source and destination IPv4 addresses used in the transaction?

Source IP: 209.165.200.235; Destination IP: 209.165.202.133.
Based on the alert shown above, what was the source and destination ports used in the transaction?

Source port: 34484; Destination port: 6666. (Note: the source port will vary).
Based on the alert shown above, when did the download take place?

April 28th around 5pm for the example, but the student's answer will be different.
Based on the alert shown above, what was the message recorded by the IDS signature?

"Malicious Server Hit!"
On **H5**, use the tcpdump command to capture the event and download the malware file again so you can capture the transaction. Issue the following command below start the packet capture:

[root@secOps analyst]# tcpdump –i H5-eth0 –w nimda.download.pcap &

[1] 5633

[root@secOps analyst]# tcpdump: listening on H5-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes

The command above instructs tcpdump to capture packets on interface **H5-eth0** and save the capture to a file named **nimda.download.pcap**.
The **&** symbol at the end tells the shell to execute **tcpdump** in the background. Without this symbol, **tcpdump** would make the terminal unusable while it was running. Notice the **[1] 5633**; it indicates one process was sent to background and its process ID (PID) is 5366. Your PID will most likely be different.
k. Press **ENTER** a few times to regain control of the shell while **tcpdump** runs in background.
l. Now that **tcpdump** is capturing packets, download the malware again. On **H5**, re-run the command or use the up arrow to recall it from the command history facility.
[root@secOps analyst]# **wget 209.165.202.133:6666/W32.Nimda.Amm.exe**

--2017-05-02 10:26:50-- http://209.165.202.133:6666/W32.Nimda.Amm.exe
Connecting to 209.165.202.133:6666... connected.
HTTP request sent, awaiting response... 200 OK
Length: 345088 (337K) [application/octet-stream]
Saving to: 'W32.Nimda.Amm.exe'

W32.Nimda.Amm.exe 100%[==================>] 337.00K --.-KB/s in 0.003s

2017-05-02 10:26:50 (105 MB/s) - 'W32.Nimda.Amm.exe' saved [345088/345088]
m. Stop the capture by bringing **tcpdump** to foreground with the **fg** Because **tcpdump** was the only process sent to background, there is no need to specify the PID. Stop the **tcpdump** process with **Ctrl+C**. The **tcpdump** process stops and displays a summary of the capture. The number of packets may be different for your capture.
[root@secOps analyst]# **fg**
tcpdump -i h5-eth0 -w nimda.download.pcap
^C316 packets captured
316 packets received by filter
0 packets dropped by kernel
[root@secOps analyst]#
n. On **H5**, Use the ls command to verify the pcap file was in fact saved to disk and has size greater than zero:

[root@secOps analyst]# ls -l

total 1400

drwxr-xr-x 2 analyst analyst 4096 Sep 26 2014 Desktop

drwx------ 3 analyst analyst 4096 Jul 14 11:28 Downloads

drwxr-xr-x 8 analyst analyst 4096 Jul 25 16:27 lab.support.files

-rw-r--r-- 1 root root 371784 Aug 17 14:48 nimda.download.pcap
drwxr-xr-x 2 analyst analyst 4096 Mar 3 15:56 second_drive
-rw-r--r-- 1 root root 345088 Apr 14 15:17 W32.Nimda.Amm.exe
-rw-r--r-- 1 root root 345088 Apr 14 15:17 W32.Nimda.Amm.exe.1
[root@secOps analyst]#
Note: Your directory list may have a different mix of files, but you should still see the **nimda.download.pcap file**.
How can be this PCAP file be useful to the security analyst?

PCAP files contain the packets related to the traffic seen by the capturing NIC. In that way, the PCAP is very useful to re-retrace network events such as communication to malicious end points. Tools such as Wireshark can be used to facilitate PCAP analysis.

Note: The analysis of the PCAP file will be performed in another lab.

Step 2: Tuning Firewall Rules Based on IDS Alerts

In Step 1, you started an internet-based malicious server. To keep other users from reaching that server, it is recommended to block it in the edge firewall.

In this lab's topology, **R1** is not only running an IDS but also a very popular Linux-based firewall called **iptables**. In this step, you will block traffic to the malicious server identified in Step 1 by editing the firewall rules currently present in **R1**.
Note: While a comprehensive study of **iptables** is beyond the scope of this course, **iptables** basic logic and rule structure is fairly straight-forward.
The firewall **iptables** uses the concepts of *chains* and *rules* to filter traffic.
Traffic entering the firewall and destined to the firewall device itself is handled by the **INPUT** chain. Examples of this traffic are ping packets coming from any other device on any networks and sent to any one of the firewall's interfaces.
Traffic originated in the firewall device itself and destined to somewhere else, is handled by the **OUTPUT** chain. Examples of this traffic are ping responses generated by the firewall device itself.
Traffic originated somewhere else and passing through the firewall device is handled by the **FORWARD** chain. Examples of this traffic are packets being routed by the firewall.
Each chain can have its own set of independent rules specifying how traffic is to be filtered for that chain. A chain can have practically any number of rules, including no rule at all.

Rules are created to check specific characteristics of packets, allowing administrators to create very comprehensive filters. If a packet doesn't match a rule, the firewall moves on to the next rule and checks again. If a match is found, the firewall takes the action defined in the matching rule. If all rules in a chain have been checked and yet no match was found, the firewall takes the action specified in the chain's policy, usually allow the packet to flow through or deny it.

a. In the **CyberOps Workstation VM**, start a third R1 terminal window.
mininet > **xterm R1**
b. In the new **R1** terminal window, use the iptables command to list the chains and their rules currently in use:

```
[root@secOps ~]# iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 6 packets, 504 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

[root@secOps ~]#
```

What chains are currently in use by **R1**?
INPUT, OUTPUT and FORWARD
c. Connections to the malicious server generate packets that must transverse the **iptables** firewall on R1. Packets traversing the firewall are handled by the FORWARD rule and therefore, that is the chain that will receive the blocking rule. To keep user computers from connecting to the malicious server identified in Step 1, add the following rule to the FORWARD chain on **R1**:
[root@secOps ~]# **iptables -I FORWARD -p tcp -d 209.165.202.133 --dport 6666 -j DROP**
[root@secOps ~]#
Where:

- **-I FORWARD**: inserts a new rule in the FORWARD chain.
- **-p tcp**: specifies the TCP protocol.
- **-d 209.165.202.133**: specifies the packet's destination
- **–dport 6666**: specifies the destination port
- **-j DROP**: set the action to drop.

d. Use the iptables command again to ensure the rule was added to the FORWARD chain. The CyberOps Workstation VM may take a few seconds to generate the output:
[root@secOps analyst]# **iptables -L -v**
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target   prot opt in   out   source   destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target   prot opt in   out   source   destination
    0   0 DROP   tcp -- any   any   anywhere   209.165.202.133   tcp dpt:6666

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target   prot opt in   out   source   destination
[root@secOps analyst]#
e. On **H5**, try to download the file again:
[root@secOps analyst]# **wget 209.165.202.133:6666/W32.Nimda.Amm.exe**
--2017-05-01 14:42:37-- http://209.165.202.133:6666/W32.Nimda.Amm.exe
Connecting to 209.165.202.133:6666... failed: Connection timed out.
Retrying.

--2017-05-01 14:44:47-- (try: 2) http://209.165.202.133:6666/W32.Nimda.Amm.exe
Connecting to 209.165.202.133:6666... failed: Connection timed out.
Retrying.
Enter **Ctrl+C** to cancel the download, if necessary.
Was the download successful this time? Explain.

No. The firewall is blocking connections to the malware hosting server.
What would be a more aggressive but also valid approach when blocking the offending server?

Instead of specifying IP, protocol and port, a rule could simply block the server's IP address. This would completely cut access to that server from the internal network.

Part 3: Terminate and Clear Mininet Process

a. Navigate to the terminal used to start Mininet. Terminate the Mininet by entering **quit** in the main CyberOps VM terminal window.
b. After quitting Mininet, clean up the processes started by Mininet. Enter the password **cyberops** when prompted.

[analyst@secOps scripts]$ sudo mn –c

[sudo] password for analyst:

Lab – Extract an Executable from a PCAP

Part 1: Analyze Pre-Captured Logs and Traffic Captures

In Part 2, you will work with the **nimda.download.pcap** file. Captured in a previous lab, **nimda.download.pcap** contains the packets related to the download of the Nimda malware. Your version of the file, if you created it in the previous lab and did not reimport your CyberOps Workstation VM, is stored in the /home/analyst directory. However, a copy of that file is also stored in the **CyberOps Workstation VM**, under the **/home/analyst/lab.support.files/pcaps** directory so that you can complete this lab. For consistency of output, the lab will use the stored version in the **pcaps** directory.
While **tcpdump** can be used to analyze captured files, **Wireshark's** graphical interface makes the task much easier. It is also important to note that **tcpdump** and **Wireshark** share the same file format for packet captures; therefore, PCAP files created by one tool can be opened by the other.
a. Change directory to the **support.files/pcaps** folder, and get a listing of files using the `ls –l` command.

[analyst@secOps ~]$ cd lab.support.files/pcaps

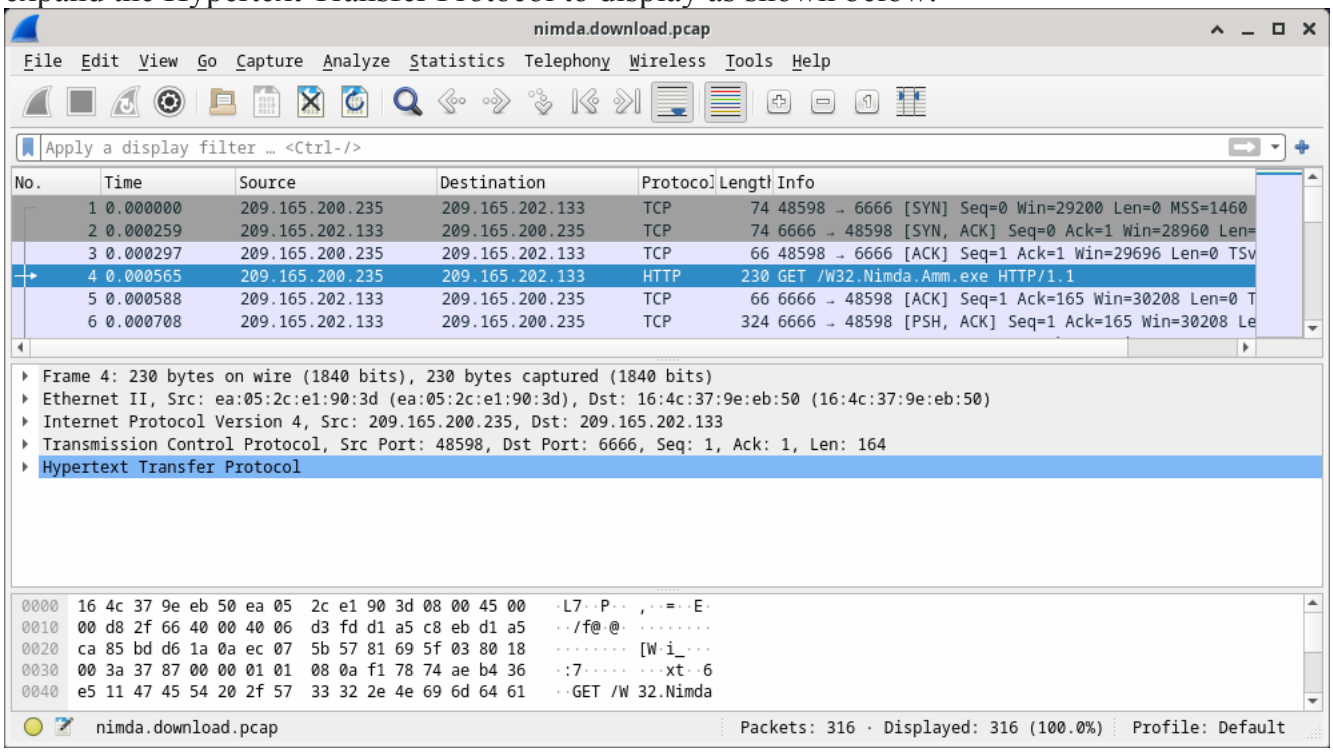
[analyst@secOps pcaps]$ ls -l

total 7460

-rw-r--r-- 1 analyst analyst 3510551 Aug 7 15:25 lab_prep.pcap

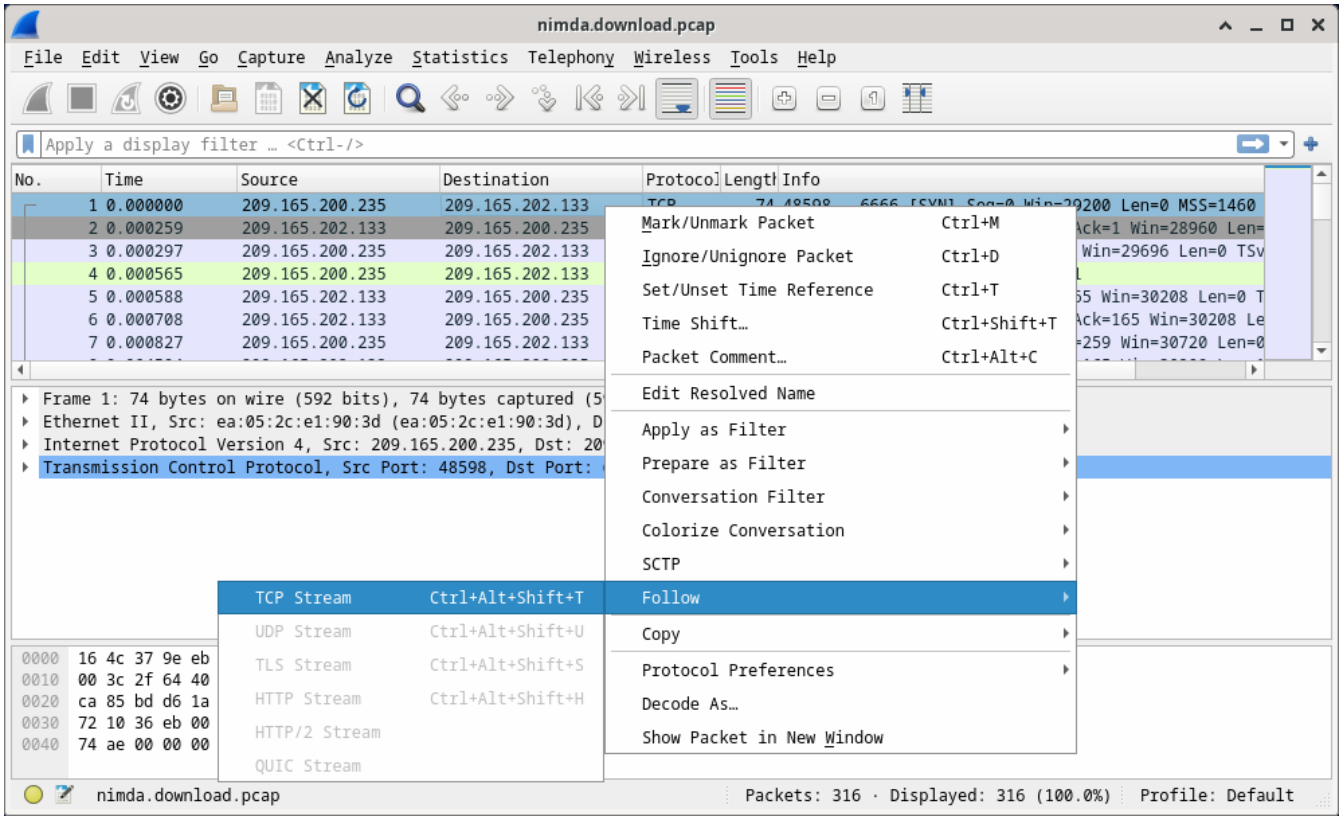
-rw-r--r-- 1 analyst analyst 371462 Jun 22 10:47 nimda.download.pcap
-rw-r--r-- 1 analyst analyst 3750153 May 25 11:10 wannacry_download_pcap.pcap
[analyst@secOps pcaps]$

b. Issue the command below to open the **download.pcap** file in Wireshark.
[analyst@secOps pcaps]$ **wireshark nimda.download.pcap &**
c. The **download.pcap** file contains the packet capture related to the malware download performed in a previous lab. The **pcap** contains all the packets sent and received while **tcpdump** was running. Select the fourth packet in the capture and expand the Hypertext Transfer Protocol to display as shown below.

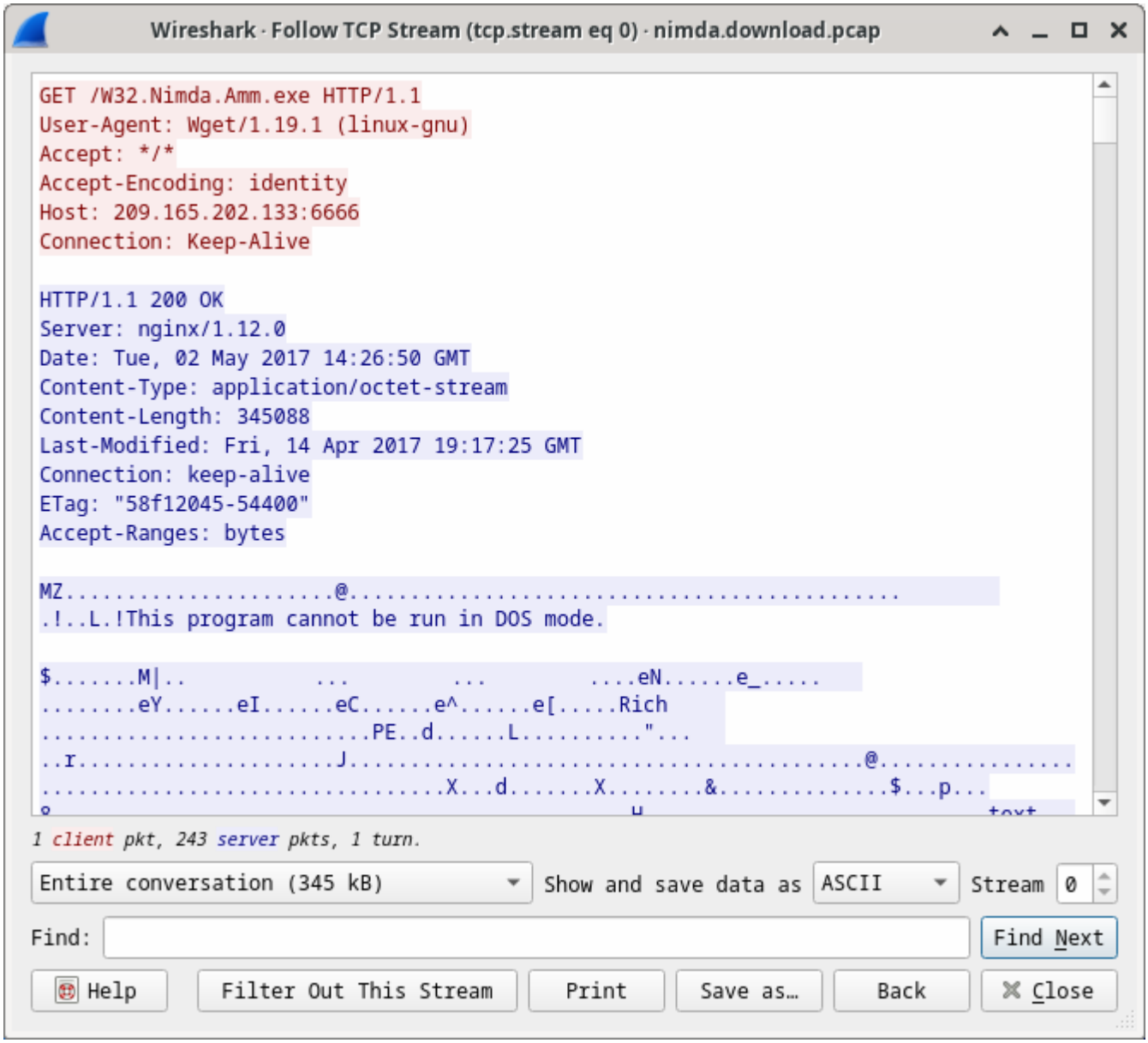


d. Packets one through three are the TCP handshake. The fourth packet shows the request for the malware file. Confirming what was already known, the request was done over HTTP, sent as a GET request.

e. Because HTTP runs over TCP, it is possible to use **Wireshark**'s **Follow TCP Stream** feature to rebuild the TCP transaction. Select the first TCP packet in the capture, a SYN packet. Right-click it and choose **Follow** > **TCP Stream**.



f. Wireshark displays another window containing the details for the entire selected TCP flow.



What are all those symbols shown in the **Follow TCP Stream** window? Are they connection noise? Data? Explain.

The symbols are the actual contents of the downloaded file. Because it is binary file, Wireshark does not know how to represent it. The displayed symbols are Wireshark's best guess at making sense of the binary data while decoding it as text.

There are a few readable words spread among the symbols. Why are they there?

Those are strings contained in the executable code. Usually, these words are part of messages provided by the program to the user while it runs. While more of an art than a science, a skilled analyst can extract valuable information by reading through these fragments.

Challenge Question: Despite the **W32.Nimda.Amm.exe** name, this executable is not the famous worm. For security reasons, this is another executable file that was renamed as **W32.Nimda.Amm.exe**. Using the word fragments displayed by **Wireshark**'s **Follow TCP Stream** window, can you tell what executable this really is?

Scrolling all the way down on that window reveals that this is the Microsoft Windows cmd.exe file.

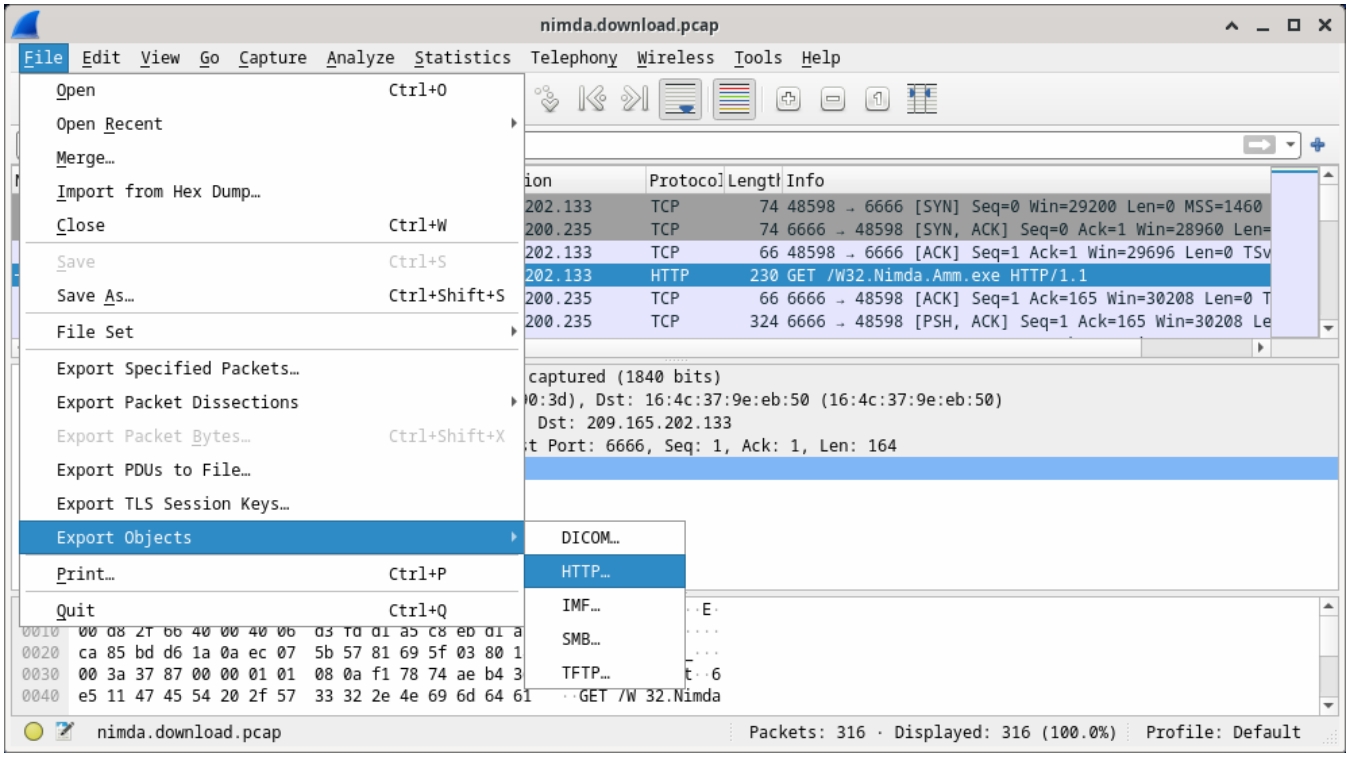
g. Click **Close** in the Follow TCP Stream window to return to the Wireshark nimda.download.pcap file.

Part 2: Extract Downloaded Files from PCAP

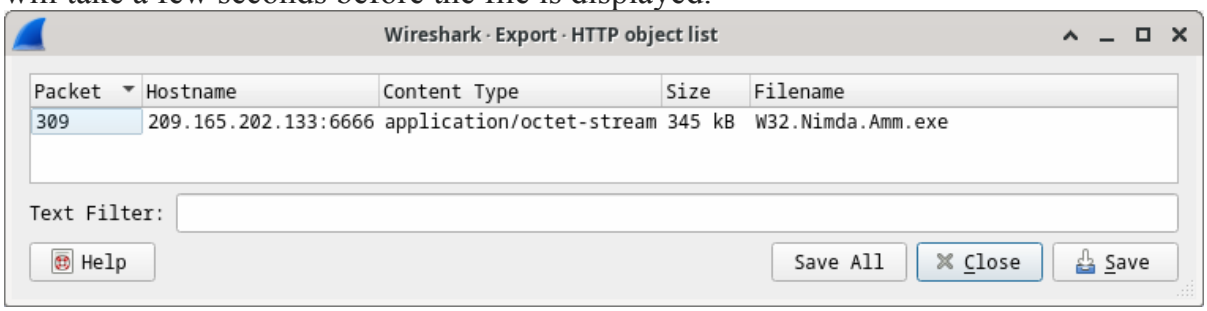
Because capture files contain all packets related to traffic, a PCAP of a download can be used to retrieve a previously downloaded file. Follow the steps below to use **Wireshark** to retrieve the Nimda malware.

a. In that fourth packet in the **download.pcap** file, notice that the **HTTP GET** request was generated from **209.165.200.235** to **209.165.202.133**. The Info column also shows this is in fact the GET request for the file.

b. With the GET request packet selected, navigate to **File > Export Objects > HTTP**, from **Wireshark**'s menu.



c. Wireshark will display all HTTP objects present in the TCP flow that contains the GET request. In this case, only the **Nimda.Amm.exe** file is present in the capture. It will take a few seconds before the file is displayed.



Why is **W32.Nimda.Amm.exe** the only file in the capture?

Because the capture was started right before the download and stopped right after. No other traffic was caught while the capture was active.
d. In the **HTTP object list** window, select the **Nimda.Amm.exe** file and click **Save As** at the bottom of the screen.
e. Click the left arrow until you see the **Home** Click Home and then click the **analyst** folder (not the analyst tab). Save the file there.
f. Return to your terminal window and ensure the file was saved. Change directory to the **/home/analyst** folder and list the files in the folder using the **ls -l**

[analyst@secOps pcaps]$ cd /home/analyst

[analyst@secOps ~]$ ls –l

total 364

drwxr-xr-x 2 analyst analyst 4096 Sep 26 2014 Desktop

drwx------ 3 analyst analyst 4096 May 25 11:16 Downloads

drwxr-xr-x 2 analyst analyst 4096 May 22 08:39 extra

drwxr-xr-x 8 analyst analyst 4096 Jun 22 11:38 lab.support.files

drwxr-xr-x 2 analyst analyst 4096 Mar 3 15:56 second_drive

-rw-r--r-- 1 analyst analyst 345088 Jun 22 15:12 W32.Nimda.Amm.exe
[analyst@secOps ~]$
Was the file saved?

Yes
g. The file command gives information on the file type. Use the file command to learn a little more about the malware, as show below:
[analyst@secOps ~]$ **file W32.Nimda.Amm.exe**
W32.Nimda.Amm.exe: PE32+ executable (console) x86-64, for MS Windows
[analyst@secOps ~]$
As seen above, **W32.Nimda.Amm.exe** is indeed a Windows executable file.
In the malware analysis process, what would be a probable next step for a security analyst?

The goal is to identify the type of malware and analyze its behavior. Therefore, the malware file should be moved to a controlled environment and execute it to watch its behavior. Malware analysis environments often rely on virtual machines and are sandboxed to avoid damage to non-test systems. Such environments usually contain tools that facilitate monitoring of the malware execution; resources usage, network connections and operating system changes are common monitored aspects.
There are also a few Internet-based malware analysis tools. VirusTotal (virustotal.com) is one example. Analysts upload malware to VirusTotal, which in turn, executes the malicious code. After execution and a number of other checks, VirusTotal returns a report to the analyst.

Lab – Exploring DNS Traffic

Objectives

- **Part 1: Capture DNS Traffic**
- **Part 2: Explore DNS Query Traffic**
- **Part 3: Explore DNS Response Traffic**

Background / Scenario

Wireshark is an open source packet capture and analysis tool. Wireshark gives a detailed breakdown of the network protocol stack. Wireshark allows you to filter traffic for network troubleshooting, investigate security issues, and analyze network protocols. Because Wireshark allows you to view the packet details, it can be used as a reconnaissance tool for an attacker.

In this lab, you will install Wireshark and use Wireshark to filter for DNS packets and view the details of both DNS query and response packets.

Required Resources

- 1 PC with internet access and Wireshark installed

Instructor Note: Using a packet sniffer such as Wireshark may be considered a breach of the security policy of the school. It is recommended that permission is obtained before running Wireshark for this lab. If using a packet sniffer such as Wireshark is an issue, the instructor may wish to assign the lab as homework or perform a walk-through demonstration.

Instructions

Part 1: Capture DNS Traffic

Step 1: Download and install Wireshark.

a. Download the latest stable version of Wireshark from www.wireshark.org. Choose the software version you need based on your PC's architecture and operating system.
b. Follow the on-screen instructions to install Wireshark. If you are prompted to install USBPcap, **do NOT** install USBPcap for normal traffic capture. USBPcap is experimental, and it could cause USB problems on your PC.

Step 2: Capture DNS traffic.

a. Start Wireshark. Select an active interface with traffic for packet capture.

b. Clear the DNS cache.

1) In Windows, enter **ipconfig /flushdns** in Command Prompt.
2) For the majority of Linux distributions, one of the following utilities is used for DNS caching: Systemd -Resolved, DNSMasq, and NSCD. If your Linux distribution does not use one of the listed utilities, please perform an internet search for the DNS caching utility for your Linux distribution.

(i) Identify the utility used in your Linux distribution by checking the status:

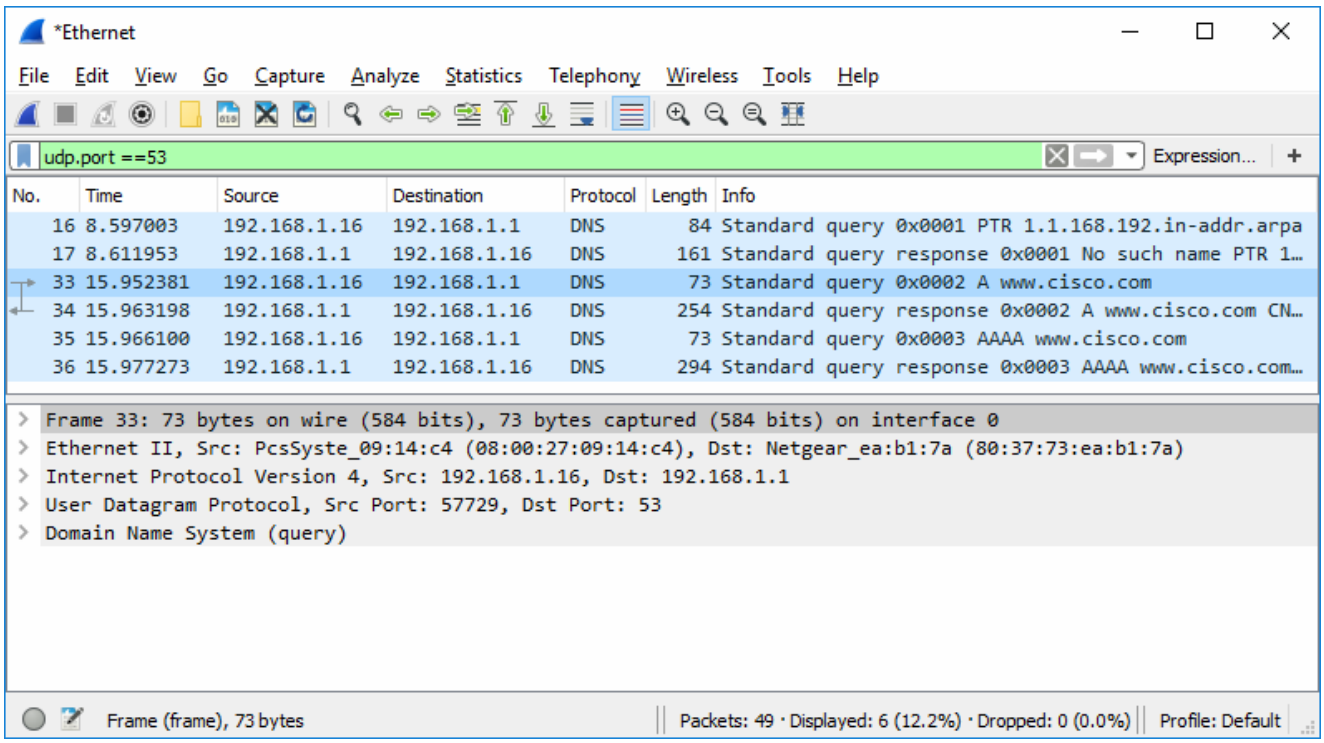
Systemd-Resolved: **systemctl status systemd-resolved.service**
DNSMasq: **systemctl status dnsmasq.service**
NSCD: **systemctl status nscd.service**
(ii) If you are using system-resolved, enter systemd-resolve –flush-caches to flush the cache for Systemd-Resolved before restarting the service. The following commands restart the associated service using elevated privileges:

Systemd-Resolved: **sudo systemctl restart systemd-resolved.service**
DNSMasq: **sudo systemctl restart dnsmasq.service**
NSCD: **sudo systemctl restart nscd.service**

3) For the macOS, enter **sudo killall -HUP mDNSResponder** to clear the DNS cache in the Terminal. Perform an internet search for the commands to clear the DNS cache for an older OS.
c. At a command prompt or terminal, type **nslookup** enter the interactive mode.
d. Enter the domain name of a website. The domain name www.cisco.com is used in this example.
e. Type **exit** when finished. Close the command prompt.
f. Click **Stop capturing packets** to stop the Wireshark capture.

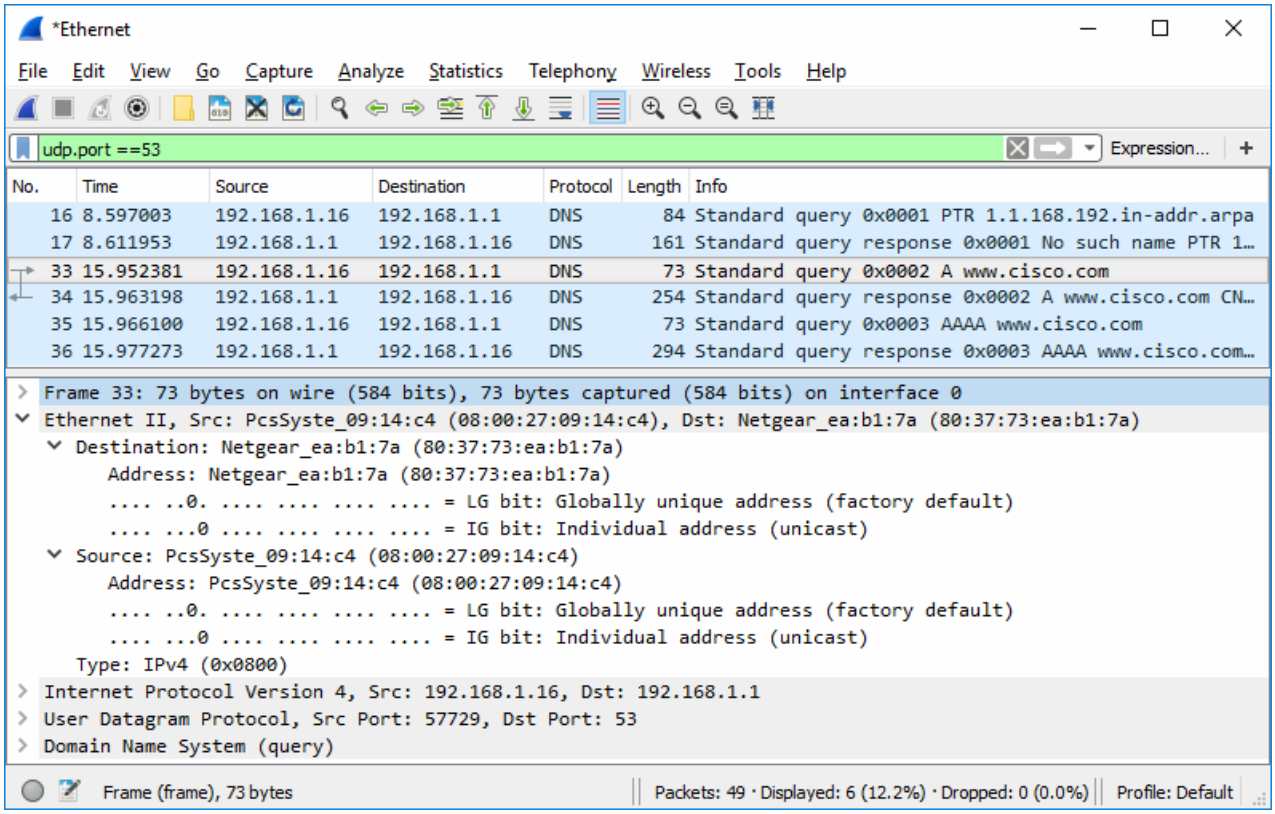
Part 2: Explore DNS Query Traffic

a. Observe the traffic captured in the Wireshark Packet List pane. Enter **udp.port == 53** in the filter box and click the arrow (or press enter) to display only DNS packets.
Note: The provided screenshots are just examples. Your output maybe slightly different.



b. Select the DNS packet contains **Standard query** and **A www.cisco.com** in the Info column.
c. In the Packet Details pane, notice this packet has Ethernet II, Internet Protocol Version 4, User Datagram Protocol and Domain Name System (query).

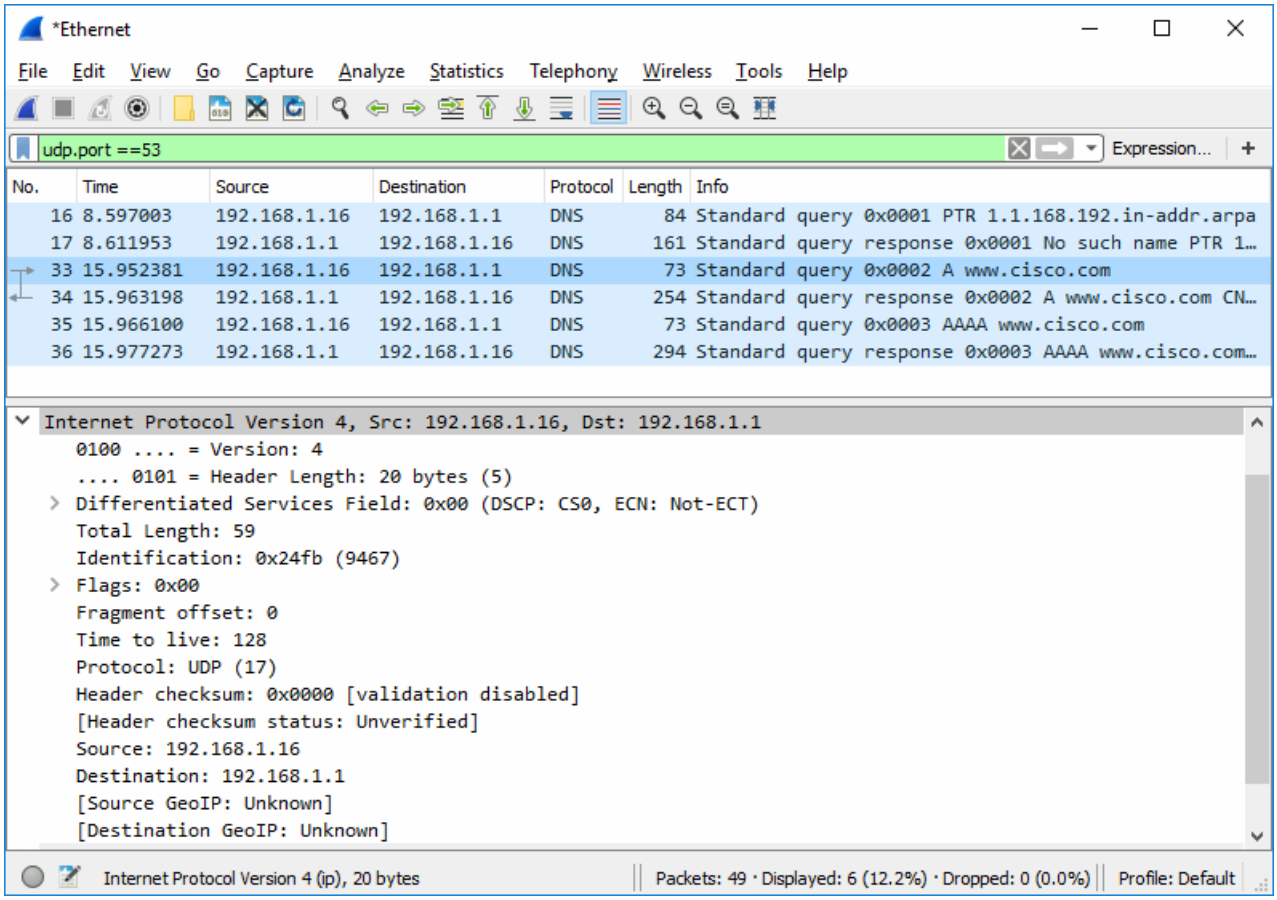
d. Expand **Ethernet II** to view the details. Observe the source and destination fields.



What are the source and destination MAC addresses? Which network interfaces are these MAC addresses associated with?

In this example, the source MAC address is associated with the NIC on the PC and the destination MAC address is associated with the default gateway. If there is a local DNS server, the destination MAC address would be the MAC address of the local DNS server.

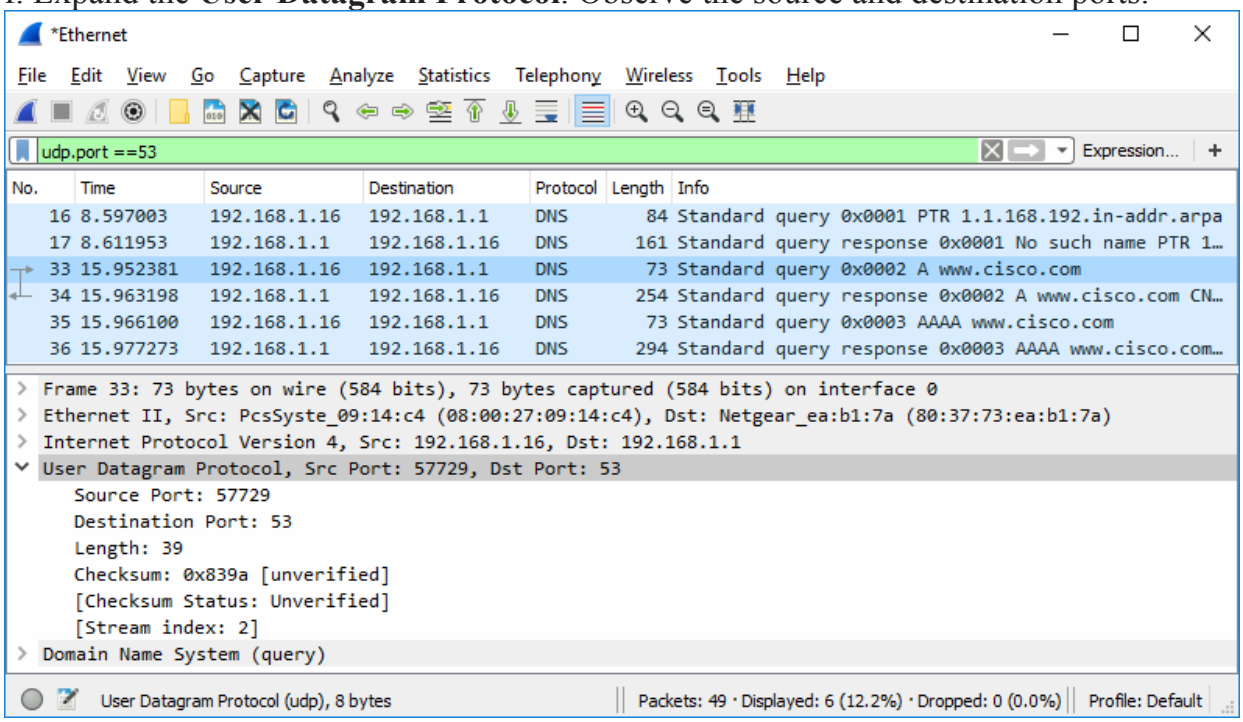
e. Expand **Internet Protocol Version 4**. Observe the source and destination IPv4 addresses.



What are the source and destination IP addresses? Which network interfaces are these IP addresses associated with?

In this example, the source IP address is associated with the NIC on the PC and the destination IP address is associated with the default gateway.

f. Expand the **User Datagram Protocol**. Observe the source and destination ports.



What are the source and destination ports? What is the default DNS port number?

The source port number is 577729 and the destination port is 53, which is the default DNS port number.

g. Determine the IP and MAC address of the PC.

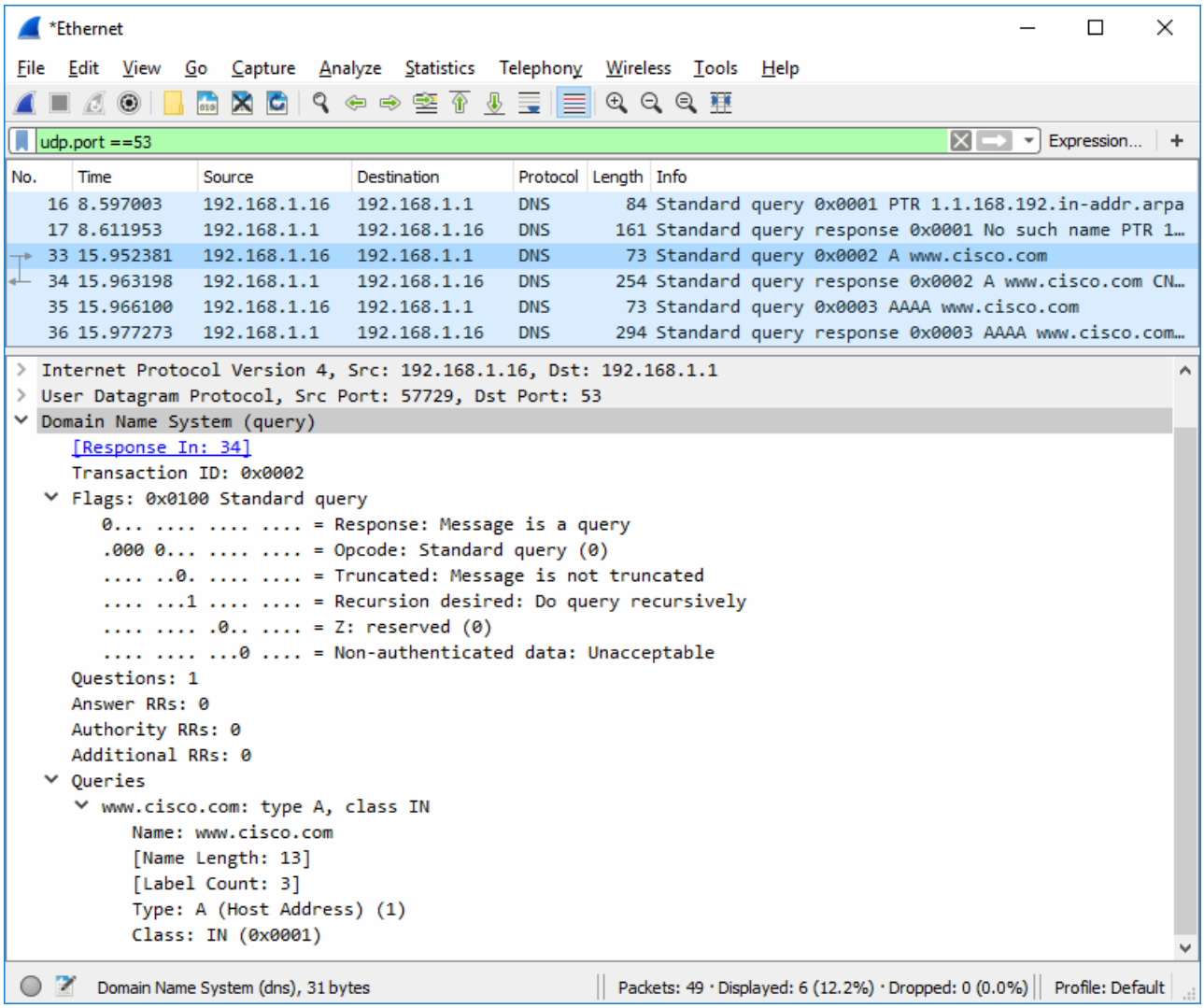
1. In a Windows command prompt, enter **arp –a** and **ipconfig /all** to record the MAC and IP addresses of the PC.
2. For Linux and macOS PC, enter **ifconfig** or **ip address** in a terminal.

Compare the MAC and IP addresses in the Wireshark results to the IP and MAC addresses. What is your observation?

The IP and MAC addresses captured in the Wireshark results are the same as the addresses listed in ipconfig /all command.

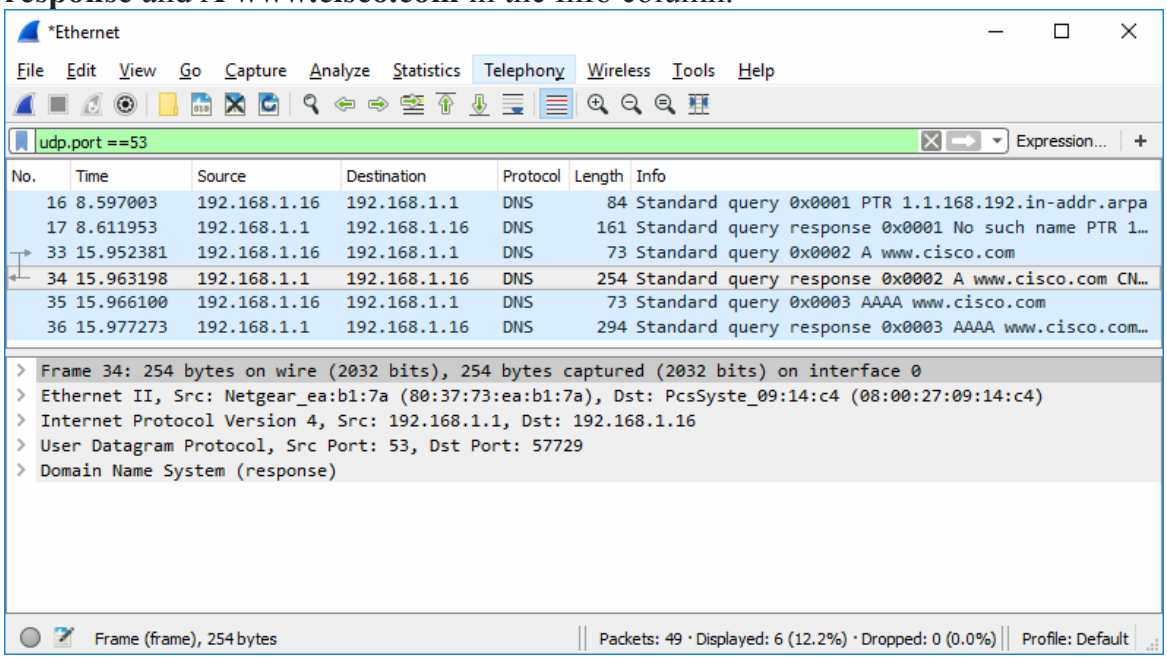
h. Expand **Domain Name System (query)** in the Packet Details pane. Then expand the **Flags** and **Queries**.

i. Observe the results. The flag is set to do the query recursively to query for the IP address to www.cisco.com.



Part 3: Explore DNS Response Traffic

a. Select the corresponding response DNS packet has **Standard query response** and **A www.cisco.com** in the Info column.



What are the source and destination MAC and IP addresses and port numbers? How do they compare to the addresses in the DNS query packets?

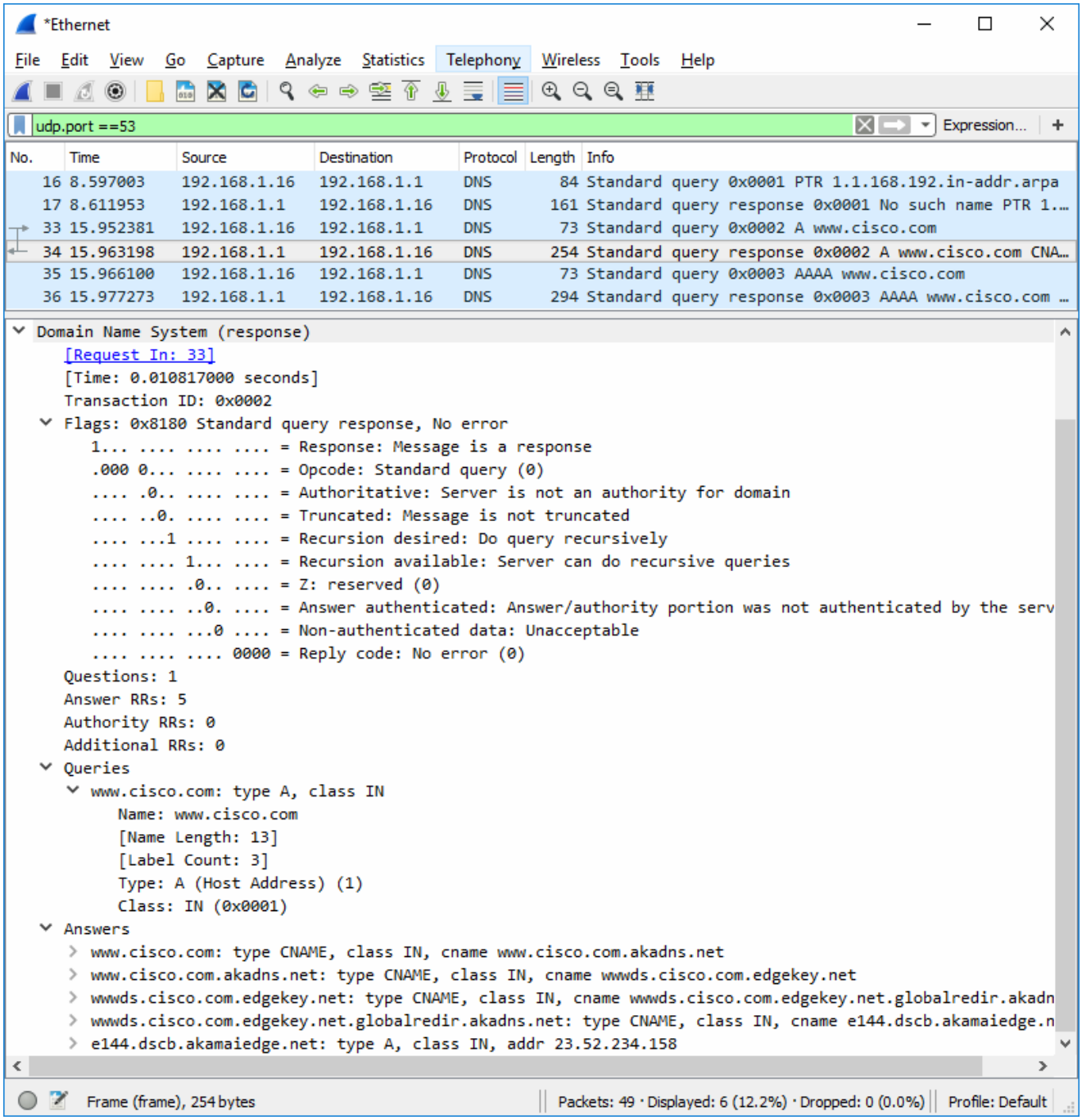
The source IP, MAC address, and port number in the query packet are now destination addresses. The destination IP, MAC address, and port number in the query packet are now source addresses.

b. Expand **Domain Name System (response)**. Then expand the **Flags, Queries,** and **Answers.**

c. Observe the results.

Can the DNS server do recursive queries?

Yes, the DNS can handle recursive queries.



d. Observe the CNAME and A records in the Answers details.

Configure Your Own Syslog Server

Rsyslog is a **rocket-fast system for log** processing and is commonly used for any kind of system logging. We use a Ubuntu server 20.04 LTS distribution to show you how to configure your own syslog server to receive your CDN logs in real time.

Syslog server installation#

Update the packages list and install the latest version of rsyslog.

1. `apt update`
2. `apt install rsyslog`

Syslog server configuration#

Configure rsyslog to receive UDP logs and define a filter where you want to store the logs.

1. Open the `rsyslog.conf` file and add the following lines.

```
vi /etc/rsyslog.conf

# provides UDP syslog reception
module(load="imudp")
```

5. Create and open your custom config file.

```
vi /etc/rsyslog.d/00-custom.conf
# Templates
template(name="ReceiveFormat" type="string" string="%msg:39:$%\n")

# UDP ruleset mapping
input(type="imudp" port="514" ruleset="customRuleset")

# Custom ruleset
ruleset(name="customRuleset") {
    if ($msg contains '366c3df6-93dd-4ec0-a218-aec9d191c59e') then {
        /var/log/cdn.log;ReceiveFormat
        stop
    }
}
```

Replace `366c3df6-93dd-4ec0-a218-aec9d191c59e` with your own custom token. Your token values must be between 8 to 45 characters.

Use the following regex expression with regex101 to validate the token value you define:

```
^[a-zA-Z0-9\-]*$
```

You may use any letters from `a-zA-Z`, `-` and numbers from `0-9` when creating your token.

19. Restart the rsyslog process.

```
systemctl restart rsyslog
```

20. Configure Log Forwarding in the KeyCDN dashboard with your syslog server details.
21. Verify if you are receiving the logs (log forwarding starts within 5 minutes).

```
tail -f /var/log/cdn.log
```

Troubleshooting commands#

- `systemctl status rsyslog` Verify that rsyslog is running.

- netstat -na | grep ":<defined port>" Is rsyslog listening on the right port?
- # netstat -na | grep :514
  udp 0 0 0.0.0.0:514 0.0.0.0:*
- tcpdump port <defined port> Are you receiving any packet on the defined port?
- # tcpdump port 514
- tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
- listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
- 11:20:53.066938 IP keycdn-syslog.37960 > your-server.syslog: [|syslog]
- ^C
- 1 packet captured
- 1 packet received by filter
  0 packets dropped by kernel
- tail -f /path/to/your/logfile Check if you get new log entries

Logging Syslog Messages to Remote Linux Server:

Step 1

Configuring the Linux Syslog Server (server-syslog) to receive messages. By default, Syslog does not expect to receive messages from remote clients. Here is how to configure your Linux server to start listening for these messages. Syslog checks its /etc/syslog.conf file to determine the expected names and locations of the log files it should create. It also checks the file /etc/sysconfig/syslog to determine the various modes in which it should operate. Syslog will not listen for remote messages unless the SYSLOGD_OPTIONS variable in this file has an -r included in it as shown below. Here is an example of how to configure the /etc/sysconfig/syslog file to receive the Syslog messages.

Options to syslogd

-m 0 disables 'MARK' messages.

-r enables logging from remote machines

-x disables DNS lookups on messages received with -r

See syslogd(8) for more details SYSLOGD_OPTIONS="-m 0 -r"

Options to klogd

-2 prints all kernel oops messages twice; once for klogd to decode, and

once for processing with 'ksymoops'

-x disables all klogd processing of oops messages entirely

See klogd(8) for more details KLOGD_OPTIONS="-2"

Here is how the /etc/syslog.conf file should look on the Syslog Server: *.debug /var/log/messages

Step 2

You must restart Syslog on the server for the changes to take effect. The server now listens on UDP port 514, which you can verify using either one of the following netstat command variations:
/etc/init.d/syslog restart

[root@server-syslog tmp]#

Step 3

Configuring the Linux Client:

a. The Syslog server (server-syslog) is now expecting to receive Syslog messages.

b. Configure your remote Linux client to send messages to the Syslog server.

This is done by editing the /etc/hosts file on the Linux client named oer-host:

– Determine the IP address and fully qualified hostname of your remote logging host.

– Add an entry in the /etc/hosts file in the format:

IP-address fully-qualified-domain-name hostname "loghost"

For example:

10.10.10.1 server-syslog.domain.com server-syslog loghost

Now your /etc/hosts file has a nickname of “loghost” for the server-syslog server.

Step 4

Edit the /etc/syslog.conf file to send Syslog messages to your new “loghost” nickname.

*.debug @loghost

*.debug /var/log/messages

In this example all information messages and higher are being logged to both server-syslog server (“loghost”) and the local /var/log/messages file.

Step 5

Restart Syslog:

/etc/init.d/syslog restart

Step 6

Run a test to verify that the destination Syslog server is receiving the messages in the /var/log/messages file. Every Configuration Engine message has the “OER_MC” tag attached.

Summary

The following files must be modified on the destination server:

• /etc/sysconfig/syslog

• /etc/syslog.conf

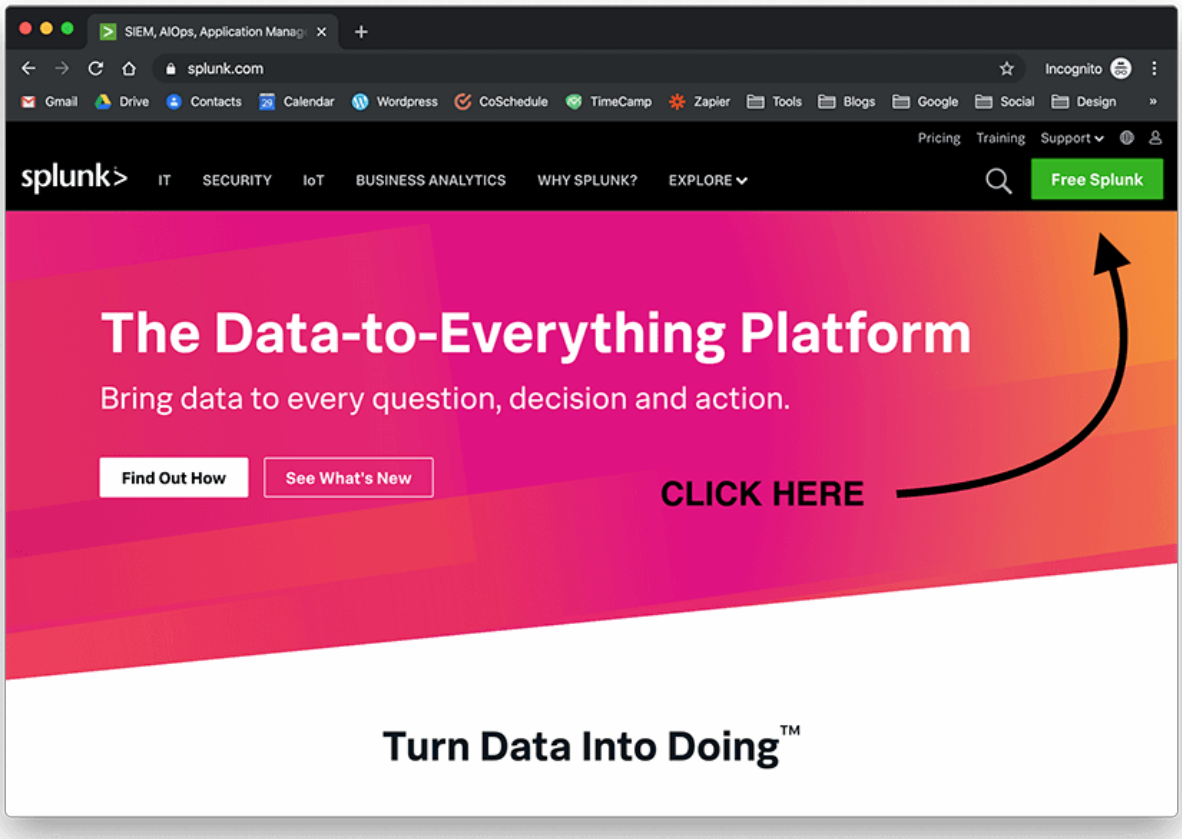
The following files must be modified on the client sending the messages to the server:

• /etc/hosts

• /etc/syslog.conf

Installing Splunk in Linux

Download the Splunk Installation Package



To install Splunk on Linux, you will need to download the appropriate installation package from the official Splunk website. You can download the package from the following link: https://www.splunk.com/en_us/download/splunk-enterprise.html.

Once you have downloaded the package, you should move it to a directory on your Linux machine where you have write permissions.

How to Install Splunk for the First Time

To install Splunk on Linux, you will need to run the installation script as the root user. To do this, open a terminal window and navigate to the directory where you saved the installation package. Then, enter the following command:

sudo ./splunk-<version>-linux-<architecture>.rpm

Replace **<version>** and **<architecture>** with the appropriate values for your installation package. This will start the installation process, which may take a few minutes to complete.

1. Configure Splunk

Once the installation is complete, you can configure Splunk to suit your needs. By default, Splunk will be configured to run on port 8000, so you can access it by opening a web browser and navigating to **http://<your-server>:8000**

You will be prompted to set up a Splunk admin account, which you can use to log in to the Splunk web interface. Once you have logged in, you can configure Splunk to index and search the data you want to analyze.

2. Start Splunk

To start Splunk on Linux, you will need to run the following command:

sudo /opt/splunk/bin/splunk start

This will start the Splunk service, which will begin indexing and searching the data you have configured to analyze. You can monitor the status of the service by running the following command: **sudo /opt/splunk/bin/splunk status**

This will display information about the current status of the Splunk service, including whether it is running or stopped.

3. Stop Splunk

If you need to stop the Splunk service for whatever reason, you can do so by running the following command: **sudo /opt/splunk/bin/splunk stop**

This will stop the Splunk service and prevent it from indexing or searching for any new data until you start it again.

4. Upgrade Splunk

To upgrade Splunk on Linux, you will need to download the latest version of the installation package from the official Splunk website.

Once you have downloaded the package, you can upgrade Splunk by running the following command: **sudo rpm -Uvh splunk-<version>-linux-<architecture>.rpm**

Replace **<version>** and **<architecture>** with the appropriate values for your installation package. This will upgrade Splunk to the latest version and preserve your existing configuration and data.

How to Install Splunk with the Tar file

1. Download the appropriate Splunk TAR file for your operating system from the official Splunk website: https://www.splunk.com/en_us/download.html

2. Once the download is complete, navigate to the directory where the TAR file is saved.
3. Extract the contents of the TAR file to a directory of your choice using the following command: **tar xvzf splunk-<version>-linux-<architecture>.tgz -C <directory>**

Note: Replace <version>, <architecture>, and <directory> with the actual version number, architecture, and directory where you want to extract the files.

4. After extraction is complete, navigate to the directory where Splunk was extracted.

5. Start the Splunk service using the following command: **./splunk start**

6. Once the service has started, access the Splunk Web interface by navigating to **http://<hostname>:8000** in a web browser, where <hostname> is the hostname or IP address of the server where you installed Splunk.

7. Follow the on-screen instructions to complete the initial Splunk setup.

That's it! You should now have Splunk successfully installed on your Linux system using the TAR file.

How to Install Splunk on Red Hat Enterprise Linux (RPM)

The installation process is a little bit more involved on Red Hat Enterprise Linux, but still easy and manageable. With this distribution, you have to create a new user in order to safely operate Splunk.

We do not recommend installing Splunk on your root user as it will compromise the rest of the system. Follow these steps to install Splunk on RHEL:

1. Download the .rpm file

```
wget -O splunk-8.0.0-1357bef0a7f6-linux-2.6-x86_64.rpm 'https://www.splunk.com/bin/splunk/DownloadActivityServlet?architecture=x86_64&platform=linux&version=8.0.0&product=splunk&filename=splunk-8.0.0-1357bef0a7f6-linux-2.6-x86_64.rpm&wget=true'
```

2. Create a new Splunk user

```
groupadd splunk useradd -d /opt/splunk -m -g splunk splunk
```

3. Create the following directory

```
mkdir /opt/installers
```

4. Copy the downloaded .rpm file to the new directory

```
Cp splunk-8.0.0-1357bef0a7f6-linux-2.6-x86_64.rpm /opt/installers/
```

5. Change ownership

```
chown -R splunk: /opt/splunk/ /opt/installers
```

6. Switch user

```
su – splunk
```

7. Change directory

```
cd /opt/installers
```

8. Install Splunk

```
rpm -i splunk-8.0.0-1357bef0a7f6-linux-2.6-x86_64.rpm
```

9. Start Splunk quickly (accept license automatically)

```
/opt/splunk/bin/splunk start –accept-license
```

10. Enter an administrator username

```
Please enter an administrator username:
```

11. Enter and confirm a password

```
Password must contain at least: * 8 total printable ASCII character(s). Please enter a new password:
Please confirm new password:
```

12. Enable at boot

```
/opt/splunk/bin/splunk enable boot-start
```

Install Elasticsearch (ELK Stack) on RHEL:

How to Install ELK Stack on RHEL

Let's begin by installing the ELK stack on the central server (which is our RHEL 9 system), the same instructions apply to RHEL-based distributions such as Rocky and Alma Linux.

Let's understand with a brief explanation on what each component does:

- Elasticsearch stores the logs that are sent by the clients.
- Logstash processes those logs.
- Kibana provides the web interface that will help us to inspect and analyze the logs.

Install the following packages on the central server. First off, we will install Java JDK version 21, the latest one at the time of this writing), which is a dependency of the ELK components.

You may want to check first in the Java downloads page here to see if there is a newer update available.

```
yum update

cd /opt
```

Time to check whether the installation completed successfully:

```
java -version
```

To install the latest versions of Elasticsearch, Logstash, and Kibana, we will have to create repositories manually as follows:

Install Elasticsearch in RHEL

1. Import the Elasticsearch public GPG key to the rpm package manager.

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

2. Insert the following lines to the repository configuration file `elasticsearch.repo`:

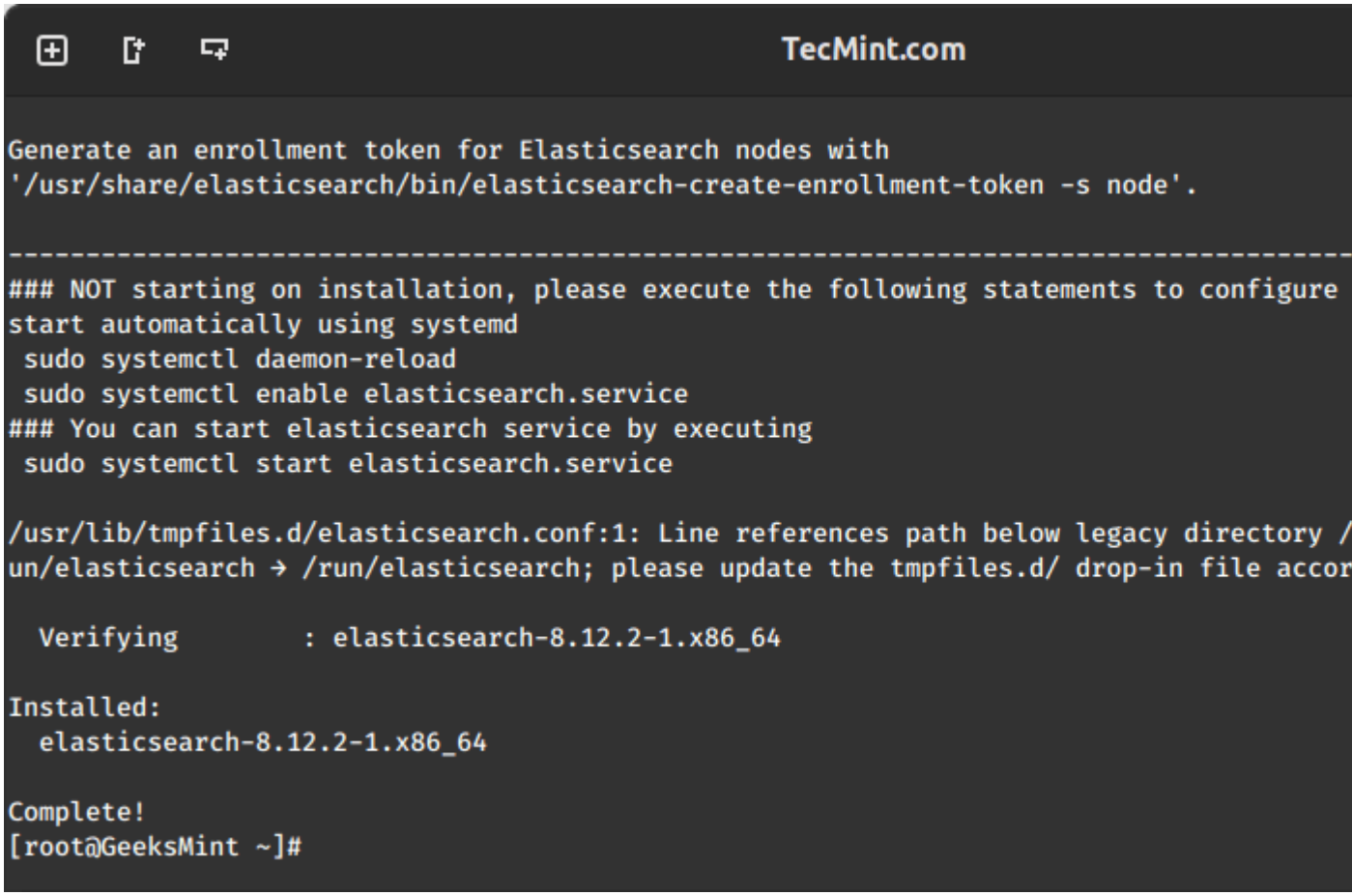
/etc/yum.repos.d/elasticsearch.repo

```
[elasticsearch]

name=Elasticsearch repository for 8.x packages

baseurl=https://artifacts.elastic.co/packages/8.x/yum

gpgcheck=1
```

3. Install the Elasticsearch package.

```
yum install --enablerepo=elasticsearch elasticsearch
```

When the installation is complete, you will be prompted to start and enable elasticsearch:



Install Elasticsearch in Linux

4. Start and enable the service.

```
systemctl daemon-reload

systemctl enable elasticsearch
```

5. Allow traffic through TCP port 9200 in your firewall:

```
firewall-cmd --add-port=9200/tcp
```

6. Check if Elasticsearch responds to simple requests over HTTP using curl command:

```
curl -X GET http://localhost:9200
```

The output of the above command should be similar to:

Verify Elasticsearch Installation

Make sure you complete the above steps and then proceed with Logstash. Since both Logstash and Kibana share the Elasticsearch GPG key, there is no need to re-import it before installing the packages.

Setting up ELK Stack in Linux

Introduction

Elasticsearch, Logstash, and Kibana are the three open-source tools that make up the ELK Stack.

For the purpose of identifying issues with servers or applications, the ELK stack offers centralized logging. It enables comprehensive log searches in one location.

Elasticsearch

It was created using Java. An open source search engine. It uses JSON format to get data, search and also save it. It comes in a HTTP dashboard web interface (Kibana).

Logstash

An open-source tool to manage logs and different events. It saves log data in JSON format then saves it in Elasticsearch.

Kibana

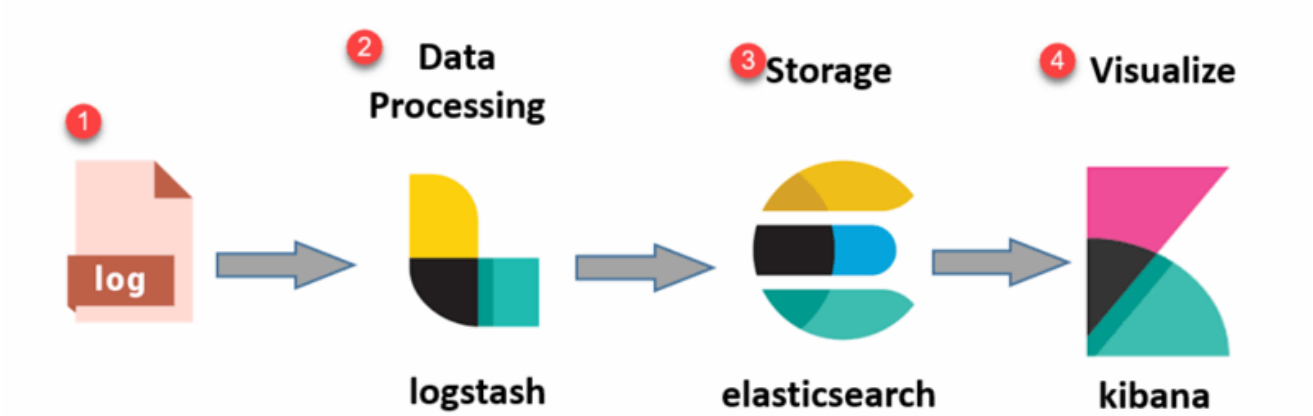
An open-source tool that let's you visualize the Elaticsearch data in the format of a dashboard.

Beats

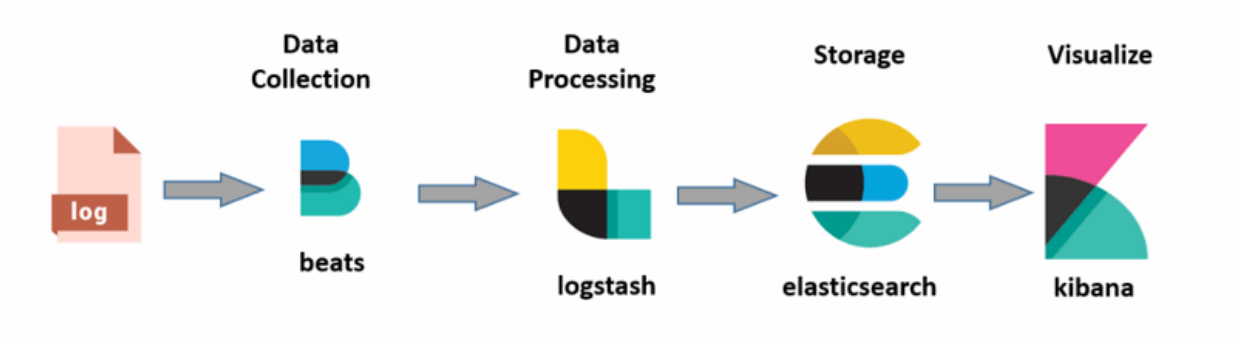
A data transfer service which can be installed on the Client and send a large amount of data from the Client to the Logstash & Elasticsearch server.

ELK Stack Architecture

A simple ELK Stack looks like



With Beats it looks like



If we are dealing with very large amounts of data, we can add kafka and for security we can include Nginx as well

Setting up the ELK Stack

As it is created from Java, we will need a JVM environment to run ELK. (4GB+ RAM required)

🔔 Selinux can interrupt with retrieving the logs using ELK Stack so we will be disabling that

vi /etc/sysconfig/selinux
setenforce 0
reboot

getenforce
Disabled

Step 1 👉 JDK Installation

As Elasticsearch is based on Java, we need the JDK.

rpm -qa | grep java

We can search for java packages and if we aren't able to find any, we can just install it using yum

yum -y install java-1.8 open-jdk*

After the installation, we can check the java version

java -version
openjdk version "1.8.0_362"

Step 2 ☞ Elasticsearch Installation

We will first create a directory named ELK and work inside there

mkdir ./ELK

cd ELK

Now, we need to download the Elasticsearch package

wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.6.1-x86_64.rpm

📯 If you don't have wget installed, you can install it first using yum

Importing the GPG Key

rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch

📯 The GPG Key ensures that the RPM package file was not altered

We can now install the package that we downloaded

rpm -ivh elasticsearch-7.6.1-x86_64.rpm
rpm -qa | grep elasticsearch
elasticsearch-7.6.1-1.x86_64

We have to edit some configurations by accessing the elasticsearch config files and its yml file

vi /etc/elasticsearch/elasticsearch.yml

43 bootstrap.memory_lock: true
55 network.host: localhost
59 http.port: 9200
vi /usr/lib/systemd/system/elasticsearch.service

33 # Memory mlockall
34 LimitMEMLOCK=infinity
vi /etc/sysconfig/elasticsearch
46 MAX_LOCKED_MEMORY=unlimited

📯 As I am setting up the Elasticsearch on the same system as the Logstash, I have set the network host as the localhost

Also enabling the mlockall function to change the memory usage setting for all operations to use physical memory

The mlockall locks all pages mapped to the address of the call process. It is sometimes used to ensure that the Java virtual machine (JVM) running Elasticsearch does not run out of memory, as this can cause the JVM to crash.

We will have to reload the daemon to make the mlockall setting configured correctly

```
systemctl daemon-reload
systemctl enable elasticsearch
systemctl start elasticsearch
```

Enabled and started the elasticsearch daemon as well

To confirm JAVA process is running on 9200 port

```
netstat -antp | grep 9200
```

Next, we will use curl to confirm mlockall status

```
curl -XGET 'localhost:9200/_nodes?filter_path=**.mlockall&pretty'
```

```
{
  "nodes" : {
    "1JrdEndhRaS5Le2JUxiDsw" : {
      "process" : {
       "mlockall" : true
      }
    }
  }
}
```

Also, we can confirm if the elasticsearch is running or not

```
curl -XGET 'localhost:9200/?pretty'
```

Step 3 ☞ Nginx & Kibana Installation

Downloading the kibana rpm package

```
wget https://artifacts.elastic.co/downloads/kibana/kibana-7.6.1-x86_64.rpm
```

Installing the Kibana package

```
rpm -ivh kibana-7.6.1-x86_64.rpm
```

Confirming if the package was successfully installed

```
rpm -qa | grep kibana
kibana-7.6.1-1.x86_64
```

We will edit the configuration files for kibana now

```
vi /etc/kibana/kibana.yml

2 server.port: 5601
7 server.host: "0.0.0.0"
28 elasticsearch.hosts: ["http://localhost:9200"]
```

Proceeded with all-network setup to set up the production port and allow external access (only certain hosts can be specified)

Enabling and starting the kibana daemon

```
systemctl enable kibana
systemctl start kibana
```

Confirming kibana service status

```
netstat -antp | grep 5601
tcp        0      0 0.0.0.0:5601            0.0.0.0:*               LISTEN      8691/node
```

It will take some time for the tcp LISTEN node to show

Now we have to install Nginx

We will be using the epel repository as well so

```
yum -y install epel-release
yum -y install nginx httpd-tools
```

The Kibana Service displays information on the screen through the Reverse WEB Proxy Server. That is why we will configure the Reverse WEB Proxy Server with the Nginx Web Server

*A **forward proxy** sits in front of clients and a **reverse proxy** sits in front of servers. Both types of proxies serve as intermediaries between clients and servers*

Editing some entries in the configuration file for Nginx

```
vi /etc/nginx/nginx.conf

38 server {
39 listen 80 default_server;
40 listen [::]:80 default_server;
41 server_name _;
42 root /usr/share/nginx/html;
43
44 # Load configuration files for the default server block.
45 include /etc/nginx/default.d/*.conf;
46
47 location / {
48 }
49
50 error_page 404 /404.html;
51 location = /40x.html {
52 }
53
54 error_page 500 502 503 504 /50x.html;
55 location = /50x.html {
56 }
57 }
```

We have to delete the above lines from the Nginx config file as we will be adding a virtual host

Including the following in kibana configuration file

```
server {
 listen 80;
 server_name example.com;
 auth_basic "Restricted Access";
 auth_basic_user_file /etc/nginx/.kibana-user;
 location / {
 proxy_pass http://localhost:5601;
 proxy_http_version 1.1;
 proxy_set_header Upgrade $http_upgrade;
 proxy_set_header Connection 'upgrade';
 proxy_set_header Host $host;
 proxy_cache_bypass $http_upgrade;
 }
}
```

This will set up virtual host for Reverse Proxy Server operation

Next, we have to set login credentials for kibana access user

```
htpasswd -c /etc/nginx/.kibana-user Admin
New password:
Re-type new password:
Adding password for user Admin
```

Verifying Nginx settings

```
nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Enabling and starting nginx. We can also confirm if its working

```
systemctl enable nginx
systemctl start nginx
netstat -antp | grep nginx
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      8889/nginx: master
```

Step 4 ☞ Logstash Installation

Downloading and installing the rpm package

```
wget https://artifacts.elastic.co/downloads/logstash/logstash-7.6.1.rpm

rpm -ivh logstash-7.6.1.rpm

rpm -qa | grep logstash
logstash-7.6.1-1.noarch
```

Now, we need to edit the openssl config file

```
226 [ v3_ca ]
227 # Server IP Address
228 subjectAltName = IP: <Your-IP-Address>
```

📯 When sending log information using SSL/TLS, it is recommended to encrypt and transmit log information

Issuing the openssl certificate

```
openssl req -config /etc/pki/tls/openssl.cnf -x509 -days 3650 -batch -nodes -newkey rsa:2048 -keyout /etc/pki/tls/private/logstash-forwarder.key -out /etc/pki/tls/certs/logstash-forwarder.crt
```

After generating and verifying the public key certificate and private key used in SSL/TLS, we generated the public key certificate and private key by referring to the SSL/TLS configuration file

ls -ld /etc/pki/tls/certs/logstash-forwarder.crt
-rw-r--r-- 1 root root 1241 2월 13 13:09 /etc/pki/tls/certs/logstash-forwarder.crt

ls -ld /etc/pki/tls/private/logstash-forwarder.key
-rw-r--r-- 1 root root 1704 2월 13 13:09 /etc/pki/tls/private/logstash-forwarder.key

Specifying expiration date and RSA Bit value for each key through additional options

```
# Using Filebeat to determine which format to accept data sent from the Client
input {
 beats {
            client_inactivity_timeout => 600
            port => 5044
            ssl => true
            ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
            ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
              }
      }


# Logstash supports various Filter Plugins (mainly using "grok" Filter Plugins)
 filter {
 if [type] == "syslog" {
 grok {
 match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}
%{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:
\[%{POSINT:syslog_pid}\])?:
%{GREEDYDATA:syslog_message}" }
 }
 date {
 match => [ "syslog_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]
 }
 }
}

# Proceeding with settings to transfer data collected by Logstash to Elasticsearch
output {
 elasticsearch {
 hosts => ["localhost:9200"]
 index => "syslog-%{+YYYY.MM.dd}"
 }
}
```

Finally, we just need to enable and start the logstash

```
systemctl enable logstash
systemctl start logstash
```

And also, adding 5044 port TCP in the firewall along with the http protocol

```
firewall-cmd --permanent --add-service=http
firewall-cmd --permanent --add-port=5044/tcp
firewall-cmd --reload
```

We can check the logstash if its working or not

```
netstat -antp | grep 5044
tcp6       0      0 :::5044                 :::*                    LISTEN      12868/java
```

🔔 *It will take some time for logstash to start working*

Step 5 ☞ Logstash Client Filebeat Installation

We need the tls certificate inside the Logstash Client, so we will copy the public key certificate from the ELK Server to the Client Linux using scp

```
# The Client with IP (192.168.1.129)
scp root@192.168.1.128:/etc/pki/tls/certs/logstash-forwarder.crt ./
root@192.168.1.128's password:
logstash-forwarder.crt                                        100% 1241
1.0MB/s   00:00

# Moving the `.crt` file to the `/etc/pki/tls/certs` directory

mv ./logstash-forwarder.crt /etc/pki/tls/certs/
```

Again importing the GPG Key for elasticsearch and downloading the filebeat rpm

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch

wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.6.1-x86_64.rpm

# Installing the filebeat package
rpm -ivh filebeat-7.6.1-x86_64.rpm

rpm -qa | grep filebeat
filebeat-7.6.1-1.x86_64
```

As usual, we have to edit some entries in the config file

vi /etc/filebeat/filebeat.yml
Enabling the Filebeat and also declaring what logs we want to see
24 enabled: true
27 paths:
28 - /var/log/*.log
29 - /var/log/secure
30 - /var/log/messages

Commenting the Elasticsearch output as we will be getting Logstash output
150 #----------------------- Elasticsearch output ------------------------------
151 #output.elasticsearch:
152 # Array of hosts to connect to.
153 #hosts: ["localhost:9200"]

Uncommenting the logstash output + declaring the ELK Host IP and the ssl certificate location
163 #------------------------------ Logstash output -------------------------------
164 output.logstash:
165 # The Logstash hosts
166 hosts: ["<ELK HOST IP>:5044"]
167 ssl.certificate_authorities: ["/etc/pki/tls/certs/logstash-forwarder.crt"]
168 bulk_max_size: 1024 # Specifying the maximum number of events that can be sent at a time

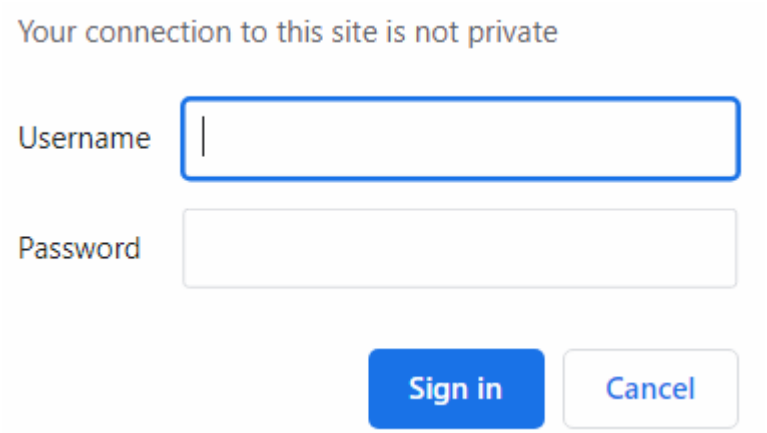
We now just have to enable and start the filebeat daemon

systemctl enable filebeat
systemctl start filebeat
systemctl status filebeat
● filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
Loaded: loaded (/usr/lib/systemd/system/filebeat.service; enabled; vendor preset: disabled)
Active: active (running) since Mon 2023-02-13 14:43:53 KST; 2min 49s ago

If we check the connection from the ELK Host

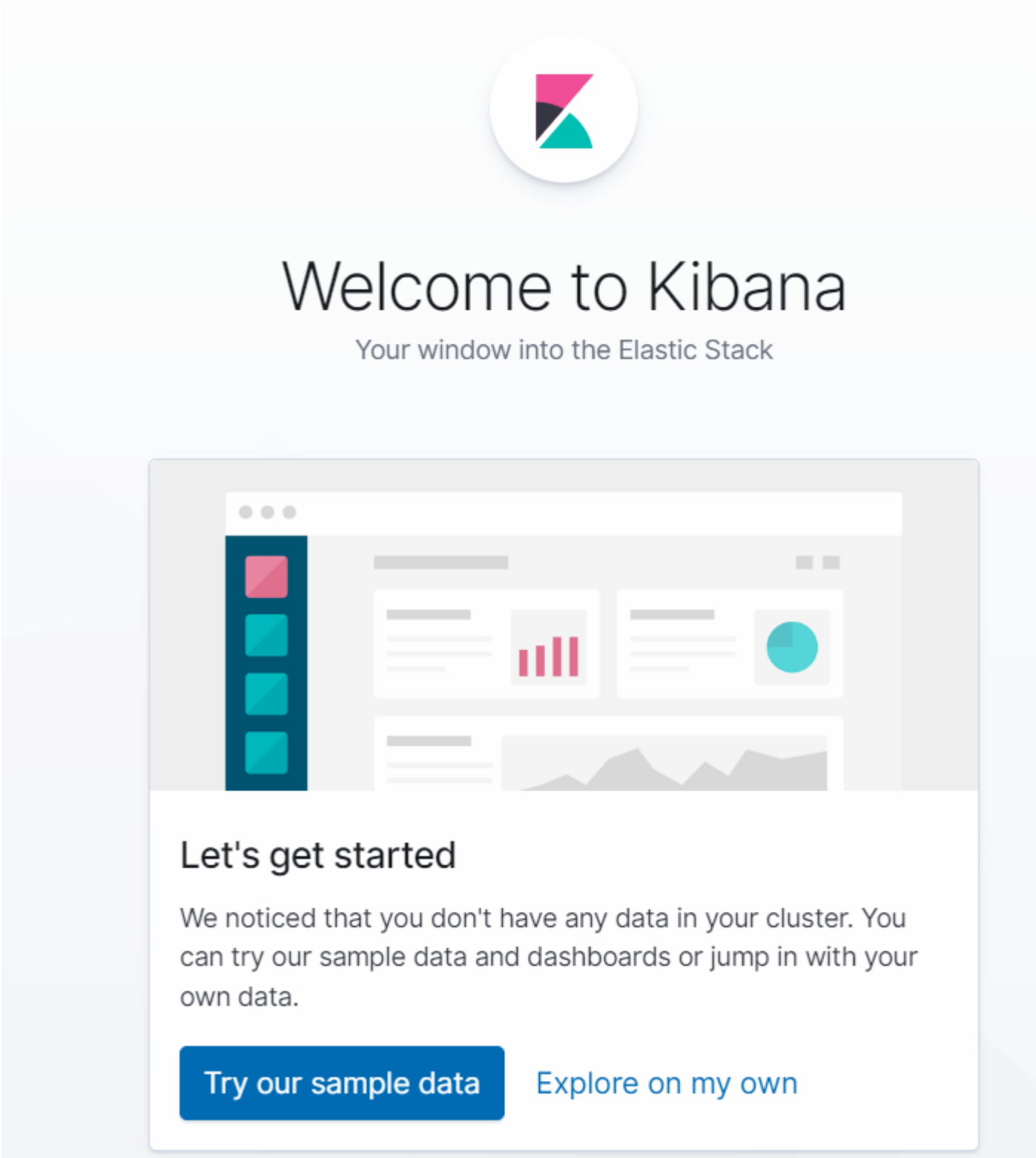
netstat -antp | grep 5044
tcp6 0 0 :::5044 :::* LISTEN 17021/java
tcp6 0 0 <ELK HOST>:5044 <Client>:35030 ESTABLISHED 17021/java

Now from your browser, you can open the ELK Host server by using its IP address:

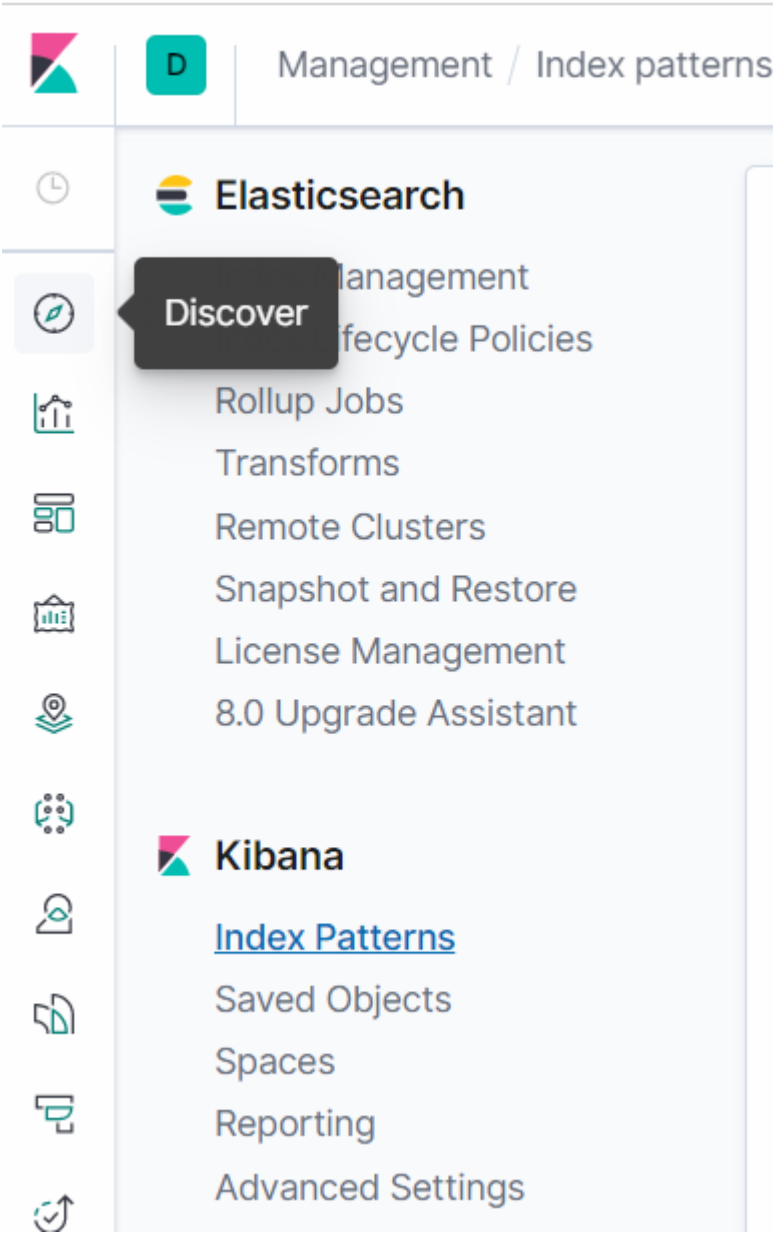


Using the username and password that we set before

Once we are in, we will be able to see this screen

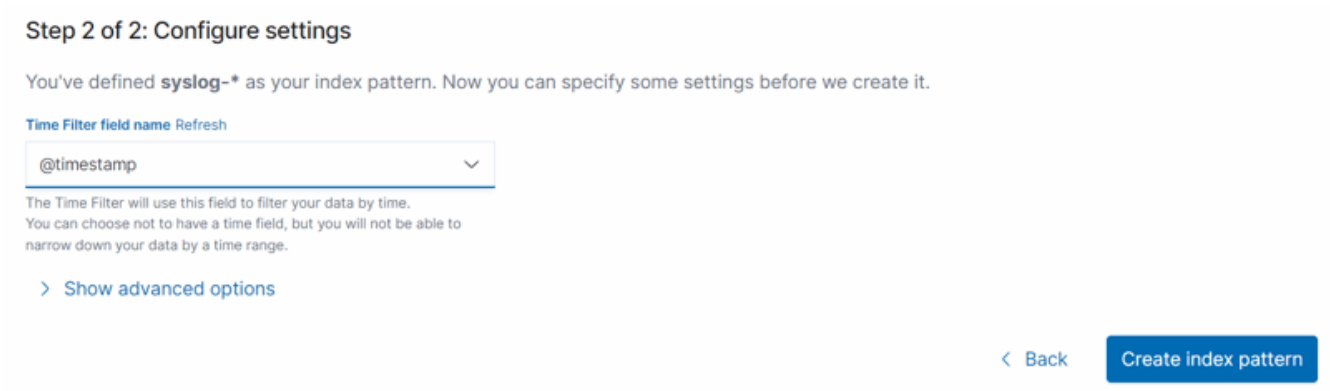


We will click on "Try our sample data" and navigate to

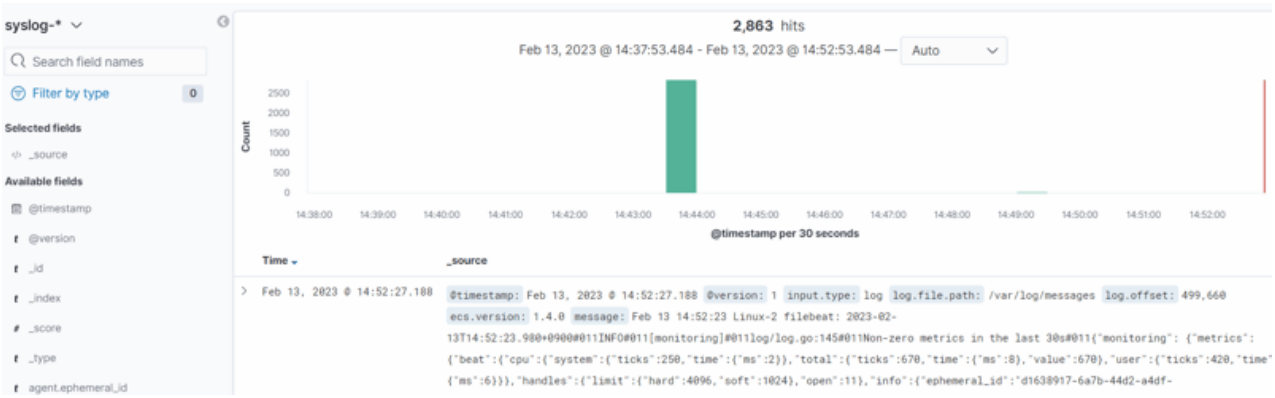


We just need to Define the index pattern

And then configure the settings



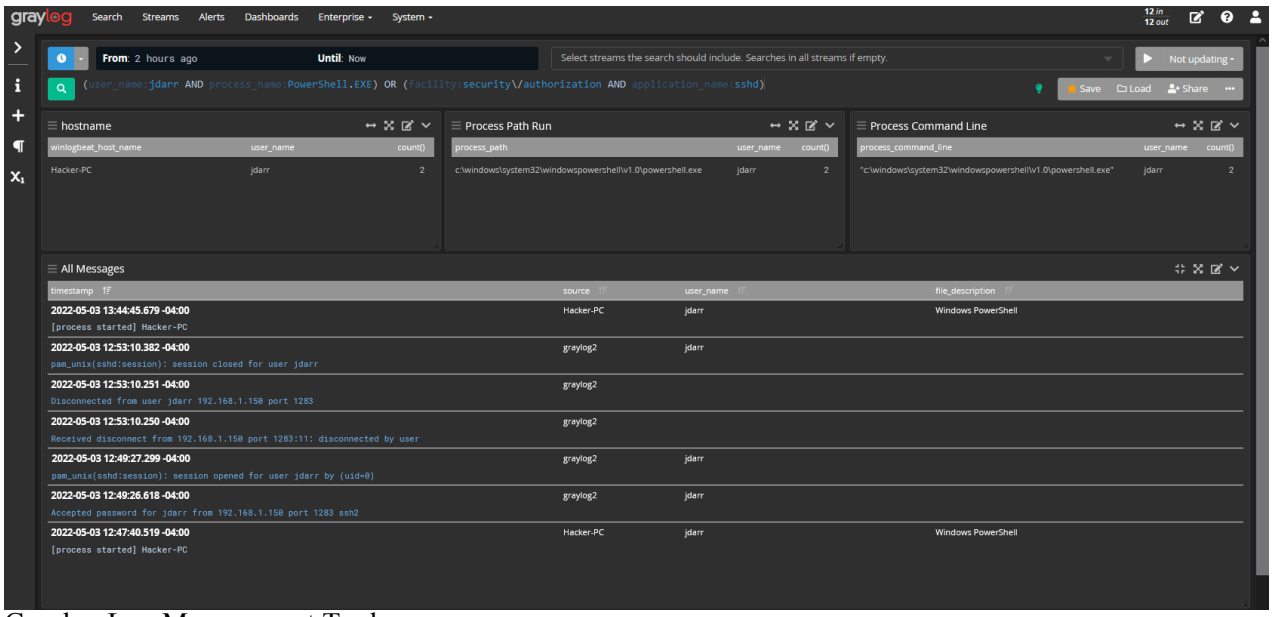
Finally, we just need to navigate to the "Discover" tab to see our logs



Install Graylog Log Management Tool on RHEL System

Graylog is an industry-leading opensource log management solution for collecting, storing, indexing, and analyzing real-time data from applications and a myriad of devices in IT infrastructures such as servers, routers, and firewalls.

Graylog helps you gain more insights into the data collected by combining multiple searches for detailed analysis and reporting. It also detects threats and possible nefarious activity by conducting a deep analysis of the logs from remote sources.



Graylog Log Management Tool

Graylog comprises the following:

- The Graylog Server – This is the main server and is used for processing logs.
- The Graylog web interface – This is a browser application that gives a glance at the data and logs collected from multiple endpoints.
- MongoDB – A NoSQL database server for storing configuration data.
- ElasticSearch – This is a free and open-source search and analytics engine that parses and indexes raw data from various sources.

Graylog's architecture accepts any type of structured data including network traffic and logs from the following:

- Syslog (TCP, UDP, AMQP, Kafka).
- AWS – AWS logs, CloudTrail, & FlowLogs.
- Netflow (UDP).

- GELF (TCP, UDP, AMQP, Kafka).
- ELK – Beats, and Logstash.
- JSON Path from HTTP API.

Some of the giant tech companies that implement Graylog in their tech stacks include Fiverr, CircleCI, CraftBase, and BitPanda.

In this guide, we will show you how to install the Graylog log management tool on RHEL 8 and RHEL-based distros like AlmaLinux, CentOS Stream, and Rocky Linux.

Step 1: Install EPEL Repo and Prerequisite Packages

To start off, you need some essential packages which will be helpful as you move along with this guide. First, install the EPEL repository which provides a rich set of software packages for RHEL & RHEL distributions.

```
$ sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

Next, install the following packages which will be required along the way.

```
$ sudo dnf install -y pwgen wget curl perl-Digest-SHA
```

Step 2: Install Java (OpenJDK) in RHEL

One of the prerequisites of installing Graylog is Java 8 and later versions. Here, we are going to install the latest LTS release of Java which is Java 11 which will be provided by OpenJDK 11.

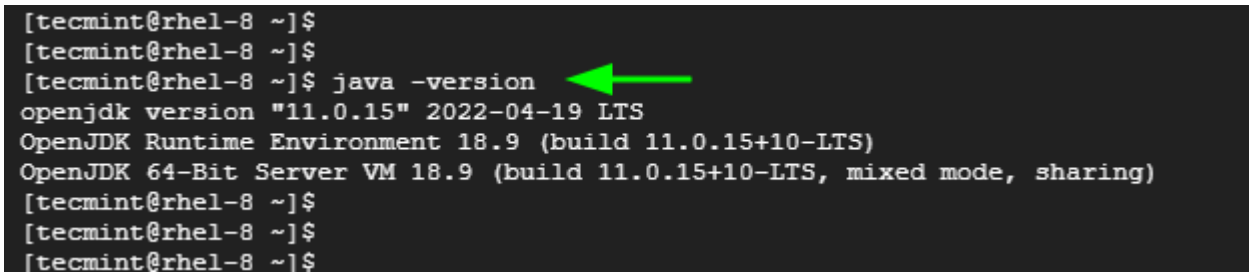
Therefore, run the following command to install OpenJDK.

```
$ sudo dnf install java-11-openjdk java-11-openjdk-devel -y
```

This installs Java dependencies and a host of other dependencies.

Once the installation is complete, verify the version installed.

```
$ java -version
```



Check Java in RHEL

Step 3: Install Elasticsearch in RHEL

Elasticsearch is a free and open-source search and analytics engine that handles a wide variety of data including structured, unstructured, numerical, geospatial, and textual data.

It is a key component of the Elastic stack, also known as ELK (Elasticsearch, Logstash, and Kibana), and is widely used for its simple REST APIs, scalability and speed.

Graylog requires Elasticsearch 6.x or 7.x. We will install Elasticsearch 7.x which is the latest release at the time of publishing this guide.

Create the Elasticsearch repository file.

```
$ sudo vim  /etc/yum.repos.d/elasticsearch.repo
```

Next, paste the following lines of code to the file.

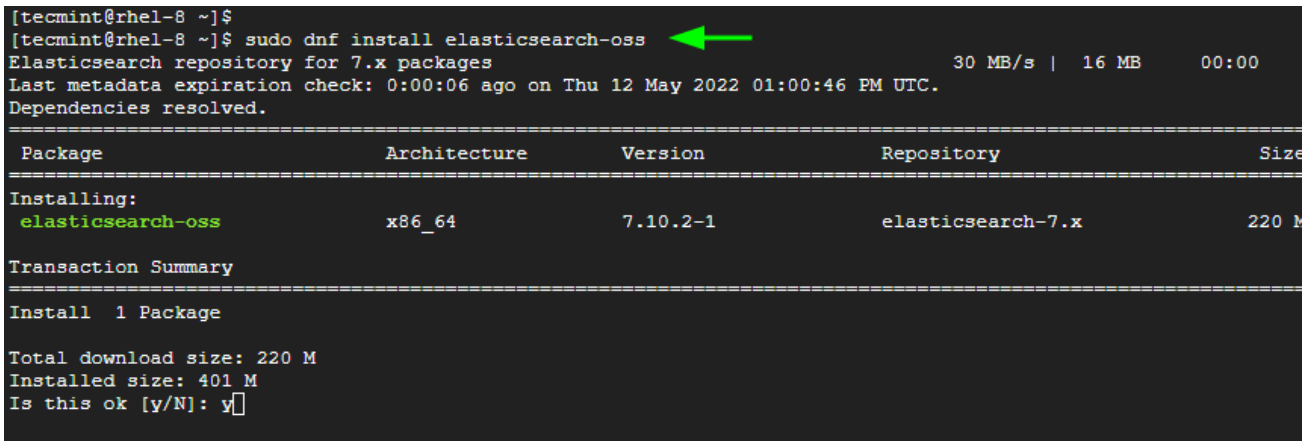
```
[elasticsearch-7.x]

name=Elasticsearch repository for 7.x packages
```

```
gpgcheck=1

gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch

enabled=1
```

Save the changes and exit.

Next, install Elasticsearch using the DNF package manager as shown.

```
$ sudo dnf install elasticsearch-oss
```



Install Elasticsearch in RHEL

For Elasticsearch to work with Graylog, a few changes are required. So open the elasticsearch.yml file.

```
$ sudo vim /etc/elasticsearch/elasticsearch.yml
```

Update the cluster name to Graylog as shown.

```
cluster.name: graylog
```

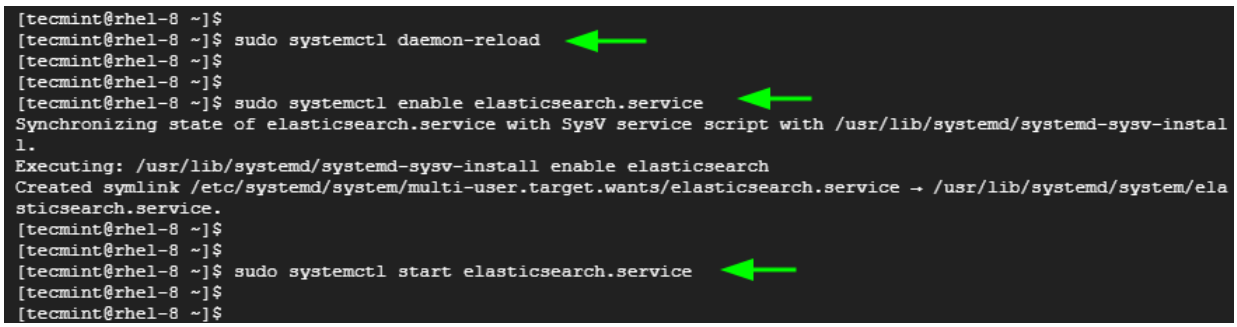
Save the changes and exit.

Then reload the systemd manager configuration.

```
$ sudo systemctl daemon-reload
```

Next, enable and start the Elasticsearch service by running the following commands.

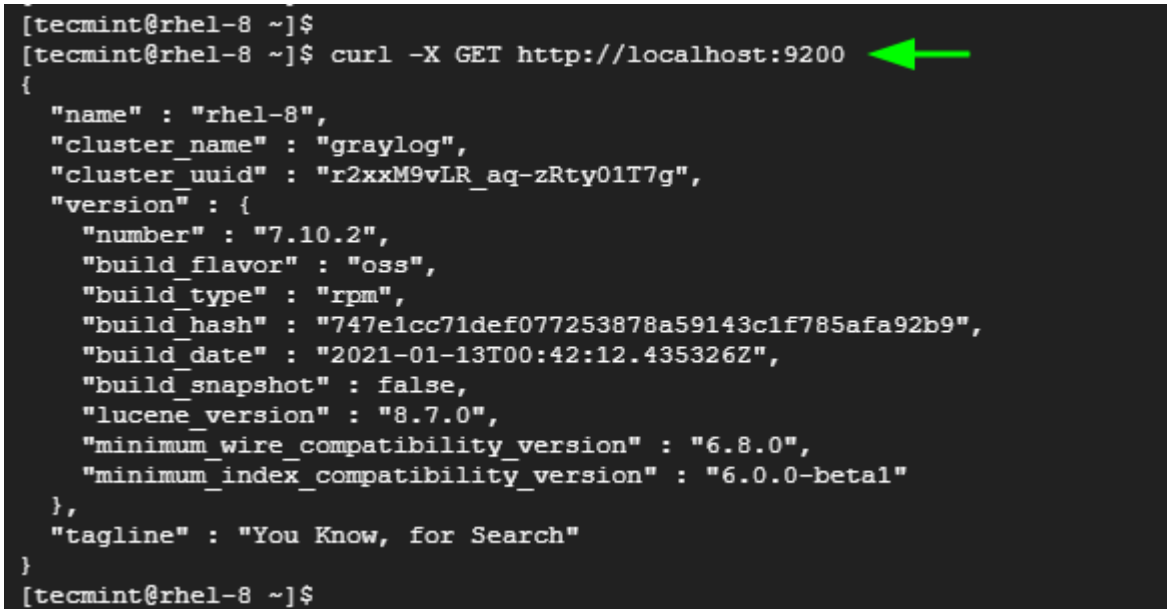
```
$ sudo systemctl enable elasticsearch.service
```



Enable Elasticsearch in RHEL

Elasticsearch listens to port 9200 by default in order to process HTTP requests. You can confirm this by sending a CURL request as shown.

```
$ curl -X GET http://localhost:9200
```



Check Elasticsearch in RHEL

Step 4: Install MongoDB in RHEL

Graylog uses a MongoDB database server to store configuration data.

We will install MongoDB 4.4, but first, create a configuration file for MongoDB.

```
$ sudo vim /etc/yum.repos.d/mongodb-org-4.repo
```

Then paste the following configuration.

```
[mongodb-org-4]

name=MongoDB Repository

baseurl=https://repo.mongodb.org/yum/redhat/8/mongodb-org/4.4/x86_64/
```

Save the changes and exit.

Next, install MongoDB as follows.

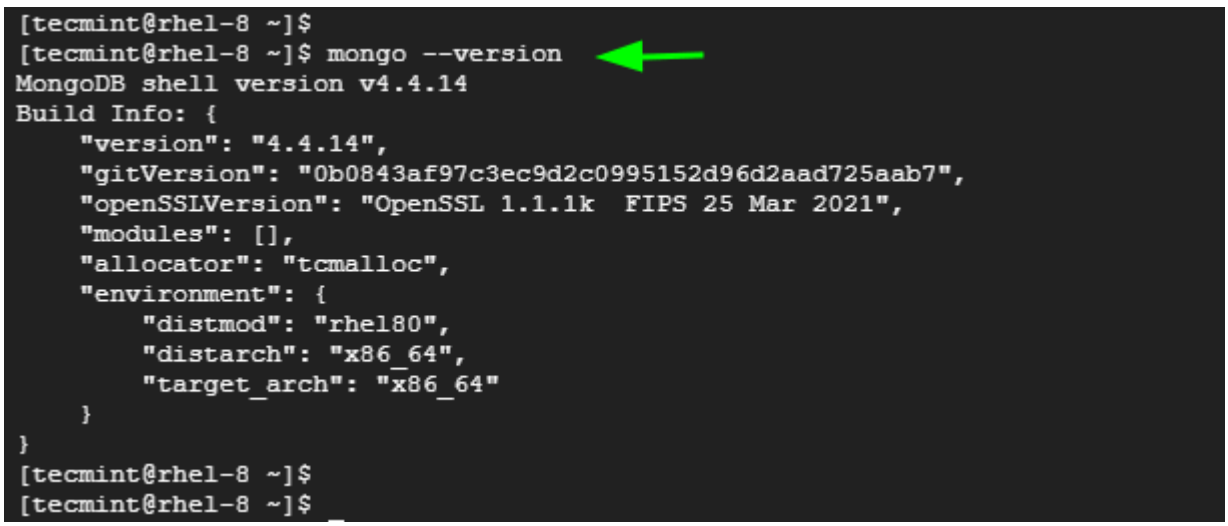
```
$ sudo dnf install mongodb-org
```

Once installed, start and enable MongoDB to start on system startup.

```
$ sudo systemctl start mongod
```

To check the MongoDB version, run the command:

```
$ mongo --version
```



Check MongoDB in RHEL

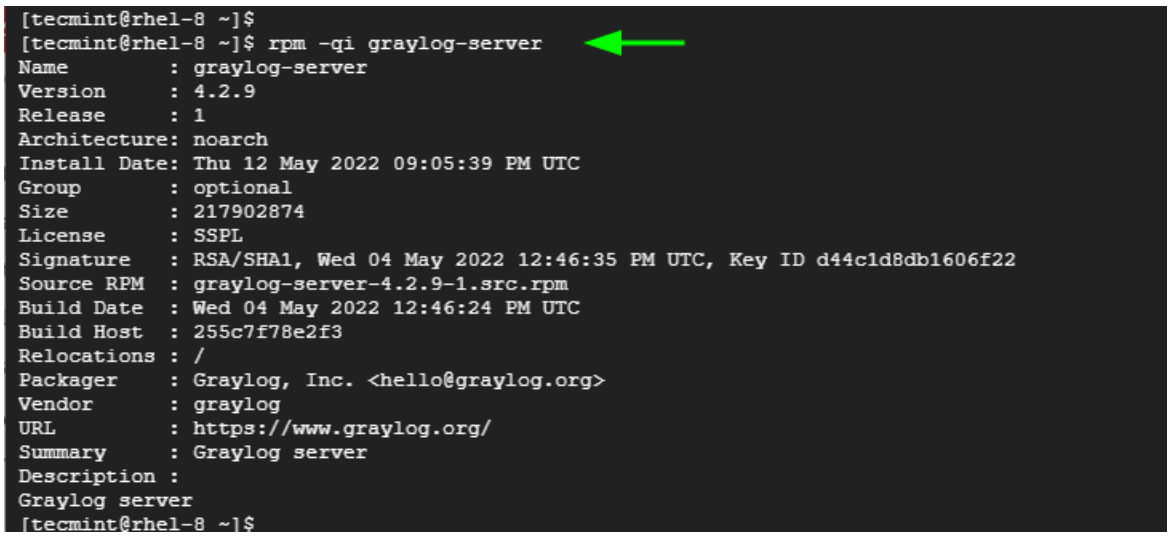
Step 5: Install the Graylog Server in RHEL

With all the prerequisites components installed, now install Graylog by running the following commands.

```
$ sudo rpm -Uvh https://packages.graylog2.org/repo/packages/graylog-4.2-repository_latest.rpm
```

You can verify the installation of Graylog as shown:

```
$ rpm -qi graylog-server
```



Check Graylog in RHEL

Now, start and enable the Graylog server to start on boot time.

```
$ sudo systemctl start graylog-server.service
```

Step 6: Configure the Graylog Server in RHEL

For Graylog to function as expected, some additional steps are required. You need to define the following parameters in the configuration file:

```
root_password_sha2
```

```
password_secret

root_username
```

We will define these variables in the /etc/graylog/server/server.conf file which is the default configuration file.

The root_password_sha2 is the hash password for the root user. To generate it run the following command. The P@ssword321 is just a placeholder. Feel free to specify your own password.

```
$ echo -n P@ssword321 | shasum -a 256
```

Output

```
68e865af8ddbeffc494508bb6181167fccf0bb7c0cab421c54ef3067bdd8d85d
```

Take note of this password and save it somewhere.

Next, generate the password_secret as follows:

```
$ pwgen -N 1 -s 96
```

Output

```
T1EtSsecY0QE4jIG3t6e96A5qLU5WhS9p5SliveX9kybWjC3WKhN4246oqGYPe4B
TLXaaiOcM7LyuSd9bGAonQxkTsTjuqBf
```

Again, take note of this hashed password.

Next, open the Graylog configuration file.

```
$ sudo vim /etc/graylog/server/server.conf
```

Paste the values you generated for root_password_sha2 and password_secret as shown.

```
root_username = admin

root_password_sha2 =
68e865af8ddbeffc494508bb6181167fccf0bb7c0cab421c54ef3067bdd8d85d

password_secret =
```

Additionally, make Graylog accessible by external users by setting the http_bind_address parameter as follows.

```
http_bind_address = 0.0.0.0:9000
```

Also, configure the timezone for the Graylog server.

```
root_timezone = UTC
```

Save and exit the configuration file.

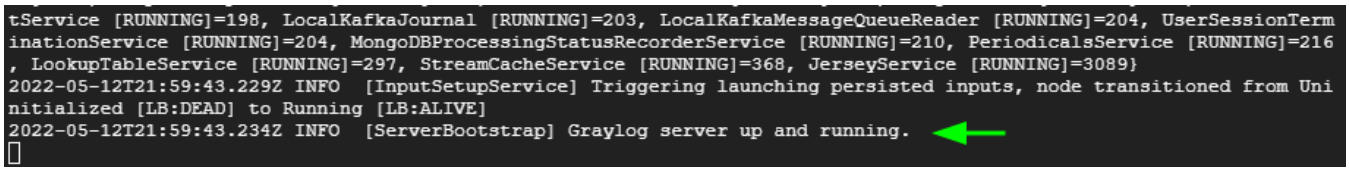
To apply the changes, restart the Graylog server.

```
$ sudo systemctl restart graylog-server.service
```

You can confirm from the log files and check if Graylog is running as expected.

```
$ tail -f /var/log/graylog-server/server.log
```

The following output at the last line shows that all is okay.



Check Graylog Status in RHEL

Graylog listens on port 9000 which provides access to the web interface. So, open this port on the firewall.

```
$ sudo firewall-cmd --add-port=9000/tcp  --permanent
```

Step 7: Access Graylog Web UI

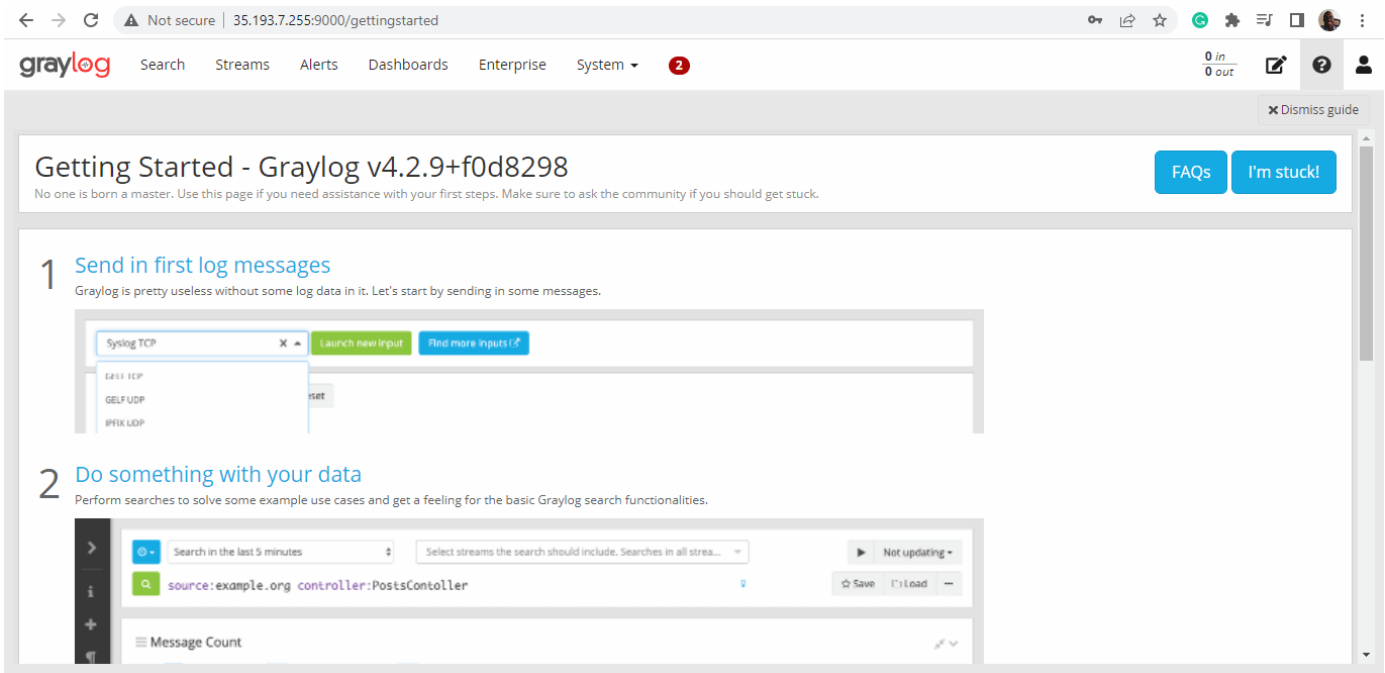
To access Graylog, browse the following URL.

```
http://server-ip:9000

OR
```

Log in with your username admin and the password configured for root_password_sha2 in the server.conf file.

Graylog User Login

Once logged in, you should see the following dashboard.



Graylog Dashboard

From here, you can proceed with analyzing data and logs collected from various data sources. Graylog continues to be a popular centralized log management solution for developers and operation teams. Analysis of the collected data provides deep insights into the working state of various applications and devices and helps find errors and optimize IT operations.

Lab – Convert Data into a Universal Format

- Part 1: Normalize Timestamps in a Log File
- Part 2: Normalize Timestamps in an Apache Log File
- Part 3: Log File Preparation in Security Onion

Background / Scenario

This lab will prepare students to learn where log files are located and how to manipulate and view log files. Log entries are generated by network devices, operating systems, applications, and various types of programmable devices. A file containing a time-sequenced stream of log entries is called a log file.

By nature, log files record events that are relevant to the source. The syntax and format of data within log messages are often defined by the application developer.

Therefore, the terminology used in the log entries often varies from source to source. For example, depending on the source, the terms login, logon, authentication event, and user connection, may all appear in log entries to describe a successful user authentication to a server.

It is often desirable to have a consistent and uniform terminology in logs generated by different sources. This is especially true when all log files are being collected by a centralized point.

The term normalization refers to the process of converting parts of a message, in this case a log entry, to a common format.

In this lab, you will use command line tools to manually normalize log entries. In Part 2, the timestamp field will be normalized. In Part 3, the IPv6 field will be normalized.

Note: While numerous plugins exist to perform log normalization, it is important to understand the basics behind the normalization process.

Required Resources

- CyberOps Workstation VM
- Security Onion VM

Part 1: Normalize Timestamps in a Log File

Timestamps are used in log entries to specify when the recorded event took place. While it is best practice to record timestamps in UTC, the format of the timestamp varies from log source to log source. There are two common timestamp formats, known as Unix Epoch and Human Readable.

Unix Epoch timestamps record time by measuring the number of seconds that have passed since January 1st 1970.

Human Readable timestamps record time by representing separate values for year, month, day, hour, minute, and second.

The Human Readable **Wed, 28 Jun 2017 13:27:18 GMT** timestamp is the same as **1498656439** in Unix Epoch.
From a programmability stand point, it is much easier to work with Epoch as it allows for easier addition and subtraction operations. From an analysis perspective; however, Human Readable timestamps are much easier to interpret.

Converting Epoch to Human Readable Timestamps with AWK
AWK is a programming language designed to manipulate text files. It is very powerful and especially useful when handling text files where the lines contain multiple fields, separated by a delimiter character. Log files contain one entry per line and are formatted as delimiter-separated fields, making AWK a great tool for normalizing.

Consider the **applicationX_in_epoch.log** file below. The source of the log file is not relevant.

```
2|Z|1219071600|AF|0

3|N|1219158000|AF|89

4|N|1220799600|AS|12

1|Z|1220886000|AS|67
```

The log file above was generated by application X. The relevant aspects of the file are:

- The columns are separated, or delimited, by the | character. Therefore, the file has five columns.
- The third column contains timestamps in Unix Epoch.
- The file has an extra line at the end. This will be important later in the lab.

Assume that a log analyst needed to convert the timestamps to the Human Readable format. Follow the steps below to use AWK to easily perform the manual conversion:

a. Launch the **CyberOps Workstation VM** and then launch a terminal window.
b. Use the **cd** command to change to the **/home/analyst/lab.support.files/** directory. A copy of the file shown above is stored there.

```
[analyst@secOps ~]$ cd ./lab.support.files/

[analyst@secOps lab.support.files]$ ls -l

total 580

-rw-r--r-- 1 analyst analyst    649 Jun 28 18:34 apache_in_epoch.log

-rw-r--r-- 1 analyst analyst    126 Jun 28 11:13 applicationX_in_epoch.log
```

```
-rw-r--r-- 1 analyst analyst    102 Jul 20 09:37 confidential.txt
<output omitted>
[analyst@secOps lab.support.files]$
```

c. Issue the following AWK command to convert and print the result on the terminal:

Note: It is easy to make a typing error in the following script. Consider copying the script out to a text editor to remove the extra line breaks. Then copy the script from the text editor into the **CyberOps Workstation VM** terminal window. However, be sure to study the script explanation below to learn how this script modifies the timestamp field.

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS="|"}

{$3=strftime("%c",$3)} {print}' applicationX_in_epoch.log

2|Z|Mon 18 Aug 2008 11:00:00 AM EDT|AF|0

3|N|Tue 19 Aug 2008 11:00:00 AM EDT|AF|89

4|N|Sun 07 Sep 2008 11:00:00 AM EDT|AS|12

1|Z|Mon 08 Sep 2008 11:00:00 AM EDT|AS|67
```

The command above is an AWK script. It may seem complicated. The main structure of the AWK script above is as follows:

- **awk** – This invokes the AWK interpreter.
- **'BEGIN** – This defines the beginning of the script.
- **{}** – This defines actions to be taken in each line of the input text file. An AWK script can have several actions.
- **FS = OFS = "|"** – This defines the field separator (i.e., delimiter) as the bar (|) symbol. Different text files may use different delimiting characters to separate fields. This operator allows the user to define what character is used as the field separator in the current text file.
- **$3** – This refers to the value in the third column of the current line. In the **applicationX_in_epoch.log,** the third column contains the timestamp in epoch to be converted.
- **strftime** – This is an AWK internal function designed to work with time. The %c and $3 in between parenthesis are the parameters passed to **strftime.**
- **applicationX_in_epoch.log** – This is the input text file to be loaded and used. Because you are already in the **lab.support.files** directory, you do not need to

add path information, **/home/analyst/lab.support.files/applicationX_in_epoch.log.**

The first script action, defined in the first set of curly brackets is to define the field separator character as the "|". Then, in the second set of curly brackets, it rewrites the third column of each line with the result of the execution of the **strftime()** function. **strftime()** is an internal AWK function created to handle time conversion. Notice that the script tells the function to use the contents of the third column of each line before the change **($3)** and to format the output **(%c).**

Were the Unix Epoch timestamps converted to Human Readable format? Were the other fields modified? Explain.

Yes, the script converted from Epoch to Human Readable. The script changed only the timestamp field, preserving the rest of the file.

Compare the contents of the file and the printed output. Why is there the line, **||Wed 31 Dec 1969 07:00:00 PM EST**?

The reason for the extra line is because the file has an empty line at the end, which led the script to mistakenly interpret it as 0 and convert that into a Human Readable timestamp.

By interpreting the empty line as 0, the script converted 0 Unix Epoch to Human Readable. 0 Unix Epoch translates to 0 seconds after midnight of Jan 1st, 1970. The script displays "Wed 31 Dec 1969 07:00:00 PM EST" because it automatically adjusts for the timezone. Because the CyberOps Workstation is configured for EST (UTC -5), the script displays the midnight, Jan 1st 1970 minus 5 hours.

d. Use **nano** (or your favorite text editor) to remove the extra empty line at the end of the file and run the **AWK** script again.

```
[analyst@secOps lab.support.files]$ nano applicationX_in_epoch.log
```

Is the output correct now? Explain.

Yes. Because the empty line was removed, no extra data was created and added to the log file by the script.

e. While printing the result on the screen is useful for troubleshooting the script, analysts will likely need to save the output in a text file. Redirect the output of the script above to a file named **applicationX_in_human.log** to save it to a file:

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS="|"}

{$3=strftime("%c",$3)} {print}' applicationX_in_epoch.log >

applicationX_in_human.log
```

What was printed by the command above? Is this expected?

Nothing was printed on the screen. Yes, it is expected, as the command output was redirected to a text file named **applicationX_in_human.log.**

f. Use **cat** to view the **applicationX_in_human.log.** Notice that the extra line is now removed and the timestamps for the log entries have been converted to human readable format.

```
[analyst@secOps lab.support.files]$ cat applicationX_in_human.log

2|Z|Mon 18 Aug 2008 11:00:00 AM EDT|AF|0

3|N|Tue 19 Aug 2008 11:00:00 AM EDT|AF|89

4|N|Sun 07 Sep 2008 11:00:00 AM EDT|AS|12

1|Z|Mon 08 Sep 2008 11:00:00 AM EDT|AS|67
```

Part 2: Normalize Timestamps in an Apache Log File

Similar to what was done with the **applicationX_in_epoch.log** file, Apache log files can also be normalized. Follow the steps below to convert Unix Epoch to Human Readable timestamps. Consider the following Apache log file, **apache_in_epoch.log:**

```
[analyst@secOps lab.support.files]$ cat apache_in_epoch.log

198.51.100.213 - - [1219071600] "GET

/twiki/bin/edit/Main/Double_bounce_sender?topicparent=Main.ConfigurationVariable
s

HTTP/1.1" 401 12846

198.51.100.213 - - [1219158000] "GET

/twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1.3&rev2=1.2 HTTP/1.1" 200 4523

198.51.100.213 - - [1220799600] "GET /mailman/listinfo/hsdivision HTTP/1.1" 200
6291

198.51.100.213 - - [1220886000] "GET /twiki/bin/view/TWiki/WikiSyntax
HTTP/1.1" 200
```

```
/twiki/bin/oops/TWiki/AppendixFileSystem?template=oopsmore&m1=1.12&m2=1.1
2 HTTP/1.1"
```

The Apache Log file above contains six entries which record events related to the Apache web server.
Each entry has seven fields. The fields are delimited by a space:

- The first column contains the IPv4 address, **198.51.100.213,** of the web client placing the request.
- The second and third columns are not used and a "-" character is used to represent no value.
- The fourth column contains the timestamp in Unix Epoch time, for example **[1219071600].**
- The fifth column contains text with details about the event, including URLs and web request parameters. All six entries are HTTP GET messages. Because these messages include spaces, the entire field is enclosed with quotes.
- The sixth column contains the HTTP status code, for example **401.**
- The seventh column contains the size of the response to the client (in bytes), for example **12846.**

Similar to part one, a script will be created to convert the timestamp from Epoch to Human Readable.

a. First, answer the questions below. They are crucial for the construction of the script.

In the context of timestamp conversion, what character would work as a good delimiter character for the Apache log file above?
The space character.
How many columns does the Apache log file above contain?
7
In the Apache log file above, what column contains the Unix Epoch Timestamp?
Column 4
b. In the **CyberOps Workstation VM** terminal, a copy of the Apache log file, apache_in_epoch.log, is stored in the /home/analyst/lab.support.files.
c. Use an **awk** script to convert the timestamp field to a human readable format. Notice that the command contains the same script used previously, but with a few adjustments for the timestamp field and file name.

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS=" "}{$4=strftime("%c

",$4)} {print}'
```

Was the script able to properly convert the timestamps? Describe the output.
No. All timestamps are now Wed 31 Dec 1969 07:00:00 PM EST.

d. Before moving forward, think about the output of the script. Can you guess what caused the incorrect output? Is the script incorrect? What are the relevant differences between the **applicationX_in_epoch.log** and **apache_in_epoch.log?**
The problem is the square brackets in the course file. The script expects the timestamp to be in the Unix Epoch format which does not include the square brackets. Because the script does not know what number represents the "[" character, it assumes zero and returns the Unix beginning of time in UTC -5.
e. To fix the problem, the square brackets must be removed from the timestamp field before the conversion takes place. Adjust the script by adding two actions before the conversion, as shown below:

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS=" "}

{gsub(/\[|\]/,"",$4)} {print} {$4=strftime("%c",$4)} {print}'
```

Notice after specifying space as the delimiter with **{FS=OFS=" "}**, there is a regular expression action to match and replace the square brackets with an empty string, effectively removing the square brackets that appear in the timestamp field. The second action prints the updated line so the conversion action can be performed.

- **gsub()** – This is an internal AWK function used to locate and substitute strings. In the script above, **gsub()** received three comma-separated parameters, described below.
- /\[|\]/ – This is a regular expression passed to **gsub()** as the first parameter. The regular expression should be read as '**find "[" OR "]"**'. Below is the breakdown of the expression:
  - The first and last "/" character marks the beginning and end of the search block. Anything between the first "/" and the second "/" are related to the search. The "\" character is used to escape the following "[". Escaping is necessary because "[" can also be used by an operator in regular expressions. By escaping the "[" with a leading "\", we tell the interpreter that the "]" is part of the content and not an operator. The "|" character is the OR operator. Notice that the "|" is not escaped and will therefore, be seen as an operator. Lastly, the regular expression escapes the closing square bracket with "\]", as done before.
- **""** – This represents no characters, or an empty string. This parameter tells **gsub()** what to replace the "[" and "]" with, when found. By replacing the "[" and "]" with "", **gsub()** effectively removes the "[" and "]" characters.
- **$4** – This tells **gsub()** to work only on the fourth column of the current line, the timestamp column.

Note: Regular expression interpretation is a SECOPS exam topic. Regular expressions are covered in more detail in another lab in this chapter. However, you may wish to search the Internet for tutorials.
f. In a CyberOps Workstation VM terminal, execute the adjusted script, as follows:

```
[analyst@secOps lab.support.files]$ awk 'BEGIN {FS=OFS="
  {gsub(/\[|\]/,"",$4)} {print}
```

apache_in_epoch.log

Was the script able to properly convert the timestamps this time? Describe the output.
Yes. The output now displays two lines for each log entry. The first line displays the timestamp in Unix Epoch format and the second line is the same log entry with the timestamp displayed using Human Readable format.

Part 3: Log File Preparation in Security Onion

Because log file normalization is important, log analysis tools often include log normalization features. Tools that do not include such features often rely on plugins for log normalization and preparation. The goal of these plugins is to allow log analysis tools to normalize and prepare the received log files for tool consumption.

The Security Onion appliance relies on a number of tools to provide log analysis services. **ELSA, Bro, Snort** and **SGUIL** are arguably the most used tools.
ELSA (Enterprise Log Search and Archive) is a solution to achieve the following:

- Normalize, store, and index logs at unlimited volumes and rates.
- .Provide a simple and clean search interface and API.
- Provide an infrastructure for alerting, reporting and sharing logs.
- Control user actions with local or LDAP/AD-based permissions.
- Plugin system for taking actions with logs.
- ·Exist as a completely free and open-source project.

Bro is a framework designed to analyze network traffic and generate event logs based on it. Upon network traffic analysis, Bro creates logs describing events such as the following:

- TCP/UDP/ICMP network connections
- DNS activity
- FTP activity
- HTTPS requests and replies
- SSL/TLS handshakes

Snort and SGUIL
Snort is an IDS that relies on pre-defined rules to flag potentially harmful traffic. Snort looks into all portions of network packets (headers and payload), looking for patterns defined in its rules. When found, Snort takes the action defined in the same rule.

SGUIL provides a graphical interface for Snort logs and alerts, allowing a security analyst to pivot from SGUIL into other tools for more information. For example, if a potentially malicious packet is sent to the organization web server and Snort raised an alert about it, SGUIL will list that alert. The analyst can then right-click that alert to search the ELSA or Bro databases for a better understanding of the event.

Note: The directory listing maybe different than the sample output shown below.

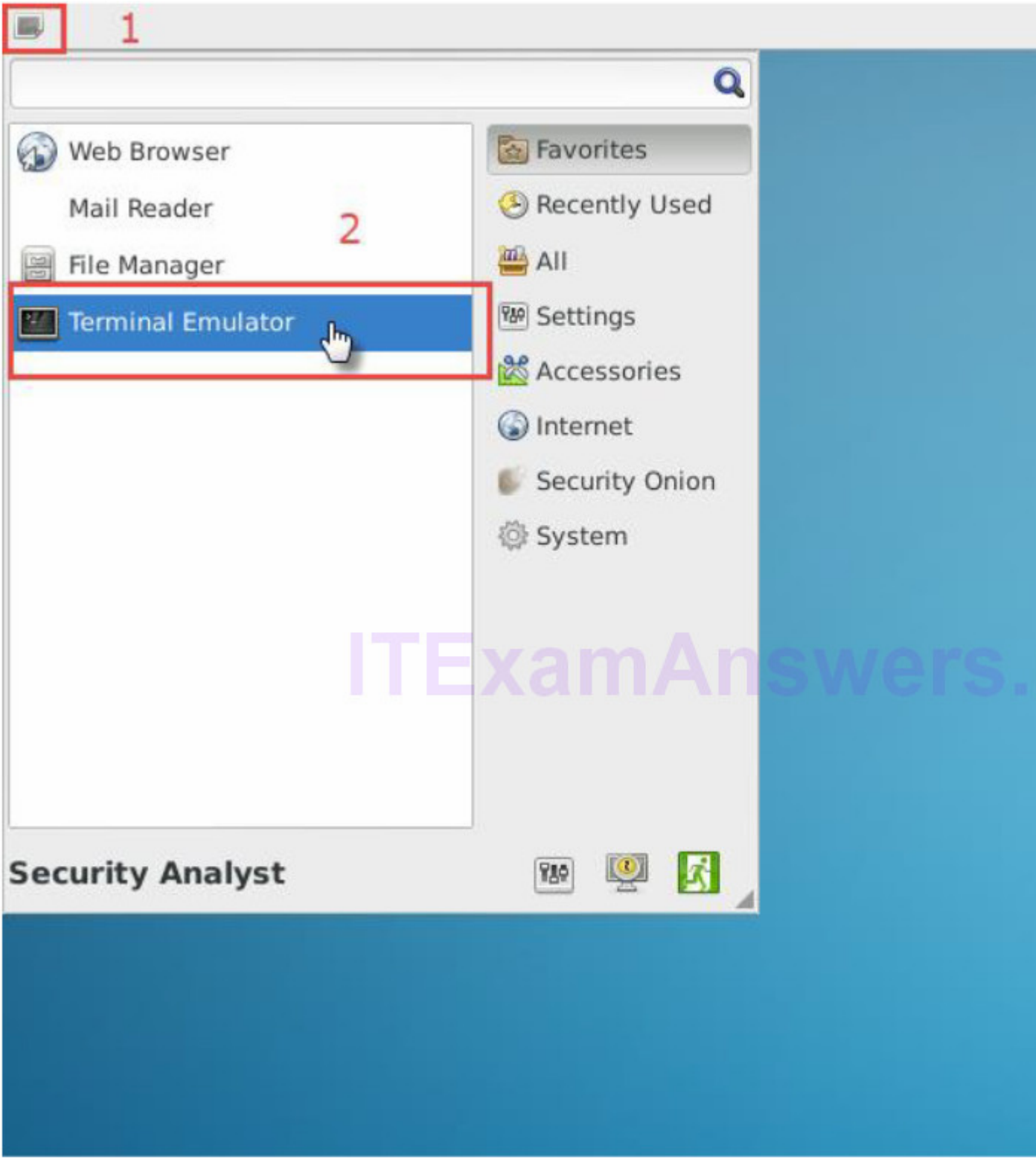
Step 1: Switch to Security Onion.

Launch the **Security Onion VM** from VirtualBox's Dashboard (username: **analyst** / password: **cyberops).**

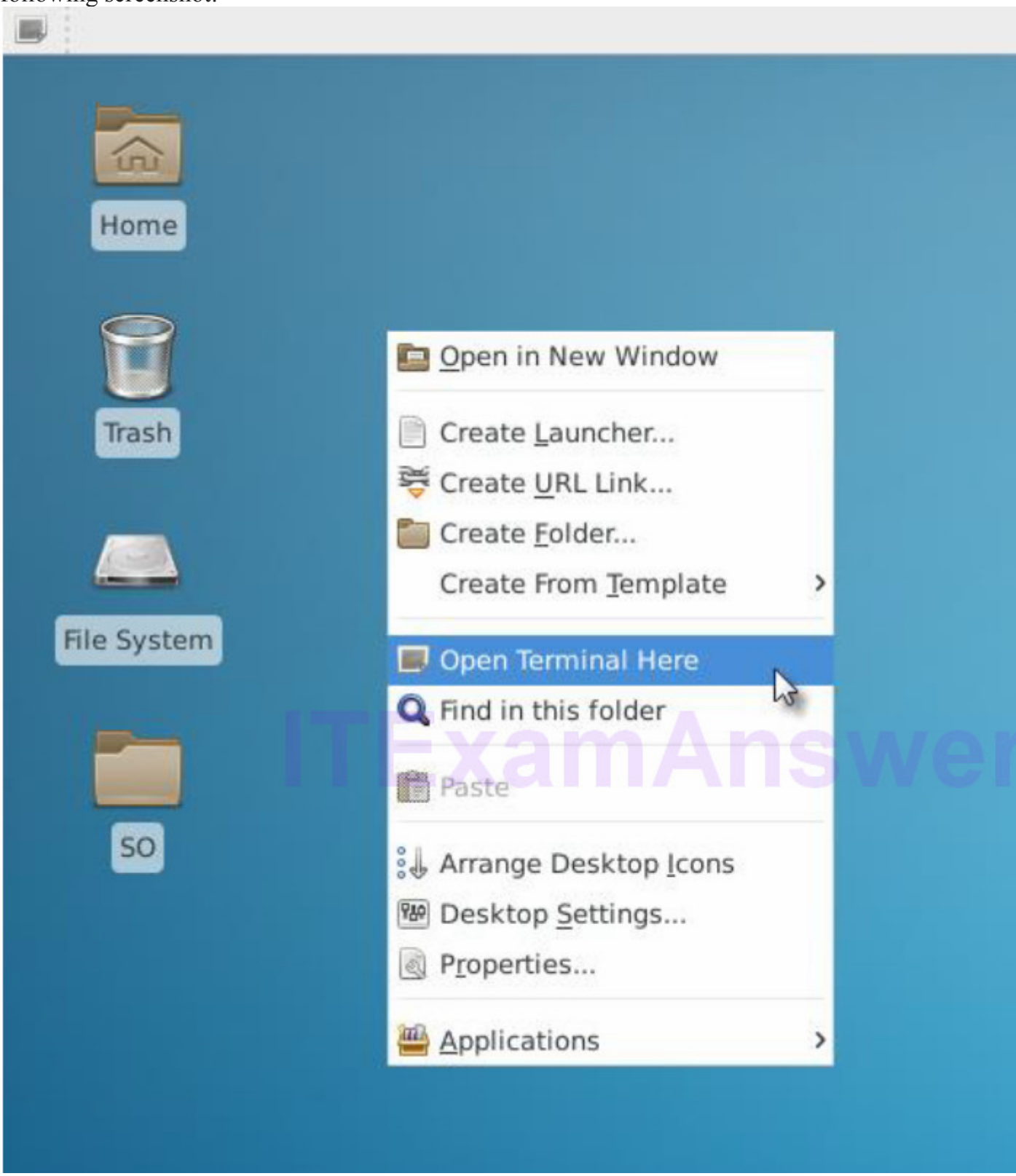
The **CyberOps Workstation VM** can be closed to free up memory in the host computer for this part of the lab

Step 2: ELSA Logs

a. Open a terminal window in the Security Onion VM. You can access to the applications menu is shown in the following screenshot:



b. You can also right-click the **Desktop > Open Terminal Here**, as show in the following screenshot:



c. ELSA logs can be found under the **/nsm/elsa/data/elsa/log/** directory. Change the directory using the following command:

```
analyst@SecOnion:~/Desktop$ cd /nsm/elsa/data/elsa/log
```

```
analyst@SecOnion:/nsm/elsa/data/elsa/log$
```

d. Use the **ls –l** command to list the files:

```
analyst@SecOnion:/nsm/elsa/data/elsa/log$ ls -l

total 99112

total 169528

-rw-rw---- 1 www-data sphinxsearch 56629174 Aug 18 14:15 node.log

-rw-rw---- 1 www-data sphinxsearch 6547557 Aug 3 07:34 node.log.1.gz

-rw-rw---- 1 www-data sphinxsearch 7014600 Jul 17 07:34 node.log.2.gz

-rw-rw---- 1 www-data sphinxsearch 6102122 Jul 13 07:34 node.log.3.gz
```

Step 3: Bro Logs in Security Onion

a. Bro logs are stored at **/nsm/bro/logs/.** As usual with Linux systems, log files are rotated based on the date, renamed and stored on the disk. The current log files can be found under the **current** directory. From the terminal window, change directory using the following command.

```
analyst@SecOnion:/nsm/elsa/data/elsa/log$ cd /nsm/bro/logs/current
```

b. Use the **ls -l** command to see all the log files generated by Bro:

```
analyst@SecOnion:/nsm/bro/logs/current$ ls -l

total 100

-rw-rw-r-- 1 sguil sguil 368 Aug 18 14:02 capture_loss.log
```

```
-rw-rw-r-- 1 sguil sguil 2028 Aug 18 14:16 stats.log
-rw-rw-r-- 1 sguil sguil 40 Aug 18 14:00 stderr.log
-rw-rw-r-- 1 sguil sguil 188 Aug 18 13:46 stdout.log
```

Step 4: Snort Logs

a. Snort logs can be found at **/nsm/sensor_data/.** Change directory as follows.

```
analyst@SecOnion:/nsm/bro/logs/current$ cd /nsm/sensor_data
```

b. Use the ls -l command to see all the log files generated by Snort.

```
analyst@SecOnion:/nsm/sensor_data$ ls -l
total 16
drwxrwxr-x 7 sguil sguil 4096 Jun 19 23:16 seconion-eth0
drwxrwxr-x 7 sguil sguil 4096 Jun 19 23:16 seconion-eth1
```

c. Notice that Security Onion separates files based on the interface. Because the **Security Onion VM** image has four interfaces, four directories are kept. Use the **ls –l seconion-eth0** command to see the files generated by the ethernet 0 interface.

```
analyst@SecOnion:/nsm/sensor_data$ ls -l seconion-eth0/
total 52
drwxrwxr-x 2 sguil sguil 4096 Jun 19 23:09 argus
drwxrwxr-x 10 sguil sguil 4096 Jul 7 12:09 dailylogs
drwxrwxr-x 2 sguil sguil 4096 Jun 19 23:08 portscans
```

```
-rw-r--r-- 1 root root 0 Jun 19 23:08 snort.stats
```

Step 5: Various Logs

a. While the **/nsm/** directory stores some logs files, more specific log files can be found under **/var/log/nsm/.** Change directory and use the **ls -l** command to see all the log files in the directory.

```
analyst@SecOnion:/nsm/sensor_data$ cd /var/log/nsm/

analyst@SecOnion:/var/log/nsm$ ls -l

total 8364

-rw-r--r-- 1 sguil sguil        4 Aug 18 14:56 eth0-packets.log

-rw-r--r-- 1 sguil sguil        4 Aug 18 14:56 eth1-packets.log

-rw-r--r-- 1 sguil sguil        4 Aug 18 14:56 eth2-packets.log

-rw-r--r-- 1 sguil sguil      182 Aug 18 13:46 ossec_agent.log

-rw-r--r-- 1 sguil sguil      202 Jul 11 12:02 ossec_agent.log.20170711120202

-rw-r--r-- 1 sguil sguil      202 Jul 13 12:02 ossec_agent.log.20170713120201

-rw-r--r-- 1 sguil sguil      202 Jul 14 12:02 ossec_agent.log.20170714120201

-rw-r--r-- 1 sguil sguil      202 Jul 15 12:02 ossec_agent.log.20170715120202

-rw-r--r-- 1 sguil sguil      249 Jul 16 12:02 ossec_agent.log.20170716120201
```

```
drwxr-xr-x 2 sguil sguil          4096 Jun 19 23:08 securityonion
-rw-r--r-- 1 sguil sguil          1647 Jun 19 23:09 securityonion-elsa-config.log
-rw-r--r-- 1 sguil sguil       7708106 Aug 18 14:56 sensor-clean.log
-rw-r--r-- 1 sguil sguil          1603 Aug  4 00:00 sensor-newday-argus.log
-rw-r--r-- 1 sguil sguil          1603 Aug  4 00:00 sensor-newday-http-agent.log
-rw-r--r-- 1 sguil sguil          8875 Aug  4 00:00 sensor-newday-pcap.log
-rw-r--r-- 1 sguil sguil         53163 Aug  4 05:01 sguil-db-purge.log
-rw-r--r-- 1 sguil sguil        369738 Aug  4 07:33 sid_changes.log
-rw-r--r-- 1 sguil sguil         22598 Aug  8 01:35 so-bro-cron.log
drwxrwxr-x 2 sguil securityonion  4096 Jun 19 23:09 so-elsa
```

Notice that the directory shown above also contains logs used by secondary tools such as **OSSEC, Pulledpork, Sphinx,** and **Squert.**
b. Take some time to Google these secondary tools and answer the questions below:

For each one of the tools listed above, describe the function, importance, and placement in the security analyst workflow.

Sphinx is an open source search engine and is used by ELSA to provide search capabilities.
Pulledpork is a Snort rule manage system. It facilitates Snort rules updating. Outdated Snort rules makes the entire system useless.
OSSEC is a system used to normalize and concentrate local system logs. When deployed throughout the organization, OSSEC allows an analyst to have a clear picture of what is happening in the systems.

Squert is a visual tool that attempts to provide additional context to events through the use of metadata, time series representations, and weighted and logically grouped result sets.

Part 4: Reflection

Log normalization is important and depends on the deployed environment.

Popular tools include their own normalization features, but log normalization can also be done manually.

When manually normalizing and preparing log files, double-check scripts to ensure the desired result is achieved. A poorly written normalization script may modify the data, directly impacting the analyst's work.