**RIT Dubai**

**CSEC 462-Network Security and Forensics**

**Spring 2025**

**LAB3 Snort and Firewall Rules**

|  | **Name** | **ID** |
| --- | --- | --- |
| **1** | Rania Kanaan | 769007321 |
| **2** | Fatima Mansur Rabiu | 776006447 |
| **3** | Suhaila Alfalasi | 764005288 |
| **4** | Syed Shayan Ali | 393003651 |

**Objectives:**

In this lab, you will experience with tools that generate alerts. Snort is an NSM tool that can detect security violation based on rules. Firewalls are also used to monitor connections and prevent malicious ones. Various types of alerting messages will be investigated in this lab.

**Deliverables and Rubrics:**

The rubrics of this lab are as follows:

1. Report is organized properly with suitable objectives and structure. The organization of the report should be as follows: (15%).
   1. Cover page which contains Lab general information, Group information, and Rubrics.
   2. Objectives and goals of this lab
   3. After answering the question of “Lab26.1.7: / Snort and Firewall Rules” lab and providing your own screenshots, copy and paste the whole document in here.
   4. Conclusion that provides a summary of this lab above and discusses how the experience with these tools can help a security analyst in detecting attacks and malware.
2. The report provides all the figures and screenshots that demonstrate the steps of using these tools and data that you have captured associated with discussion and clear answers to questions that are part of these Lab26.1.7 (70%)
3. The summary or conclusion provides a summary of the lab above and discusses how the experience with these tools can help a security analyst in detecting attacks and malware. (15%)

Note: Download *CyberOps Workstation* from the link below.

<https://drive.google.com/drive/folders/1xFiRgCPw_RweYlLLHqsYfZBQrAsie-a-?usp=sharing>

Username: analyst

Password: cyberops

# TABLE OF CONTENTS

[**TABLE OF CONTENTS 3**](#_gjdgxs)

[**TABLE OF FIGURES & TABLES 4**](#_1fob9te)

[**OBJECTIVES 5**](#_1j5k30ehz4j9)

[**TOPOLOGY & ARCHITECTURE 6**](#_3dy6vkm)

[**Part 1: Preparing the Virtual Environment 7**](#_xewohqqmahuo)

[**Part 2: Firewall and IDS Logs 8**](#_krs2ebnm329j)

[Step 1: Real-Time IDS Log Monitoring 9](#_4k5tiebvn1gh)

[Step 2: Tuning Firewall Rules Based on IDS Alerts 20](#_dsfz6h5cvaiz)

[**Part 3: Terminate and Clear Mininet Process 26**](#_hll5wq1w5gg)

[**CONCLUSION 28**](#)

[**References 29**](#_qsh70q)

# TABLE OF FIGURES & TABLES

| **Figure #** | **Title** | **Page #** |
| --- | --- | --- |
| Figure 1 | CyberOps Topology | 7 |
| Figure 2 | Requesting IP information via DHCP | 8 |
| Figure 3 | Testing connectivity | 9 |
| Figure 4 | mininet prompt | 11 |
| Figure 5 | Opening the shell on R1 | 12 |
| Figure 6 | Starting Snort | 13 |
| Figure 7 | Prompt as Snort is now running | 13 |
| Figure 8 | Opening shells for hosts H5 and H10 | 14 |
| Figure 9 | Starting the server on H10 shell | 15 |
| Figure 10 | Confirmation of webserver nginx | 16 |
| Figure 11 | No alerts shown in the log | 17 |
| Figure 12 | Download of file W32.Nimda.Amm.exe | 18 |
| Figure 13 | Alert log of Malicious Server | 19 |
| Figure 14 | Capturing tcpdump packets | 21 |
| Figure 15 | Downloading file W32.Nimda.Amm.exe after capturing tcpdump | 22 |
| Figure 16 | Stopping the capture by bringing tcpdump to foreground | 23 |
| Figure 17 | Verifying the pcap file | 24 |
| Figure 18 | iptables | 26 |
| Figure 19 | Forwarding the chain on R1 | 27 |
| Figure 20 | Confirmation that rule was added to the FORWARD chain | 28 |
| Figure 21 | Downloading the file after Forwarding | 29 |
| Figure 22 | Terminating the Mininet | 31 |
| Figure 23 | Cleaning up the processes started by Mininet | 32 |

# 

# 

# 

# 

# 

# 

# OBJECTIVES

The goal of this lab is to deliver a real-world training for intrusion detection and firewall security by implementing Snort and iptables features. Students will establish virtual security frameworks then monitor network events while identifying harmful activities followed by deployment of firewall rules to protect against threats. Students receive training about IDS and firewall rule structures through this lab while experiencing their joint security functions for network protection.

* **Setting up the environment**: The first step requires you to set up the CyberOps Workstation VM and verify network accessibility.
* **Monitor IDS Alerts**: Check IDS Alerts through Snort to find suspicious network traffic along with security alerts.
* **Simulate Malicious Activity**: Run a malware-hosting server and attempt to download a malicious file.
* **Capture and Analyze Network Traffic**: Network events can be recorded through tcpdump for use in security investigation.
* **Configure Firewall Rules**: The network security relies on iptables to establish rules that both reject dangerous connections and make systems more secure.
* **Validate Security Measures**: Test and confirm to prove that firewall security rules deliver protection from additional threats.

# TOPOLOGY & ARCHITECTURE

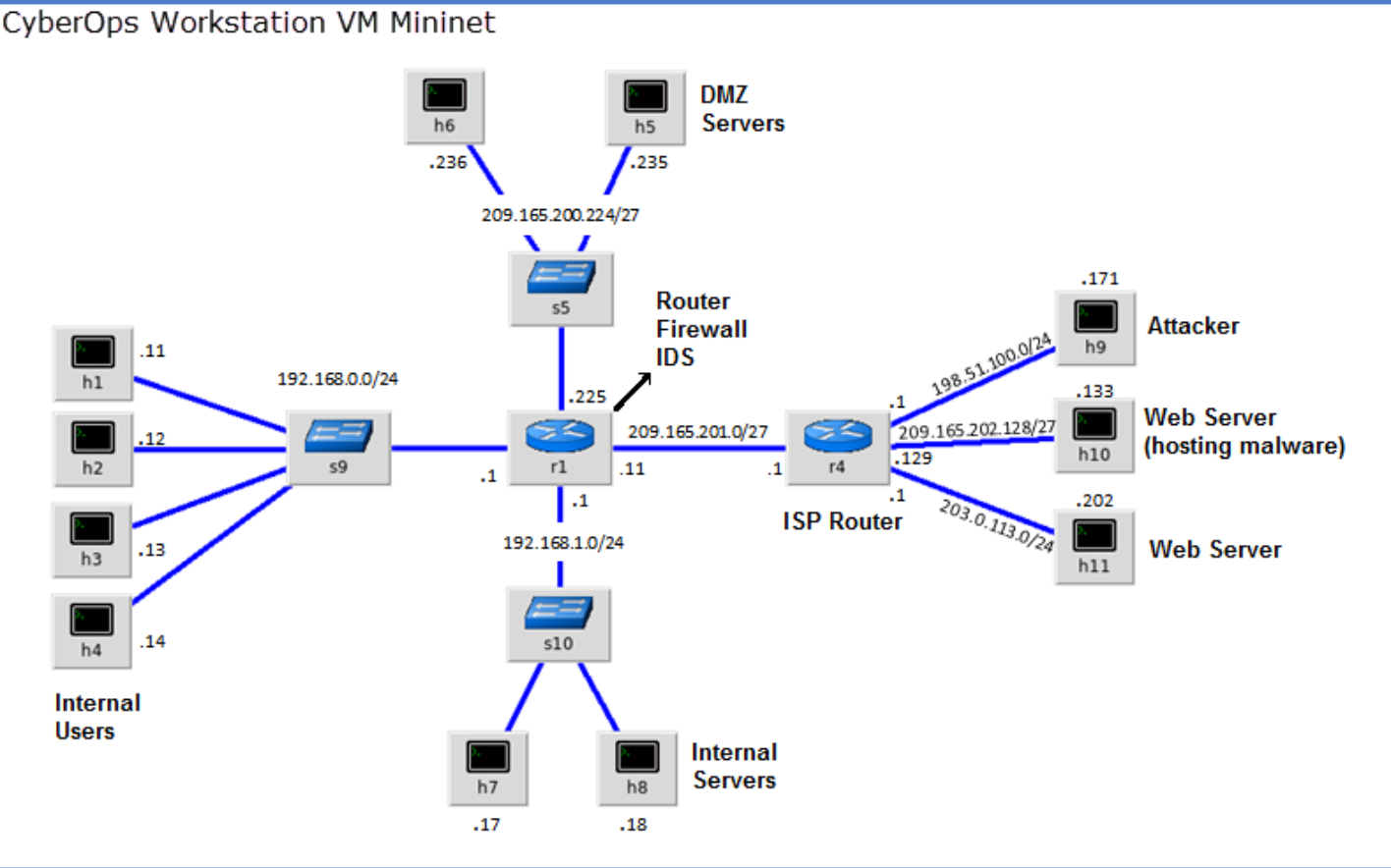


Figure 1. CyberOps Topology

# Part 1: Preparing the Virtual Environment

1. 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**.
2. 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 ~]$

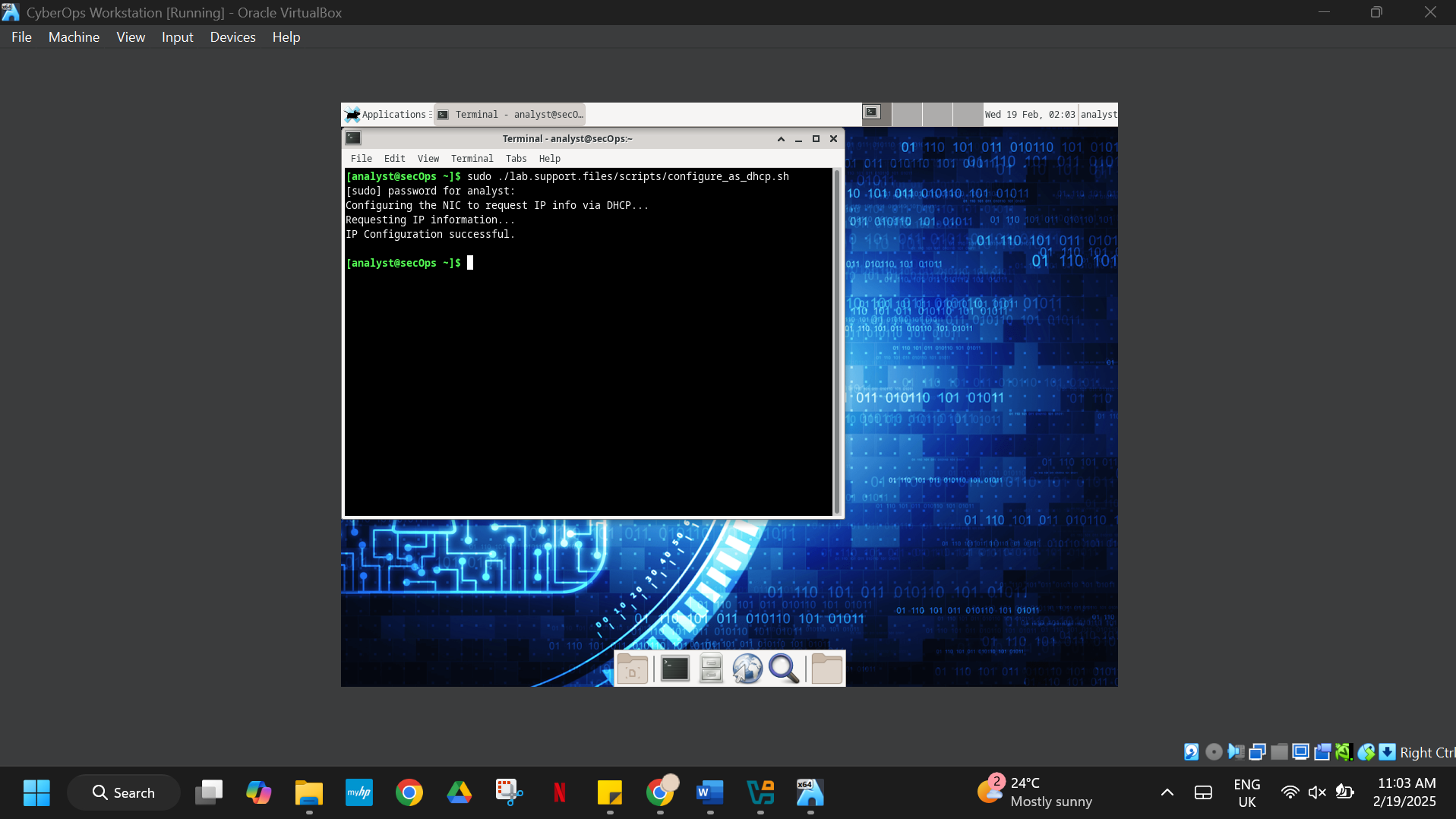


Figure 2. Requesting IP information via DHCP

1. 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

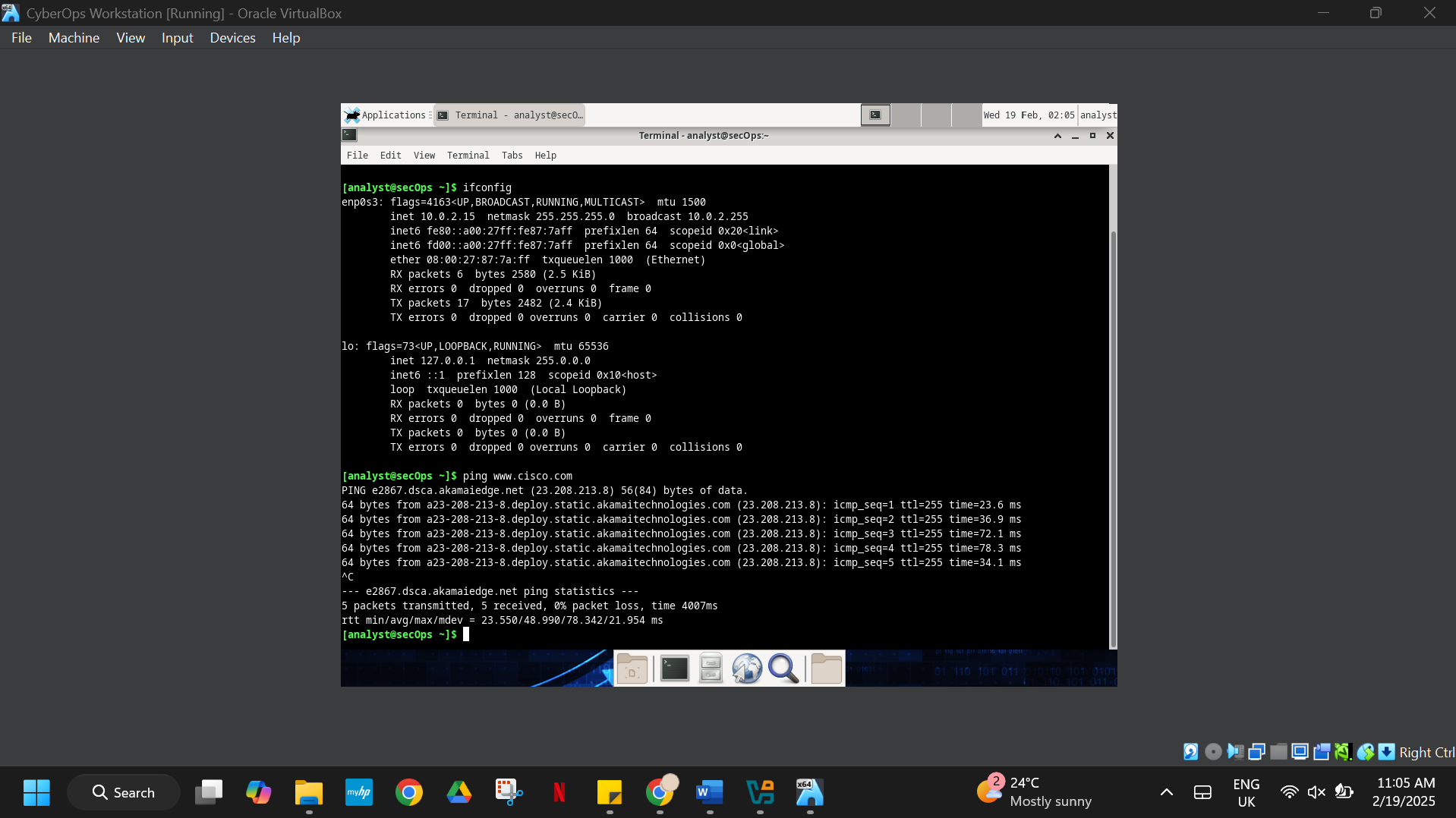


Figure 3. Testing connectivity

# 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

1. 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.

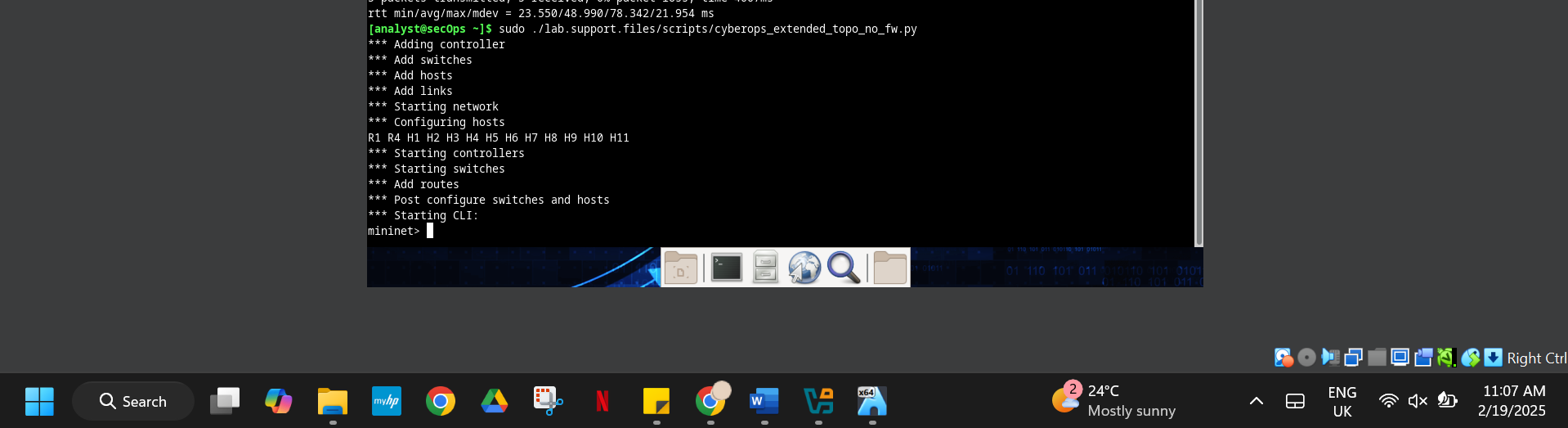


Figure 4. mininet prompt

1. From the **mininet** prompt, open a shell on **R1** using the command below:

mininet> **xterm R1**

mininet>

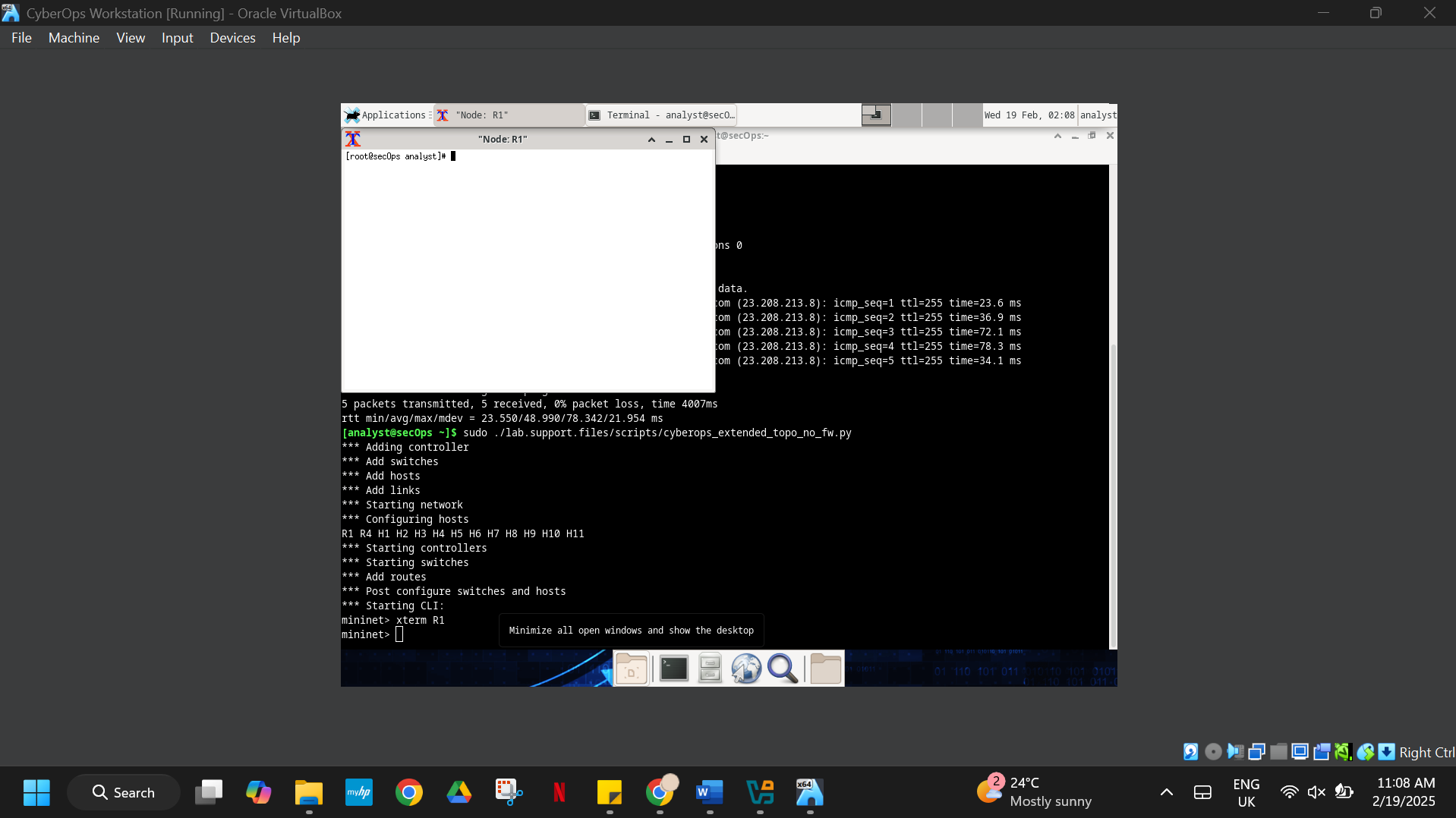


Figure 5. Opening the shell on **R1**

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?

In the terminal window it displays [root@secOps analyst] that indicates that the current logged in user is root, the hostname is secOps, and analyst is the home directory. The representation of the # instead of $ shows that there is root access.

1. 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.

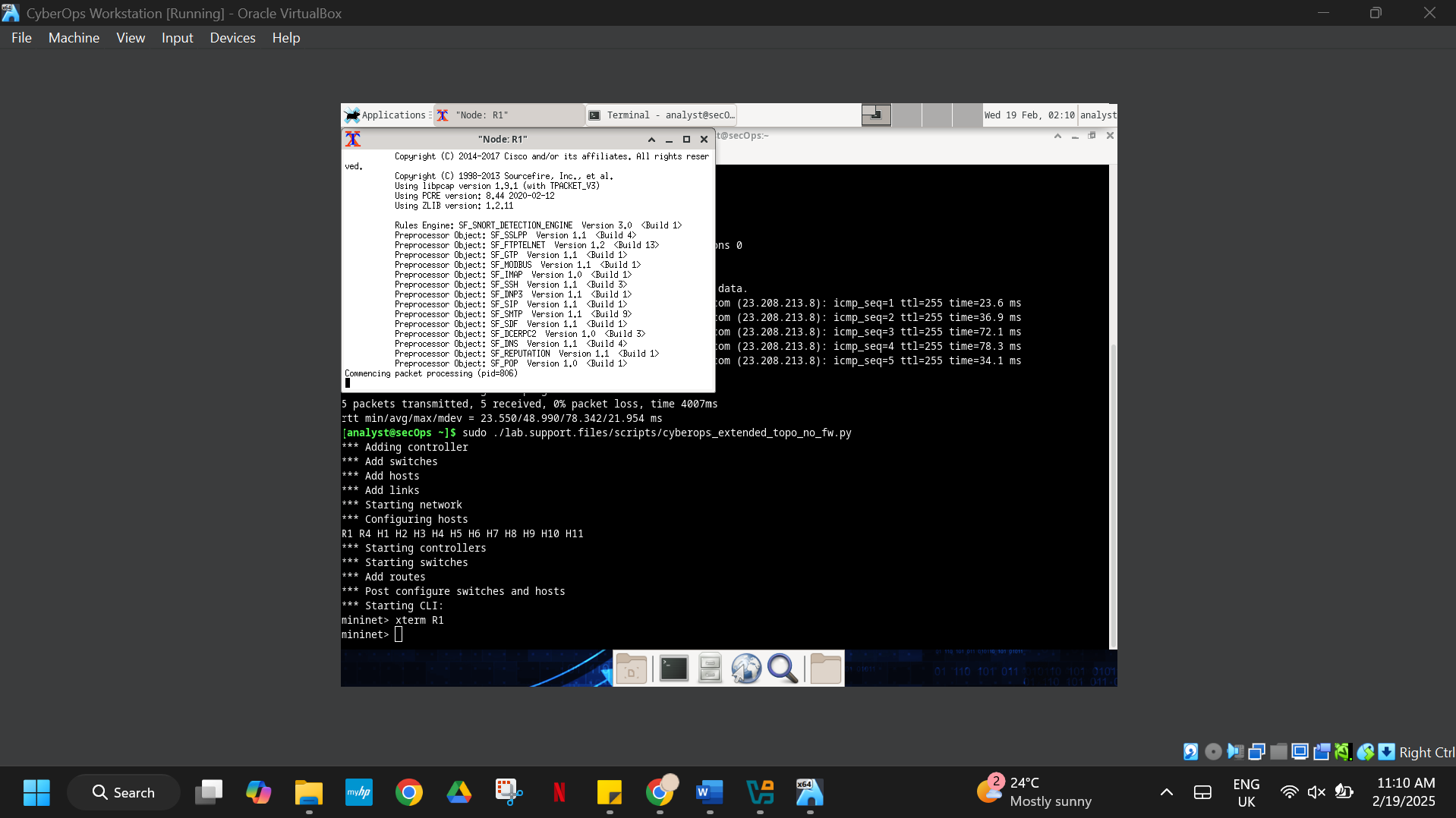


Figure 6. Starting Snort

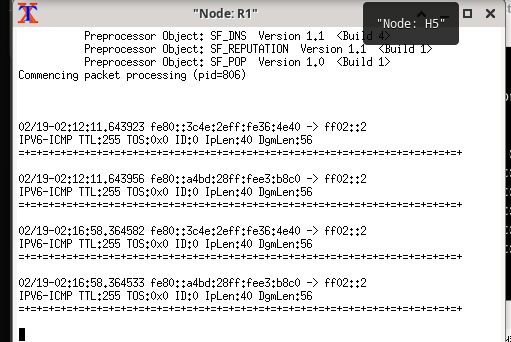


Figure 7. Prompt as Snort is now running

1. From the **CyberOps Workstation VM** **mininet** prompt, open shells for hosts **H5** and **H10**.

mininet> **xterm H5**

mininet> **xterm H10**

mininet>

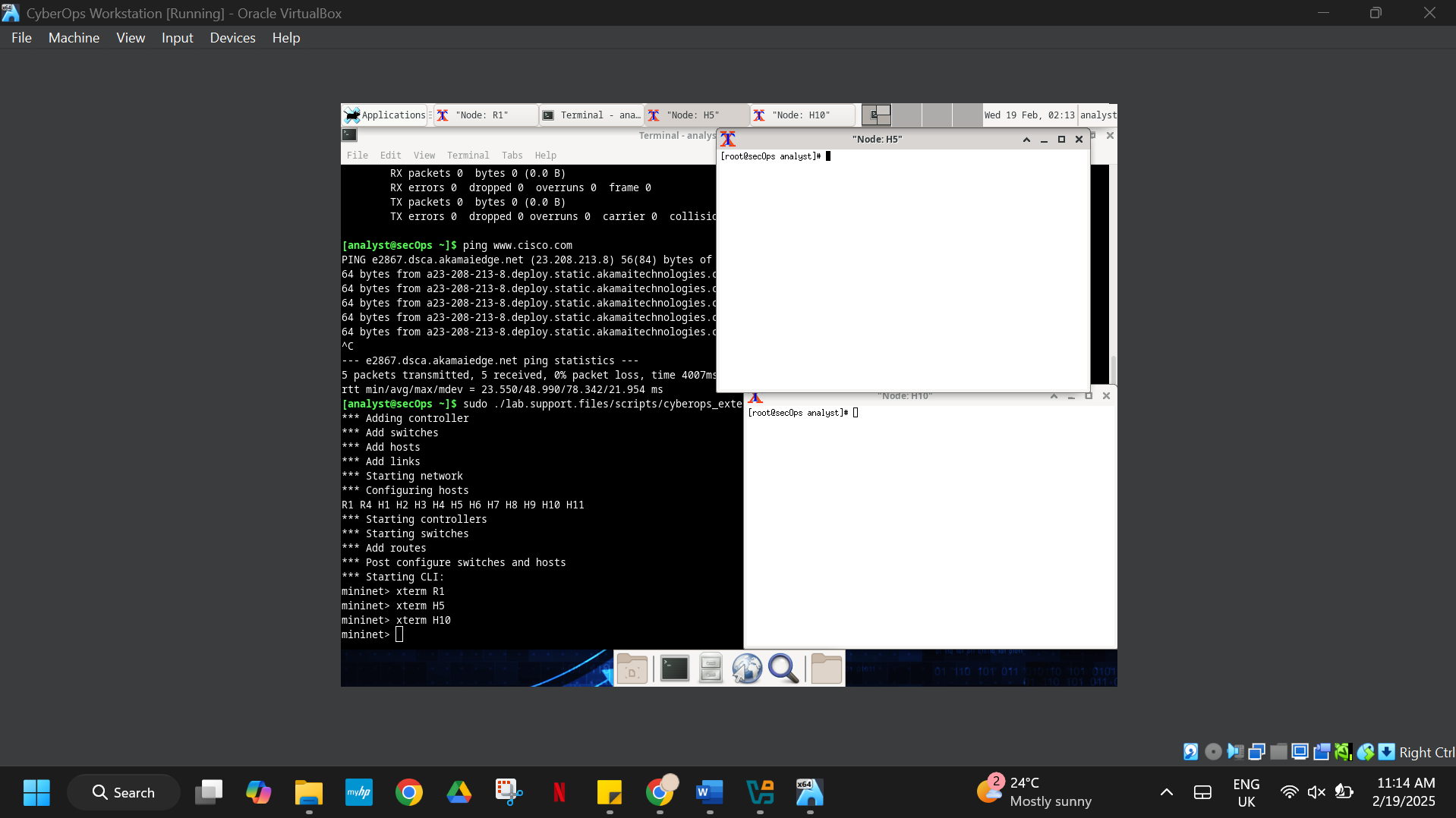
****

Figure 8. Opening shells for hosts H5 and H10

1. **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]#

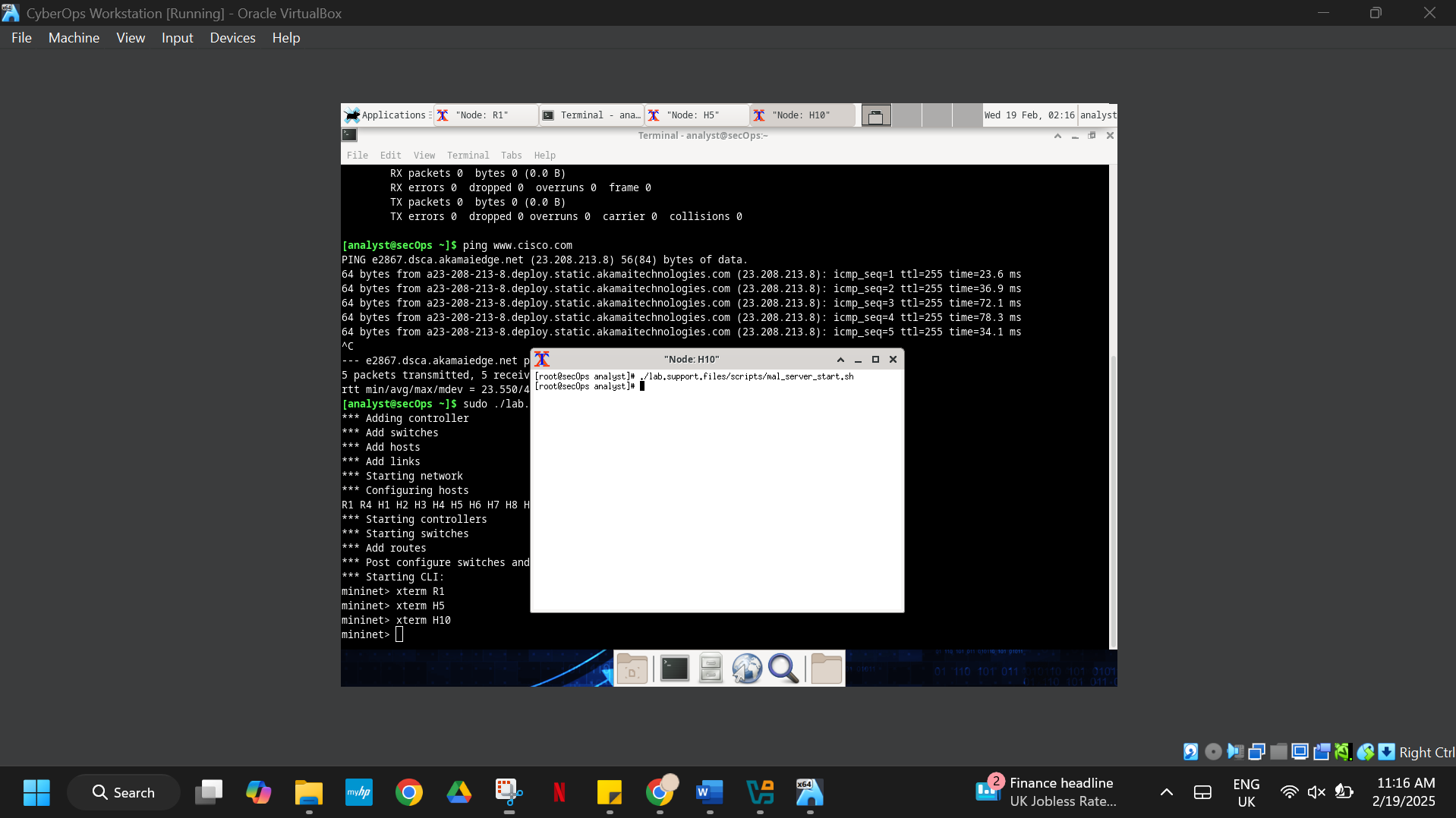


Figure 9. Starting the server on H10 shell

1. 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.

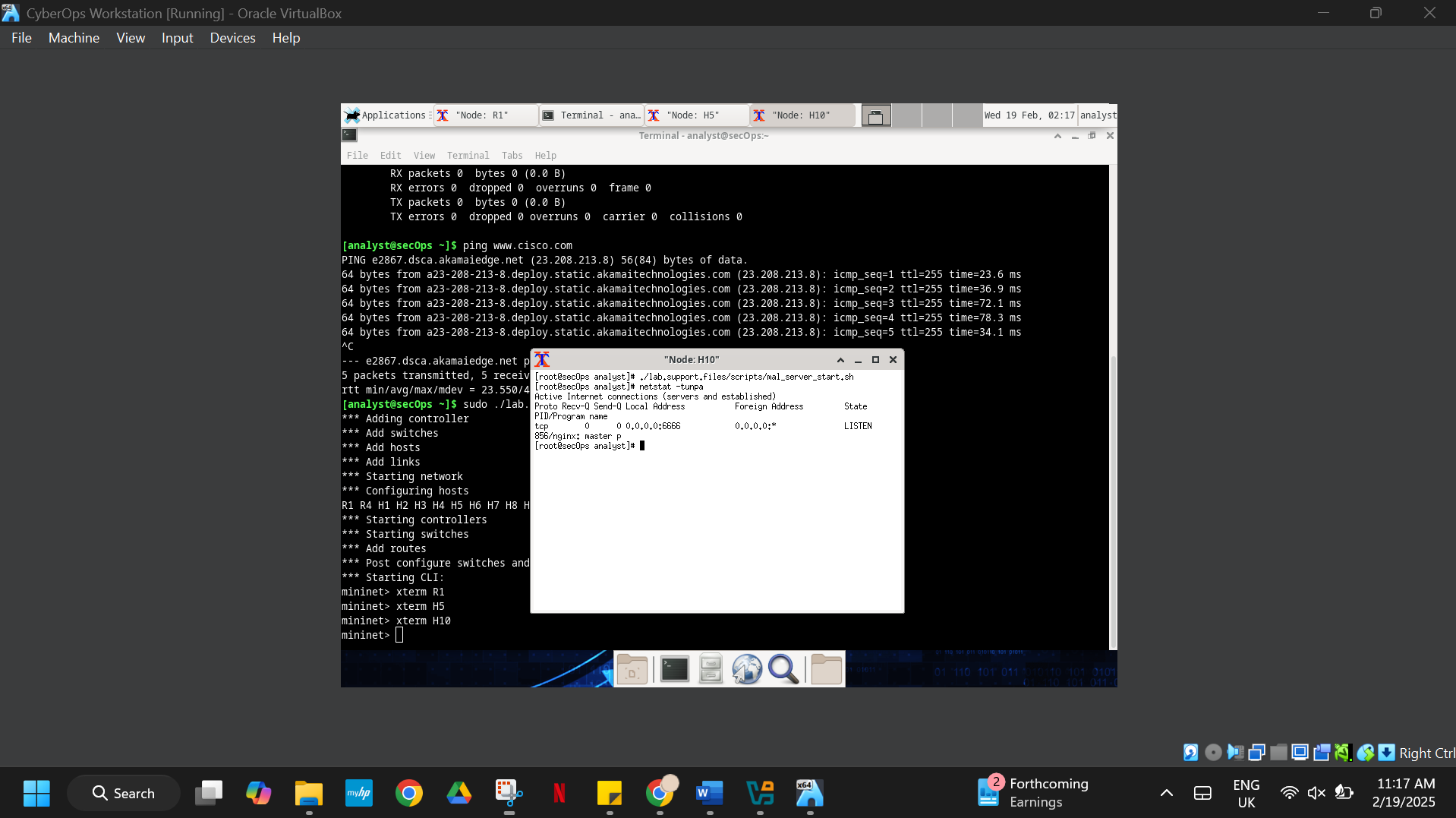


Figure 10. Confirmation of webserver nginx

1. 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.
2. 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.



Figure 11. No alerts shown in the log

1. 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]#

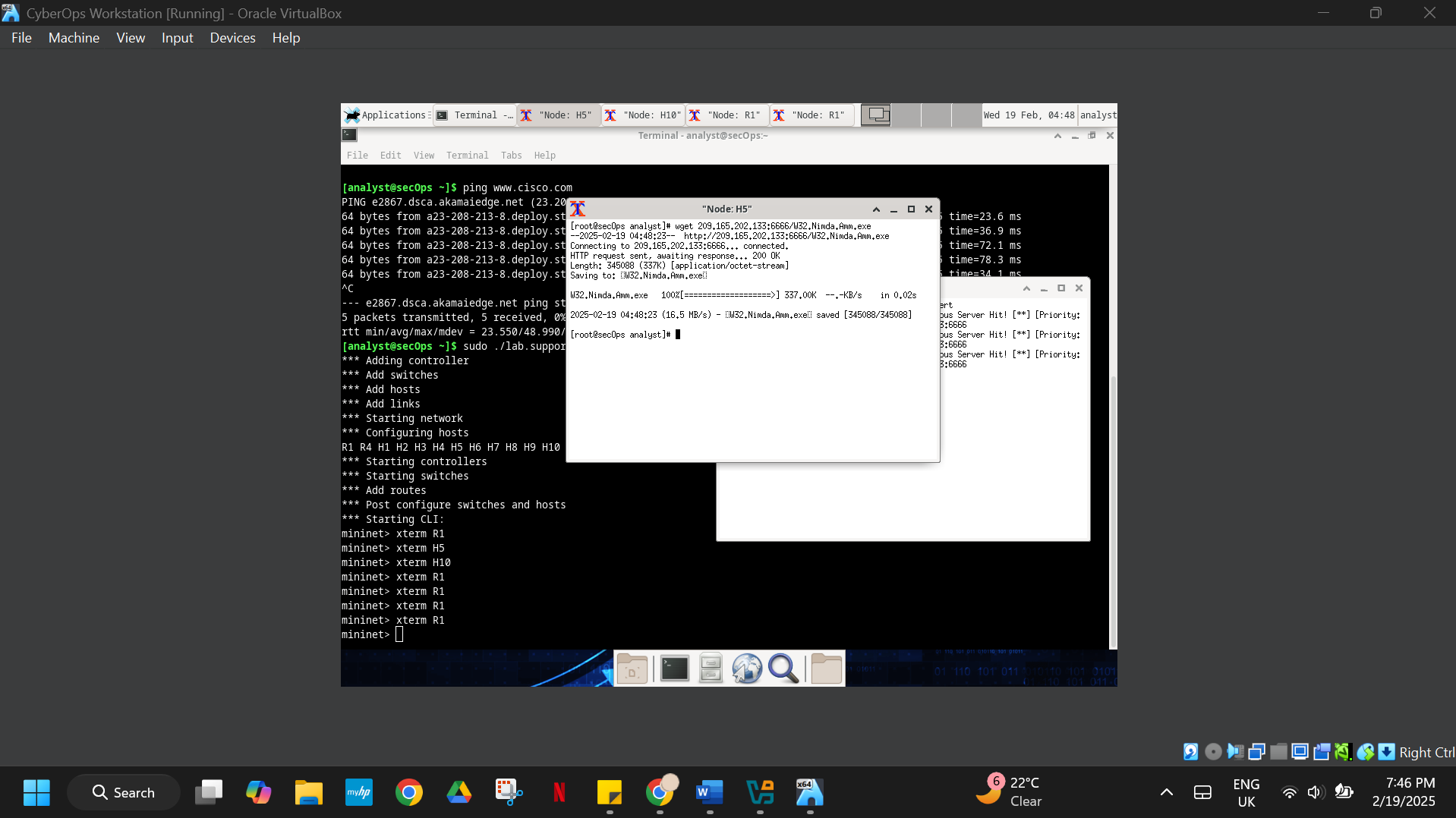


Figure 12. Download of file W32.Nimda.Amm.exe

What port is used when communicating with the malware web server? What is the indicator?

The port that is used in order to communicate with the malware web server is port 6666 that is shown in the URL after the colon “<http://209.165.202.133:6666/W32.Nimda.Amm.exe>”

Was the file completely downloaded?

yes, it is downloaded because the complete file transfer was verified during this operation. The wget command output on H5 displays that it saved the file W32.Nimda.Amm.exe with a size of 345088 bytes.

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

yes, The IDS (Snort) produced an alert because of the file download action. A "Malicious Server Hit" message appears through the “tail -f /var/log/snort/alert monitoring system”. with details of the source and destination IP addresses and ports.

1. 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

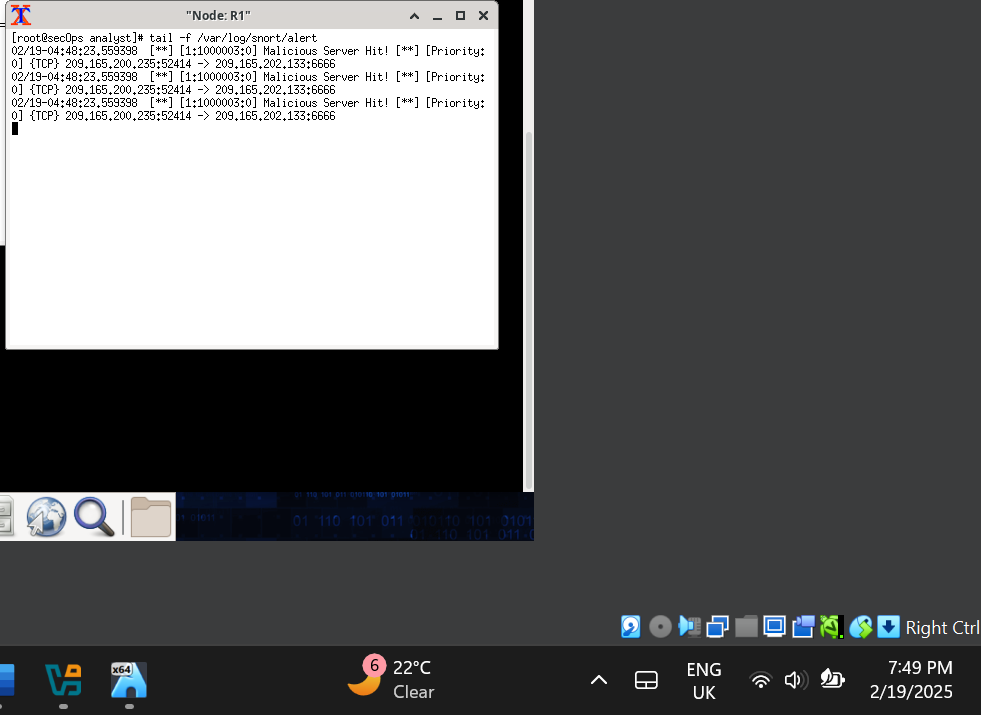


Figure 13. Alert log of Malicious Server

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

The source IPv4 address was **209.165.200.235**, and the destination IPv4 address was **209.165.202.133**.

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

The source port was **52141**, and the destination port was **6666**.

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

The download took place at **04:48:23** on **02/19**.

Based on the alert shown above, what was the message recorded by the IDS signature?

The message recorded by the IDS signature was "Malicious Server Hit!" with a priority of 0.

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.

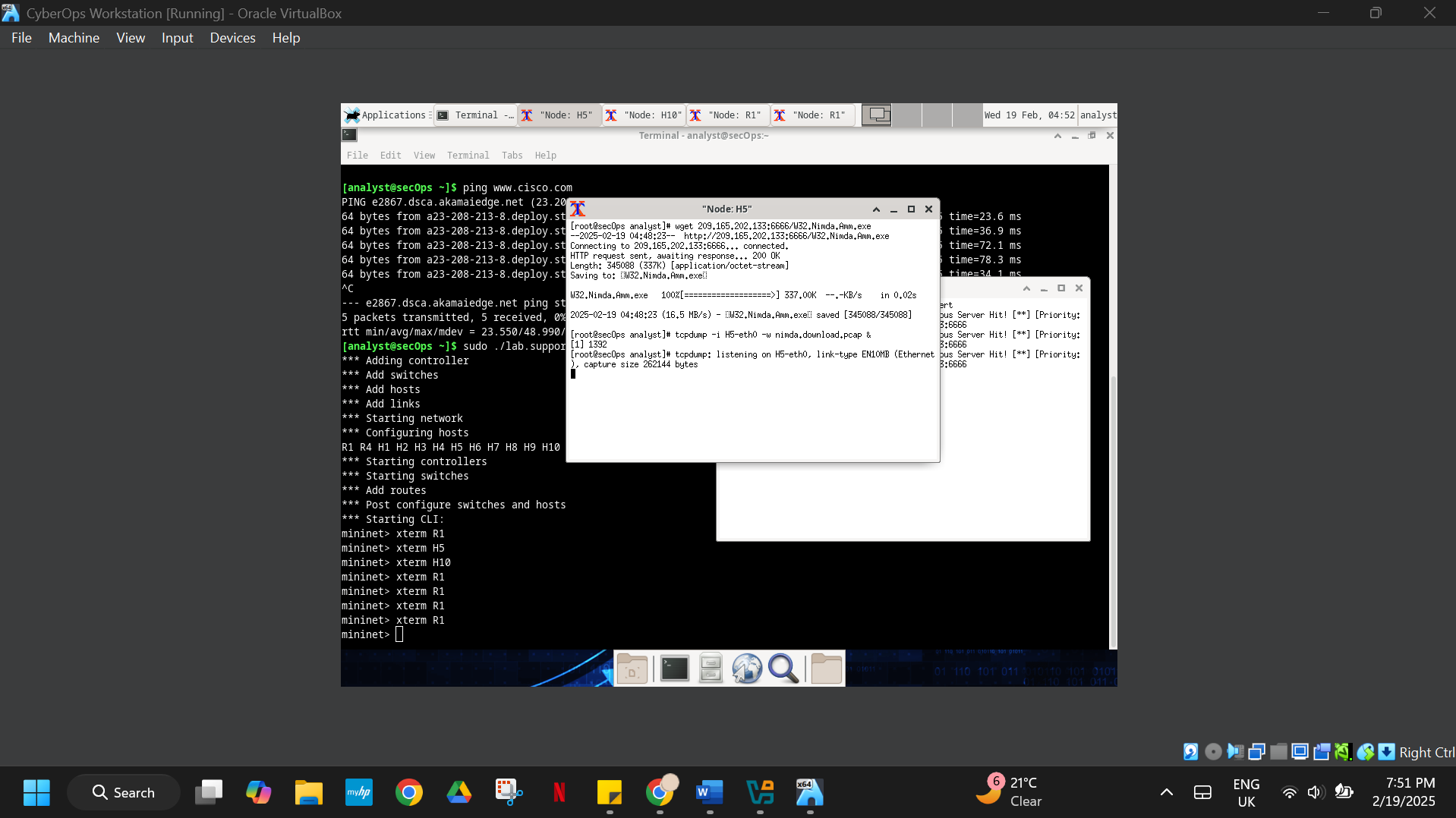
1. Press **ENTER** a few times to regain control of the shell while **tcpdump** runs in background.

Figure 14. Capturing tcpdump packets

1. 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]****Figure 15. Downloading file W32.Nimda.Amm.exe after capturing tcpdump

1. 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]#

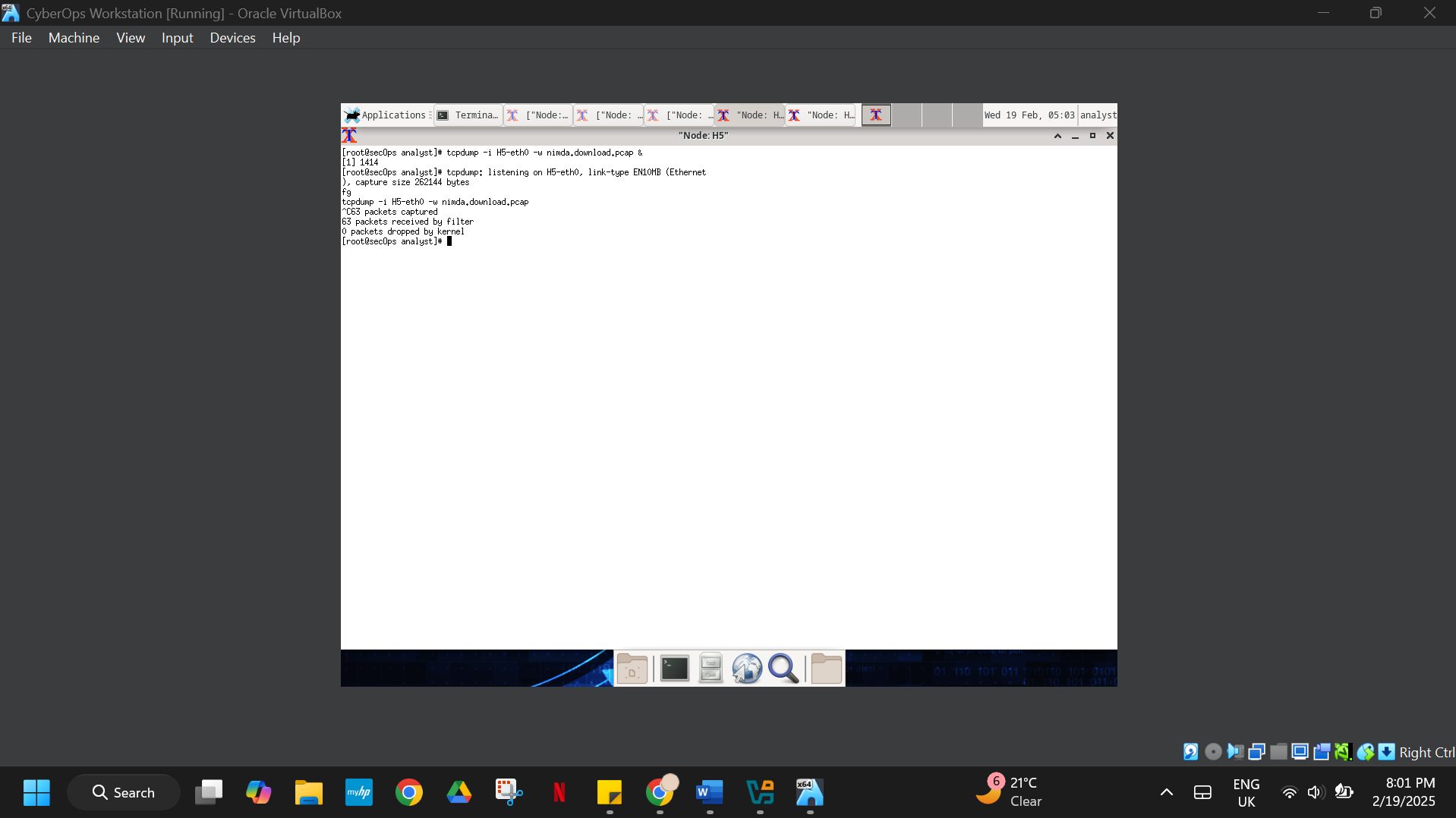


Figure 16. Stopping the capture by bringing tcpdump to foreground

1. 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]#

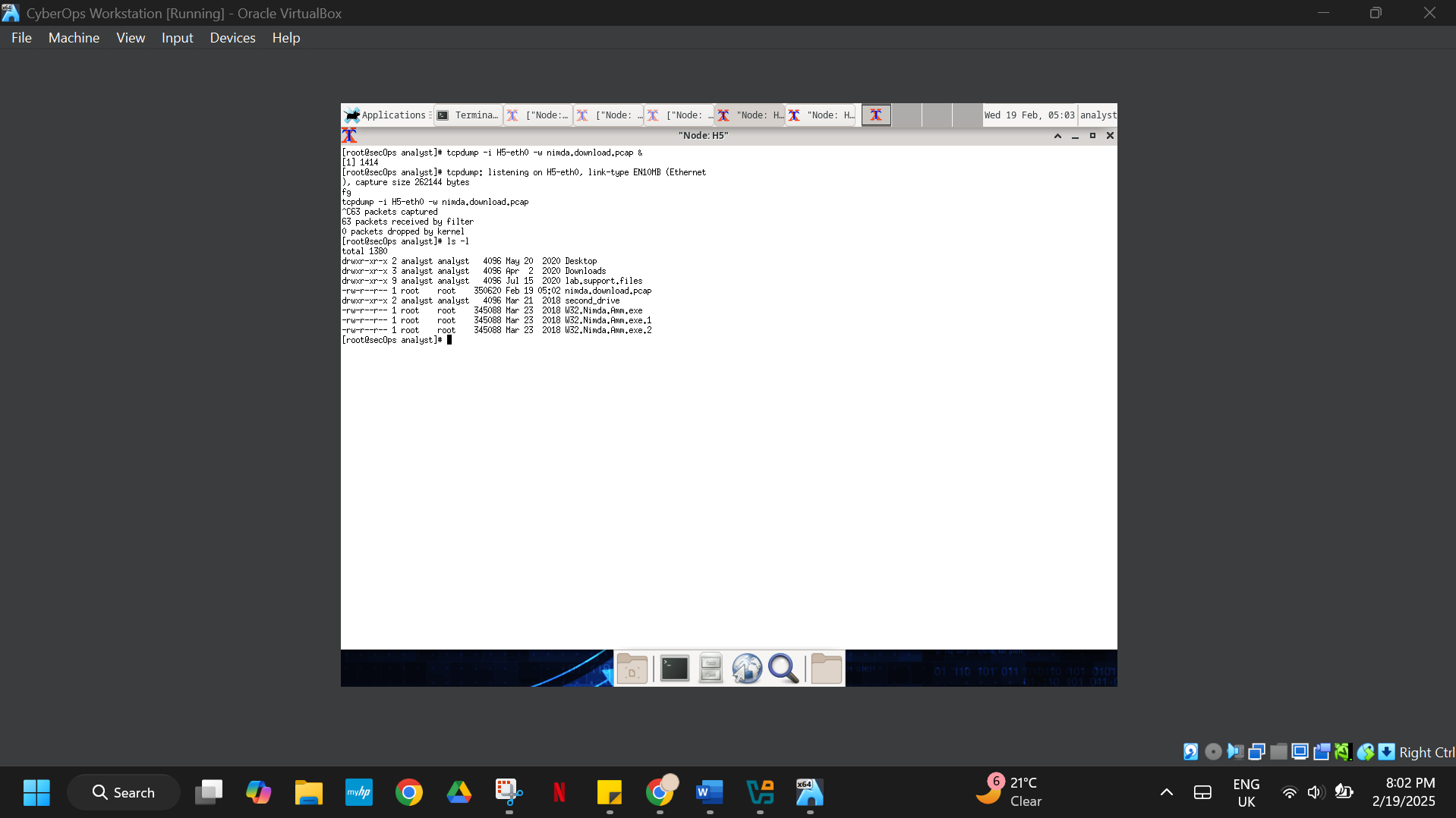


Figure 17. Verifying the pcap file

**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?

Analysis of network traffic during file download becomes possible by inspecting the PCAP file named nimda.download.pcap. Through packet inspection the analyst can study traffic characteristics for detecting malicious patterns and abnormal behavior (SolarWinds, n.d).

**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.

1. In the **CyberOps Workstation VM**, start a third R1 terminal window.

mininet > **xterm R1**

1. 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 ~]#

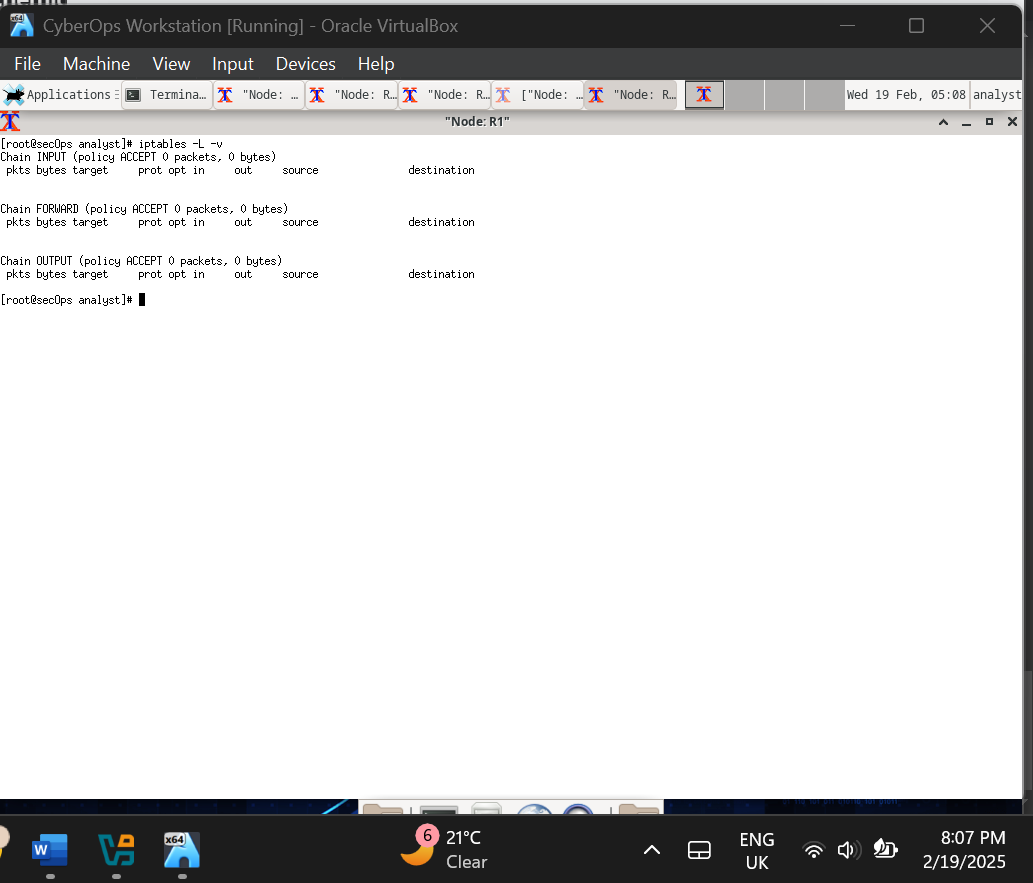


Figure 18. iptables

What chains are currently in use by **R1**?

The chains currently in use by R1 are INPUT, FORWARD, and OUTPUT, as shown by the (iptables -L -v) command output.

1. 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 ~]#

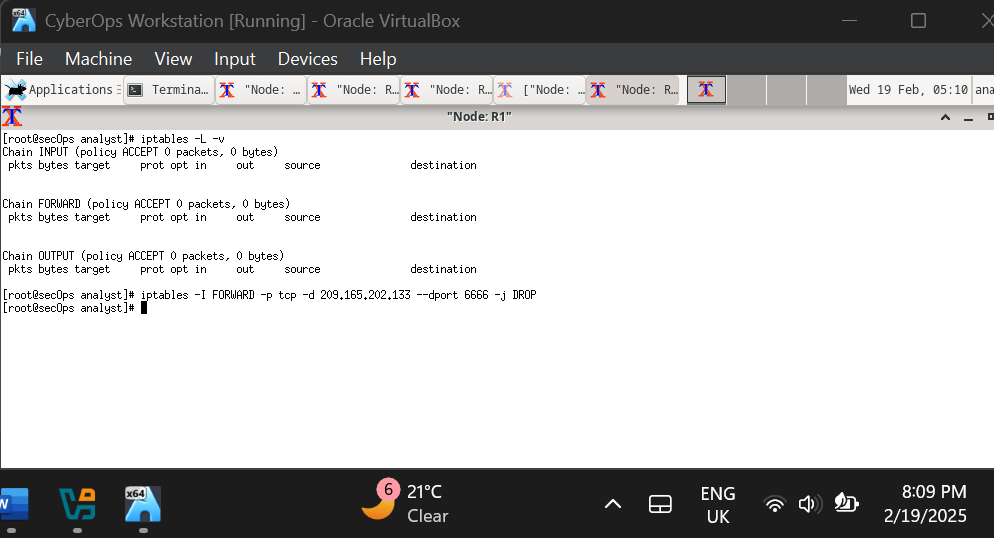


Figure 19. Forwarding the chain on R1

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.

1. 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)

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]#

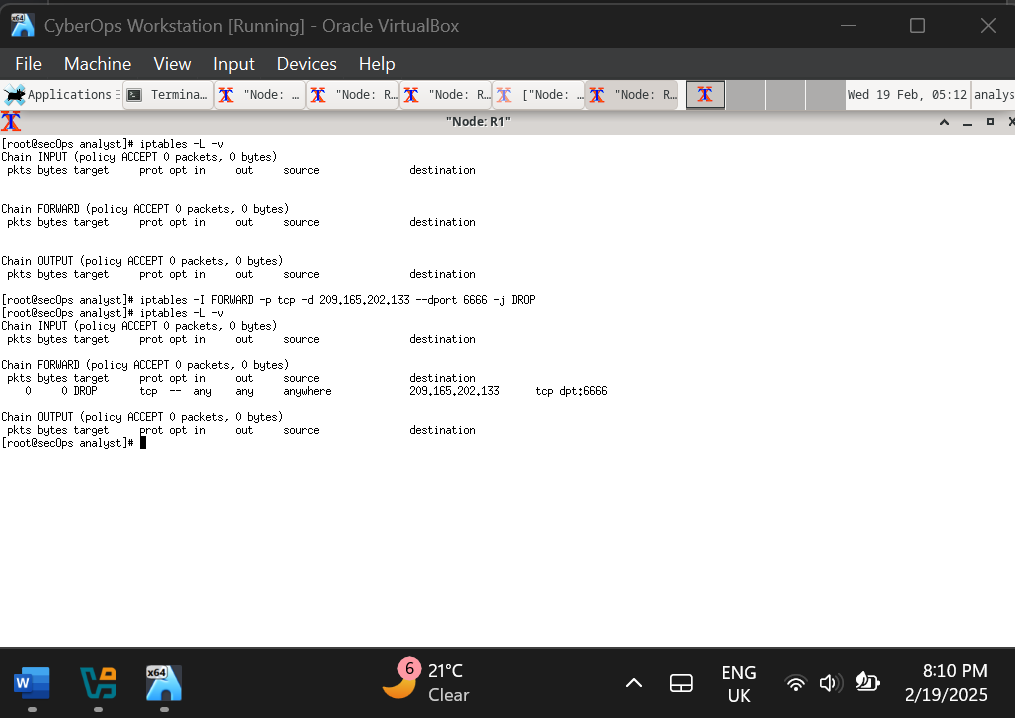


Figure 20. Confirmation that rule was added to the FORWARD chain

1. 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.

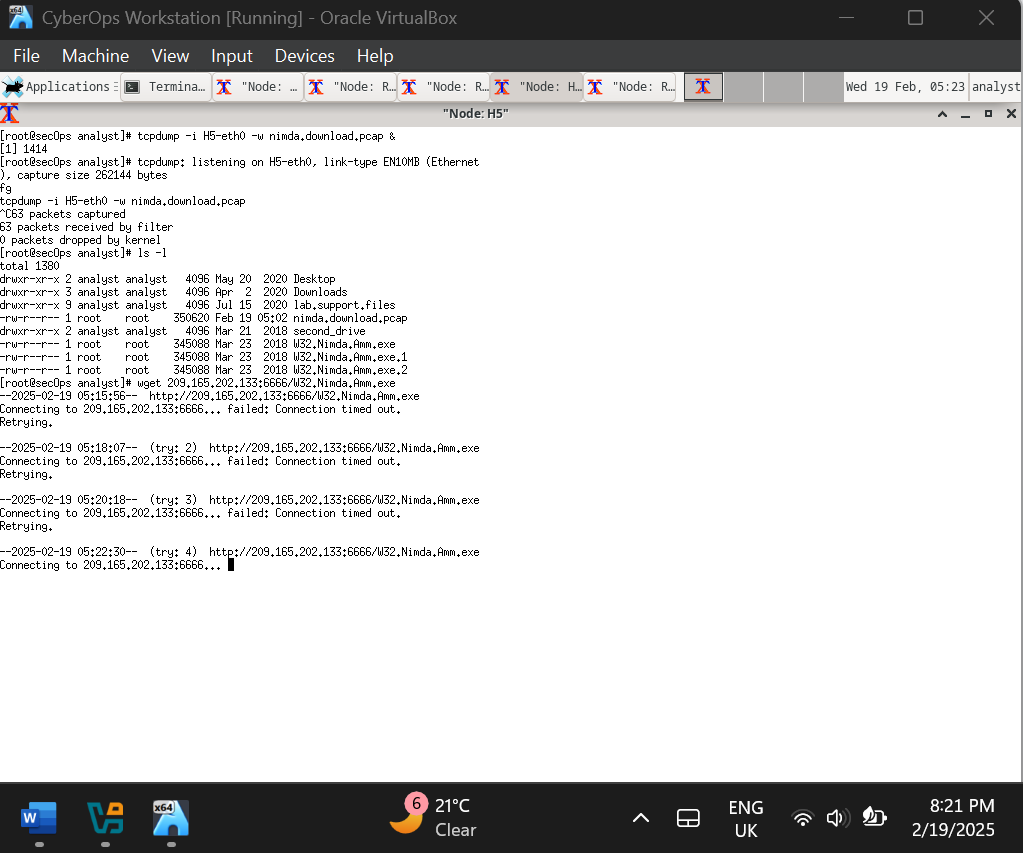


Figure 21. Downloading the file after Forwarding

Was the download successful this time? Explain.

The download operation did not succeed this time because the connection timed out. The timeout of the connection happened because R1's FORWARD chain firewall rule blocked access to the malicious port 6666 server.

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

The aggressive method involves blocking entire traffic accessing or leaving an offending server through its IP address no matter which port it uses. The security rule should block all packet traffic to and from IP address 209.165.202.133 by implementing a new blocking protocol.

***Type your answers here.***

# Part 3: Terminate and Clear Mininet Process

1. Navigate to the terminal used to start Mininet. Terminate the Mininet by entering **quit** in the main CyberOps VM terminal window.

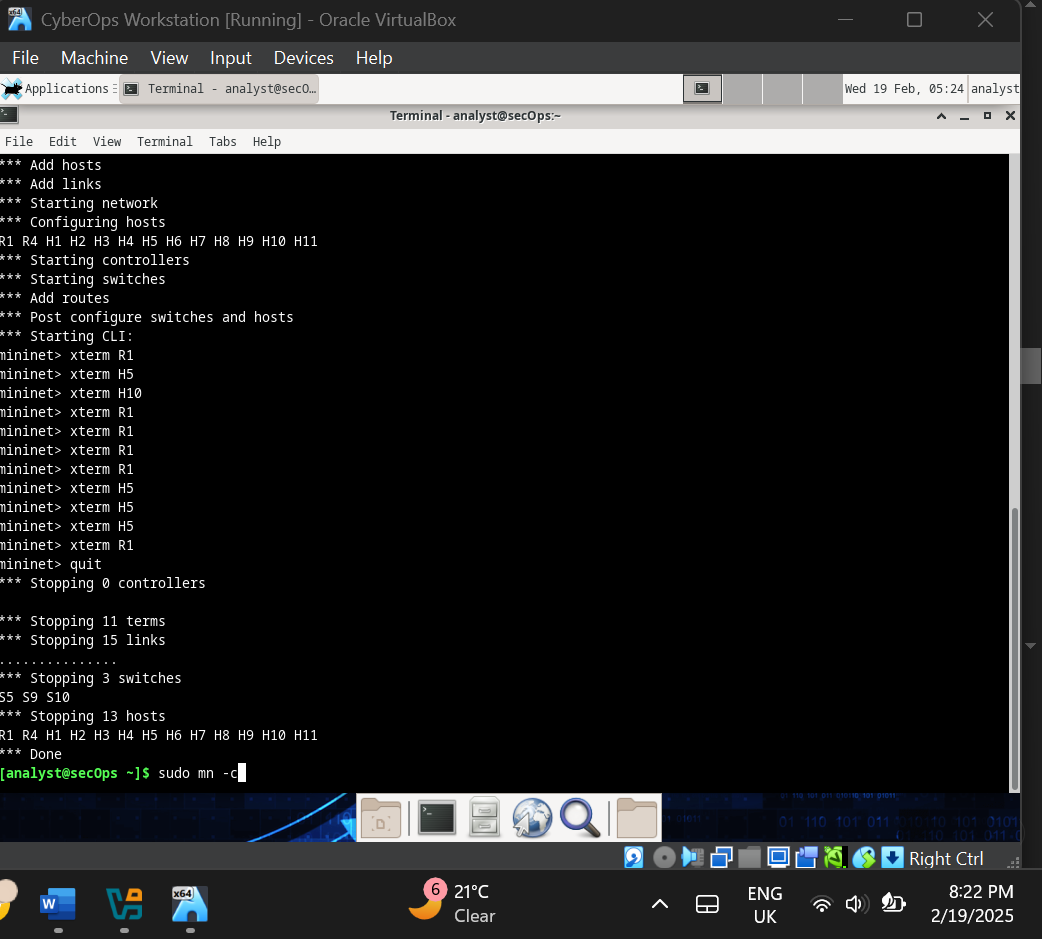


Figure 22. Terminating the Mininet

1. 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:

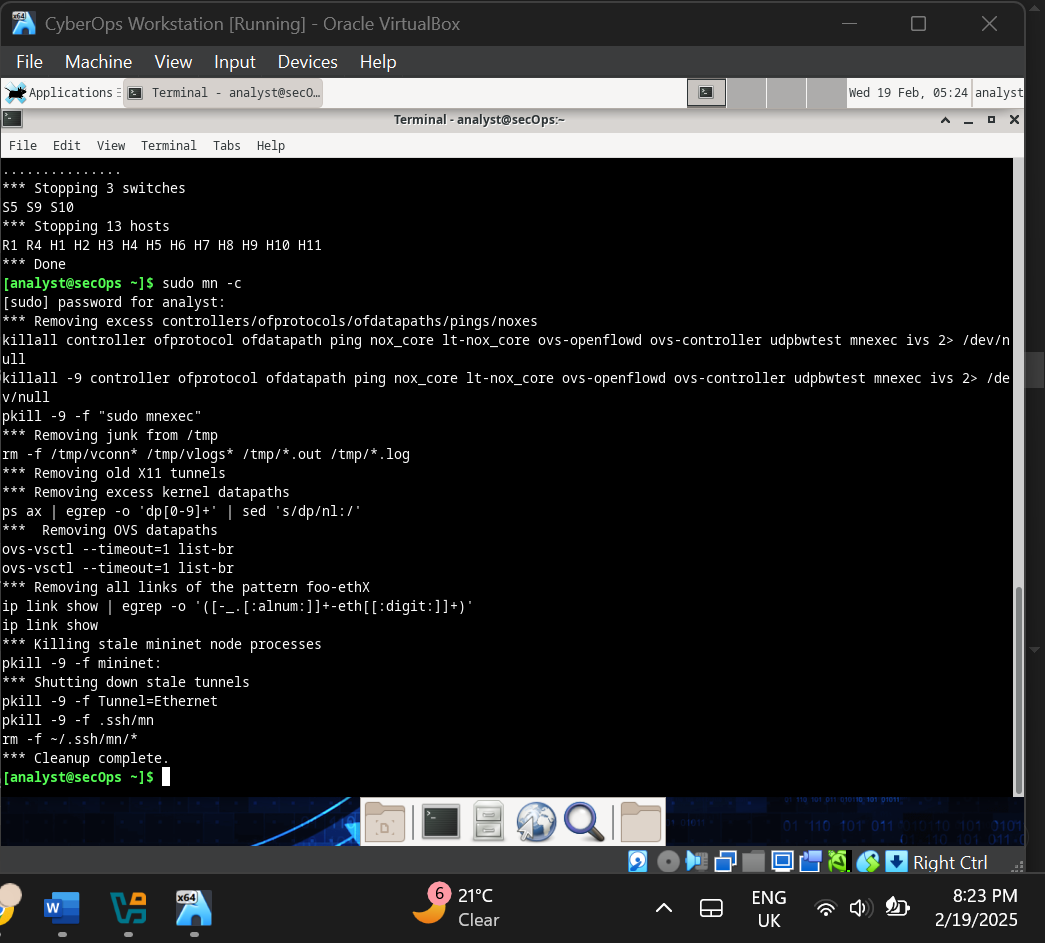


Figure 23. Cleaning up the processes started by Mininet

# CONCLUSION

The Lab teaching experience allowed us to examine intrusion detection through Snort and observe how firewalls operate using iptables. The main objective focused on understanding how these tools perform detection and remediation of network threats. We observed Snort produce alerts from defined rules during the malware download simulation while learning about the required configurations to stop dangerous network traffic. Lifeguards operating monitoring systems proved their ability to identify security threats by warning users about dangerous network situations. Network security protection depends critically on IDS because Snort successfully detected the malicious file download by analyzing traffic signatures in the database. We earned experience in building firewall rules based on iptables to guard against the dangerous server without letting unauthorized downloads enter our systems.

The exercise helped us understand how network security becomes stronger by integrating IDS with firewall technologies. Security analysts obtain better defense capabilities by analyzing alerts well to create new firewall rules that target specific threats. The laboratory develops essential lessons about uninterrupted monitoring needs and fast reaction capabilities that work together with security tools which form an active defense network infrastructure.

# References

*SolarWinds*. (n.d.). What Is Packet Capture (PCAP)? https://www.solarwinds.com/resources/it-glossary/pcap