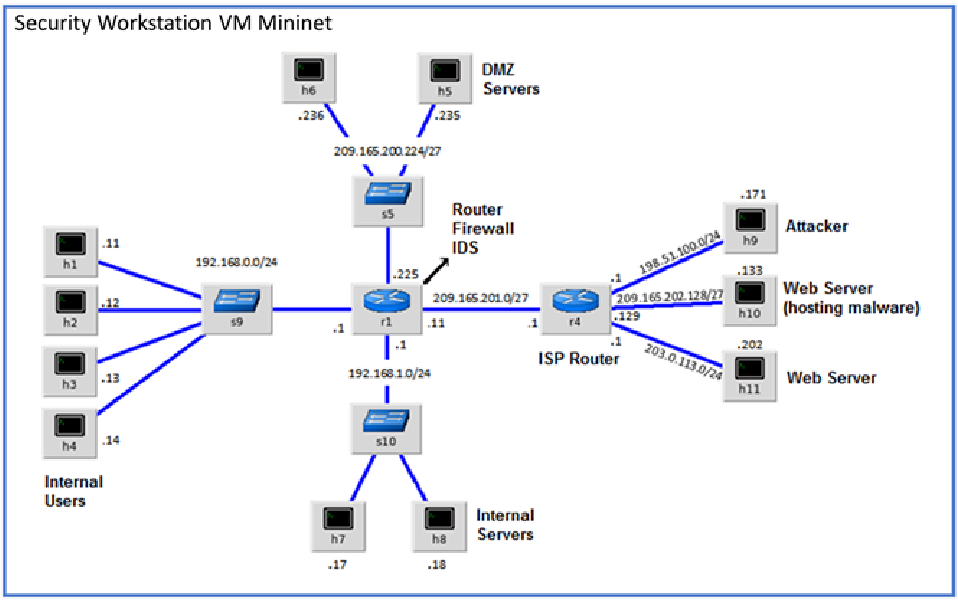
Snort and Firewall Rules

Topology



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.

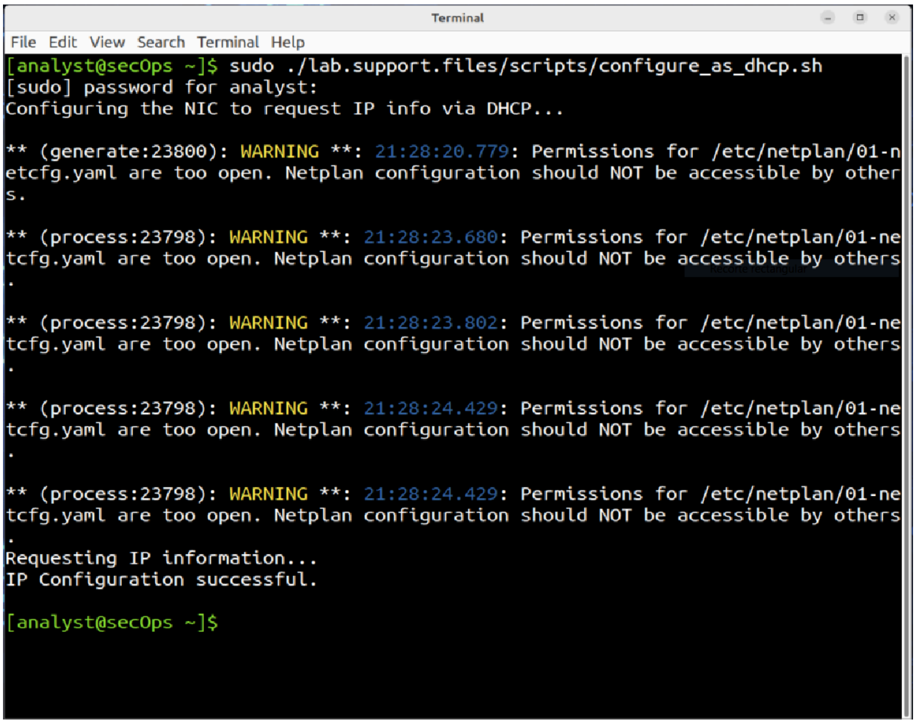
Part 1: Preparing the Virtual Environment

a. Launch Oracle VirtualBox and change the Security 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 a NAT adapter) and click OK.
b. Launch the Security 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.

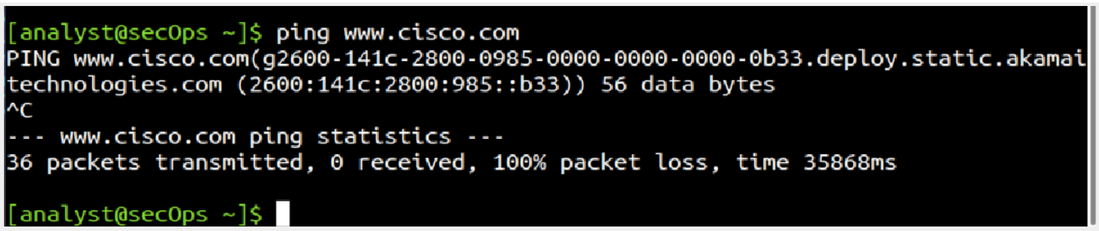
```
[analyst@secOps ~]$ sudo
./lab.support.files/scripts/configure_as_dhcp.sh
```

```
[sudo] password for analyst:
[analyst@secOps ~]$
```



```
Terminal
File Edit View Search Terminal Help
[analyst@secOps ~]$ sudo ./lab.support.files/scripts/configure_as_dhcp.sh
[sudo] password for analyst:
Configuring the NIC to request IP info via DHCP...

** (generate:23800): WARNING **: 21:28:20.779: Permissions for /etc/netplan/01-n
etcfg.yaml are too open. Netplan configuration should NOT be accessible by other
s.

** (process:23798): WARNING **: 21:28:23.680: Permissions for /etc/netplan/01-ne
tcfg.yaml are too open. Netplan configuration should NOT be accessible by others
.

** (process:23798): WARNING **: 21:28:23.802: Permissions for /etc/netplan/01-ne
tcfg.yaml are too open. Netplan configuration should NOT be accessible by others
.

** (process:23798): WARNING **: 21:28:24.429: Permissions for /etc/netplan/01-ne
tcfg.yaml are too open. Netplan configuration should NOT be accessible by others
.

** (process:23798): WARNING **: 21:28:24.429: Permissions for /etc/netplan/01-ne
tcfg.yaml are too open. Netplan configuration should NOT be accessible by others
.
Requesting IP information...
IP Configuration successful.

[analyst@secOps ~]$
```

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



```
[analyst@secOps ~]$ ping www.cisco.com
PING www.cisco.com(g2600-141c-2800-0985-0000-0000-0000-0b33.deploy.static.akamai
technologies.com (2600:141c:2800:985::b33)) 56 data bytes
^C
--- www.cisco.com ping statistics ---
36 packets transmitted, 0 received, 100% packet loss, time 35868ms

[analyst@secOps ~]$
```

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 of 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 Security Workstation VM, run the script to start mininet.

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

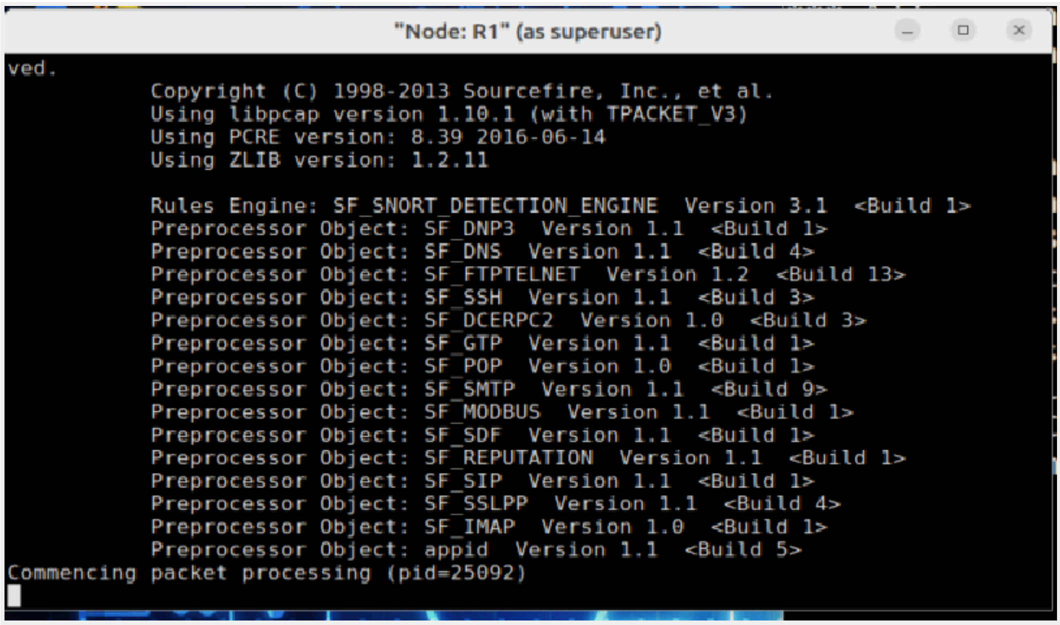
```
mininet>
```

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?

R: The root user. This is indicated by the # sign after the prompt.

c. From R1’s shell, start the Linux-based IDS, Snort.

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

Running in IDS mode

--== Initializing Snort ==--

Initializing Output Plugins!

Initializing Preprocessors!

Initializing Plug-ins!

Parsing Rules file "/etc/snort/snort.conf"

<output omitted>
```



Note: You will not see a prompt as Snort is now running in this window. If for any reason, Snort stops running and the `[root@secOps analysts]#` prompt is displayed, rerun the script to launch Snort. Snort must be running to capture alerts later in the lab.

d. From the Security Workstation VM mininet prompt, open shells for hosts H5 and H10.

```
mininet> xterm H5

mininet> xterm H10
```

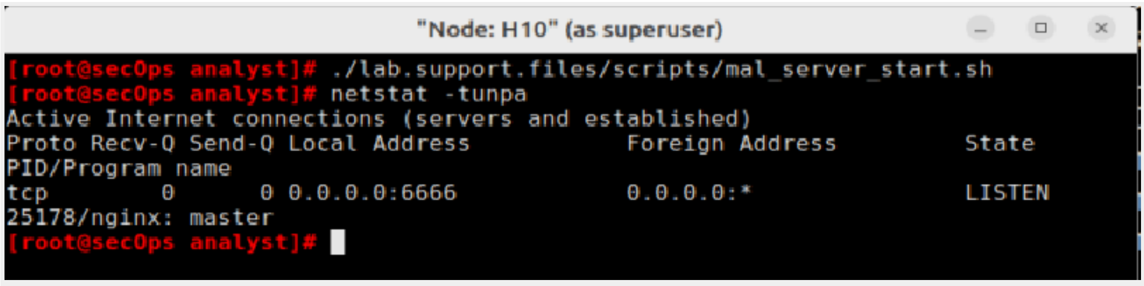
```
mininet>
```

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:



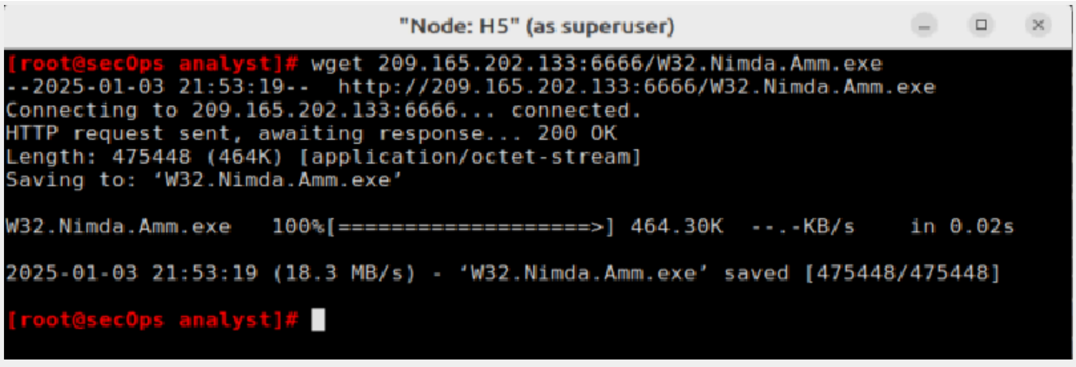
As seen by the output above, the lightweight web server nginx is running and listening to connections on port TCP 6666.

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 Security Workstation VM terminal window. You may also want to arrange the terminal windows so that you can see and interact with each device.
h. In the new R1 terminal tab, run the tail command with the `-f` option to monitor the `/var/log/snort/alert` file in real-time. This file is where snort is configured to record alerts.

```
[root@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.



```
"Node: H5" (as superuser)
[root@secOps analyst]# wget 209.165.202.133:6666/W32.Nimda.Amm.exe
--2025-01-03 21:53:19--  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: 475448 (464K) [application/octet-stream]
Saving to: 'W32.Nimda.Amm.exe'

W32.Nimda.Amm.exe   100%[===================>] 464.30K  --.-KB/s    in 0.02s

2025-01-03 21:53:19 (18.3 MB/s) - 'W32.Nimda.Amm.exe' saved [475448/475448]

[root@secOps analyst]#
```

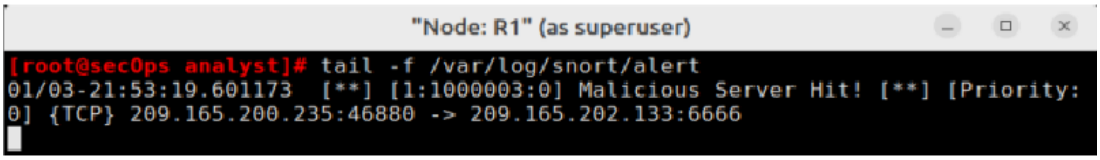
What port is used when communicating with the malware web server? What is the indicator?

R: Port 6666. The port was specified in theURL, after the : separator.

Was the file completely downloaded?

R: Yes

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



```
"Node: R1" (as superuser)
[root@secOps analyst]# tail -f /var/log/snort/alert
01/03-21:53:19.601173  [**] [1:1000003:0] Malicious Server Hit! [**] [Priority:
0] {TCP} 209.165.200.235:46880 -> 209.165.202.133:6666
```

R: Yes

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 shown below. Your timestamp will be different:

```
01/03-21:53:19.601173  [**] [1:1000003:0] Malicious Server
Hit! [**] [Priority: 0] {TCP} 209.165.200.235:46880 ->
209.165.202.133:6666
```

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

R: Source IP: 209.165.200.235; DestinationIP: 209.165.202.133.

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

R: Source port:46880; Destination port: 6666.

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

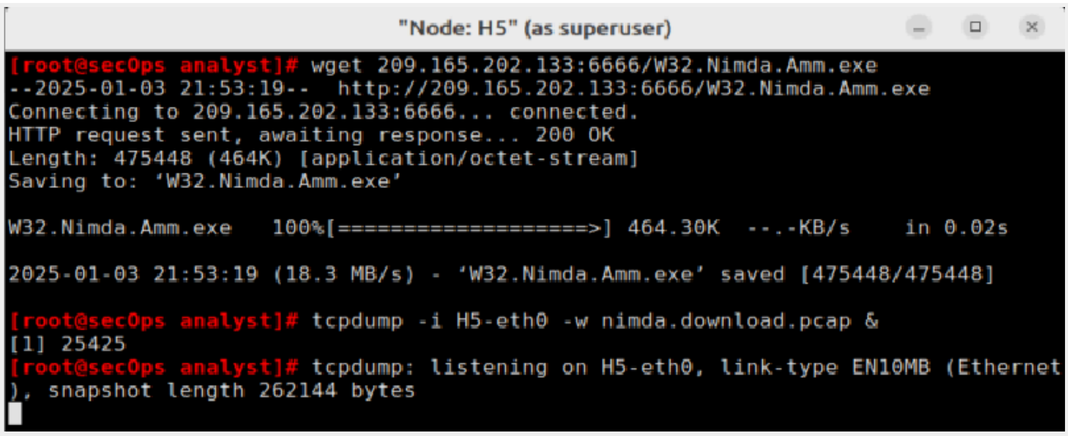
R: January 3rd at 9:53

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

R: “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 &
```



```
"Node: H5" (as superuser)
[root@secOps analyst]# wget 209.165.202.133:6666/W32.Nimda.Amm.exe
--2025-01-03 21:53:19--  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: 475448 (464K) [application/octet-stream]
Saving to: ‘W32.Nimda.Amm.exe’

W32.Nimda.Amm.exe   100%[===================>] 464.30K  --.-KB/s    in 0.02s

2025-01-03 21:53:19 (18.3 MB/s) - ‘W32.Nimda.Amm.exe’ saved [475448/475448]

[root@secOps analyst]# tcpdump -i H5-eth0 -w nimda.download.pcap &
[1] 25425
[root@secOps analyst]# tcpdump: listening on H5-eth0, link-type EN10MB (Ethernet
), snapshot length 262144 bytes
```

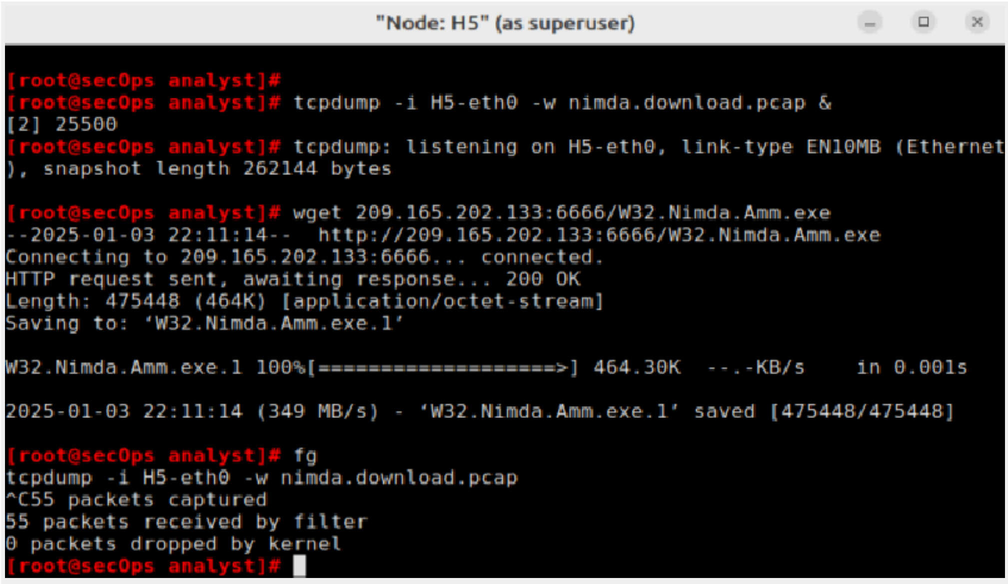
The command above instructs tcp dump to capture packets on interface H5-eth0 and save the capture to a file named nimda.download.pcap.

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] 25425`; it indicates one process was sent to background and its process ID (PID) is 25425.

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

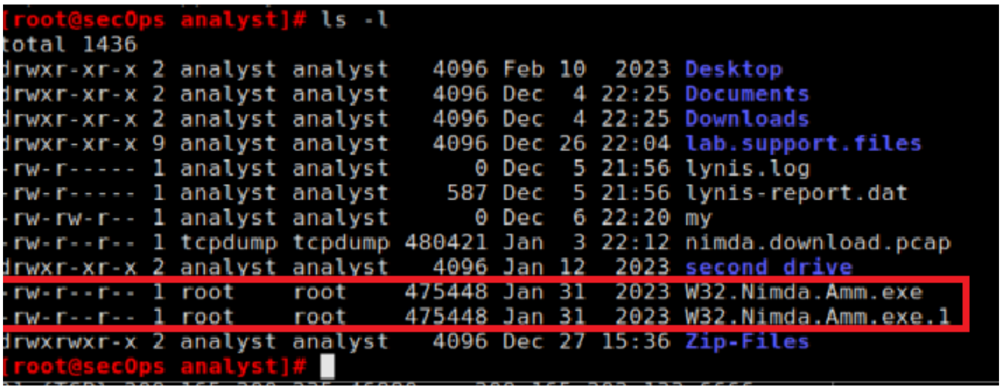
```
[root@secOps analyst]# wget
209.165.202.133:6666/W32.Nimda.Amm.exe
```

l. 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.



m. 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
```



How can this PCAP file be useful to the security analyst?

R: PCAP files contain the packets related to the traffic seen by the capturing NIC. In that way, the PCAP is very useful to re-retrace network events such as communication to malicious end points.

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.

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 anyone 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.

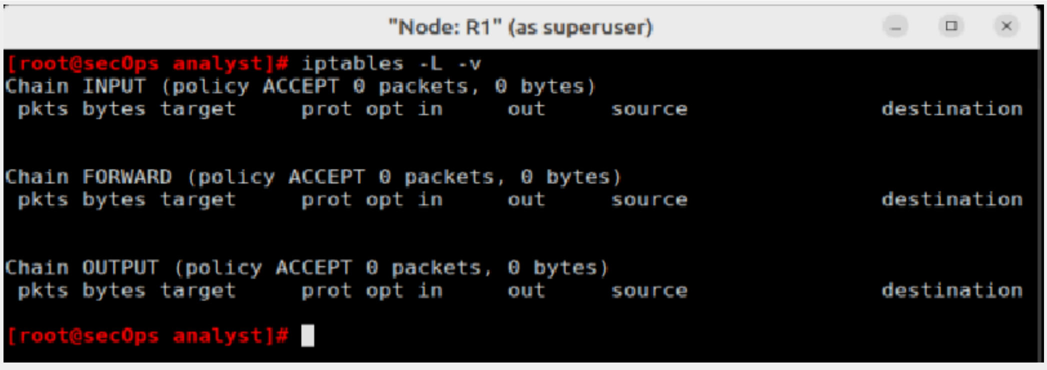
Rules are created to check specific characteristics of packets, allowing administrators to create very comprehensive filters. If a packet doesn't match a rule, the firewall moves on to the next rule and checks again. If a match is found, the firewall takes the action defined in the matching rule. If 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 allowing the packet to flow through or deny it.

a. In the Security 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
```



What chains are currently in use by R1?

R: Input, Output and Forward

c. Connections to the malicious server generate packets that must transverse the ip tables 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:

o -IFORWARD: inserts a new rule in the FORWARD chain.

o -p tcp: specifies the TCP protocol.

o -d 209.165.202.133: specifies the packet's destination

o --dport6666: specifies the destination port

o -j DROP: set the action to drop.

d. Use the iptables command again to ensure the rule was added to the FORWARD chain. The Security Workstation VM may take a few seconds to generate the output:

```
[root@secOps analyst]# iptables -L -v
```

```
"Node: R1" (as superuser)
Chain FORWARD (policy ACCEPT 0 packets, 0 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 analyst]# iptables -I FORWARD -p tcp -d 209.165.202.133 --dport 66
66 -j DROP
[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
```

```
[root@secOps analyst]# wget 209.165.202.133:6666/W32.Nimda.Amm.exe
--2025-01-03 22:29:38--  http://209.165.202.133:6666/W32.Nimda.Amm.exe
Connecting to 209.165.202.133:6666...
failed: Connection timed out.
Retrying.

--2025-01-03 22:31:49--  (try: 2)  http://209.165.202.133:6666/W32.Nimda.Amm.exe
Connecting to 209.165.202.133:6666... ^C
[root@secOps analyst]#
```

Enter `Ctrl+C` to cancel the download, if necessary.

Was the download successful this time? Explain.

R: No. The firewall is blocking connections to the malware hosting server.

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

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

Part 3: Terminate and Clear Mininet Process

a. Navigate to the terminal used to start Mininet. Terminate the Mininet by entering `quit` in the main Security Workstation 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
```

```
mininet> quit
*** Stopping 0 controllers

*** Stopping 5 terms
*** Stopping 15 links
...............
*** Stopping 3 switches
S5 S9 S10
*** Stopping 13 hosts
R1 R4 H1 H2 H3 H4 H5 H6 H7 H8 H9 H10 H11
*** Done
[analyst@secOps ~]$ sudo mn -c
[sudo] password for analyst:
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-con
trollerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-
controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
```