The University of Texas at San Antonio™

The Cyber Center for Security and Analytics

UNIVERSITY OF SOUTH CAROLINA

# ZEEK INSTRUSION DETECTION SERIES

## Lab 9: Profiling and Performance Metrics of Zeek

**Document Version: 03-13-2020**

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

# Contents

## Overview

With Zeek's event-based framework, anomalies can be detected, processed and analyzed with external software. In this lab, we explain Zeek's profiling log stream and Zeek's resource consumption.

## Objectives

By the end of this lab, students should be able to:

1. Enable Zeek's profiling log stream for session-based statistics.
2. Generate customized traffic to be captured by Zeek's profiling.
3. Implement tools necessary for testing Zeek's resource consumption.

## Lab topology

Figure 1 shows the lab topology. The topology uses 10.0.0.0/8 which is the default network assigned by Mininet. The *zeek1* and *zeek2* virtual machines will be used to generate and collect network traffic.

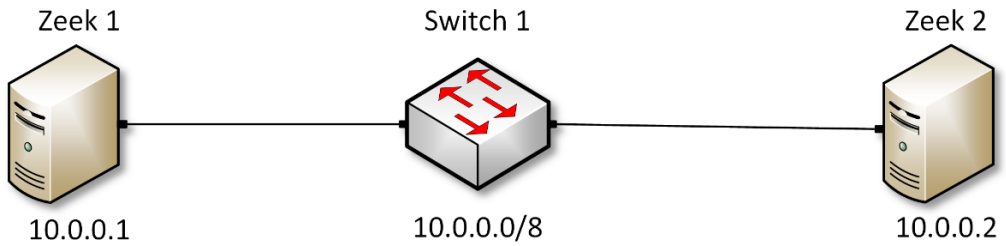



Figure 1. Lab topology.

## Lab settings

The information (case-sensitive) in the table below provides the credentials necessary to access the machines used in this lab.

Table 1. Credentials to access the Client machine

| Device | Account | Password |
|---|---|---|
| Client | admin | password |

Table 2. Shell variables and their corresponding absolute paths.

| Variable Name | Absolute Path |
|---|---|
| $ZEEK_INSTALL | /usr/local/zeek |
| $ZEEK_TESTING_TRACES | /home/zeek/zeek/testing/btest/Traces |
| $ZEEK_PROTOCOLS_SCRIPT | /home/zeek/zeek/scripts/policy/protocols |

## Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction to Zeek profiling.
2. Section 2: Generating customized malicious network traffic.
3. Section 3: Generating and viewing Zeek profiling log files.
4. Section 4: Implementing tools to test Zeek's performance.

## 1 Introduction to Zeek profiling

Zeek includes the option of enabling profiling. When profiling is enabled, a new log stream will be created to store session-related statistics. The Profile log file will contain a large variety of information, including but not limited to running time, memory usage, connection information and packet protocol statistics.

To enable profiling while using Zeek for offline packet capture file processing, you will need to implement the following functionality in a Zeek script.

```
1  module Profiling
2
3  redef profiling_file = open (fmt(<filename>, <logstream>));
4  redef profiling_interval = 3 secs;
5  redef expensive_profiling_multiple = 5;
6
7  event zeek_init() {
8      set_buf(profiling_file, F);
9  }
```

The script is explained as follows. Each number represents the respective line number:

1. Sets the module workspace as *Profiling*.
3. Specifies the name of the new profiling log file, as well as determines the format based off an input log stream.
4. Specifies the time interval for Zeek to record empirical information. In this example the time interval is 3 seconds.

5. Specifies the number of profiling intervals defined in Line 5. In this example, the profiling interval is 5 instances.
7. Initialization event.
8. Appends the new log stream information.
9. End of initialization event.

Profiling is enabled by calling the Zeek script during packet processing, as reviewed in the previous labs.

```
zeek -r <packet capture file> <Profiling Script>
```

- `<packet capture file>`: denotes the input packet capture file.
- `<Profiling Script>:` denotes the Zeek script to be run during packet processing.

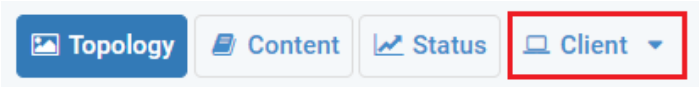
In the following section we generate customized malicious traffic to be viewed within a Zeek profiling log.

## 2 Generating customized malicious network traffic

This section introduces creating and using a Zeek profiling script, which will enable session-based statistics for Zeek packet capture file processing.

### 2.1 Starting a new instance of Zeek

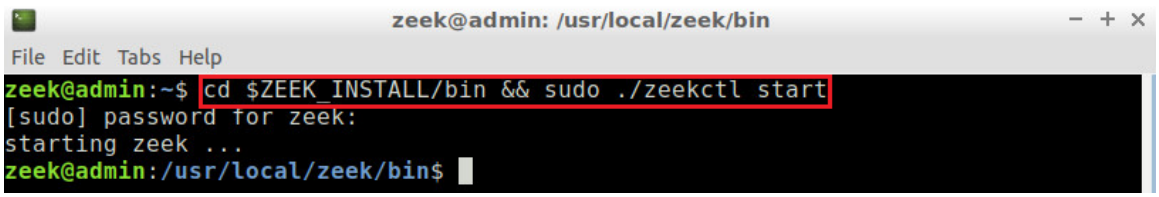
**Step 1.** From the top of the screen, click on the *Client* button as shown below to enter the *Client* machine.




**Step 2.** The *Client* machine will now open, and the desktop will be displayed. On the left side of the screen, click on the LXTerminal icon as shown below.

**Step 3.** Start Zeek by entering the following command on the terminal. This command enters Zeek's default installation directory and invokes `Zeekctl` tool to start a new instance. To type capital letters, it is recommended to hold the `Shift` key while typing rather than using the `Caps` key. When prompted for a password, type `password` and hit `Enter`.

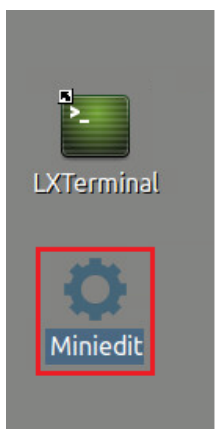
```
cd $ZEEK_INSTALL/bin && sudo ./zeekctl start
```



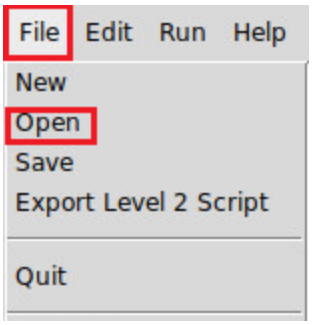

A new instance of Zeek is now active, and we are ready to proceed to the next section of the lab.

## 2.2 Launching Mininet

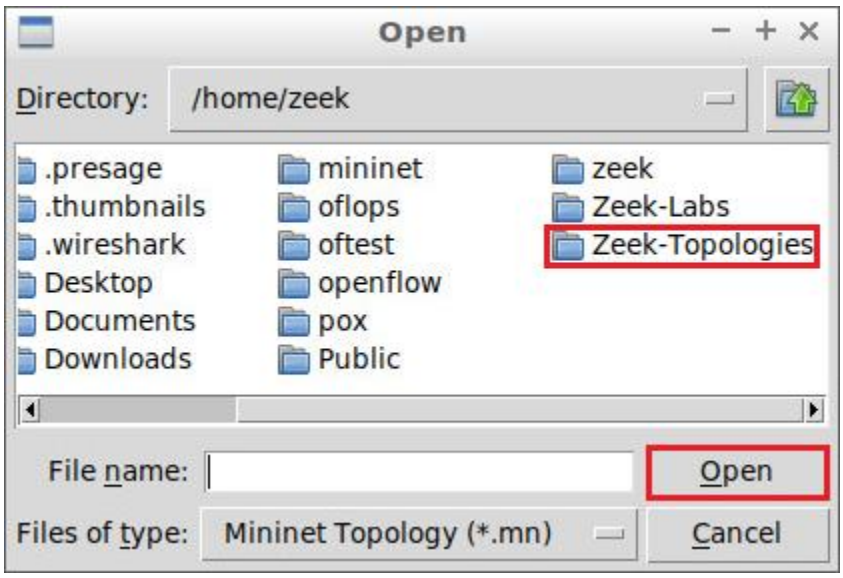
**Step 1.** From the *Client* machine's desktop, on the left side of the screen, click on the MiniEdit icon as shown below. When prompted for a password, type `password` and hit `Enter`. The MiniEdit editor will now launch.



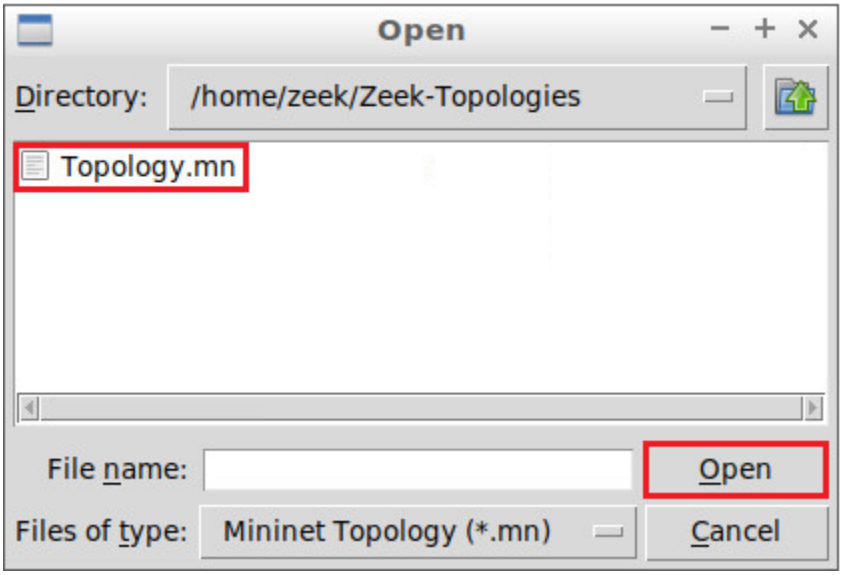

**Step 2.** The MiniEdit editor will now launch and allow for the creation of new, virtualized lab topologies. Load the correct topology by clicking the `Open` button within the `File` tab on the top left of the MiniEdit editor.



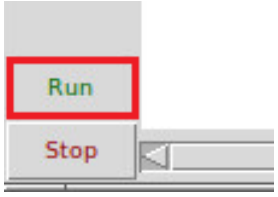

**Step 3.** Navigate to the Zeek-Topologies directory by scrolling to the right of the active directories and double clicking the Zeek-Topolgies icon, or by clicking the `Open` button.




**Step 4.** Select the *Topology.mn* file by double clicking the *Topolgies.mn* icon, or by clicking the `Open` button.

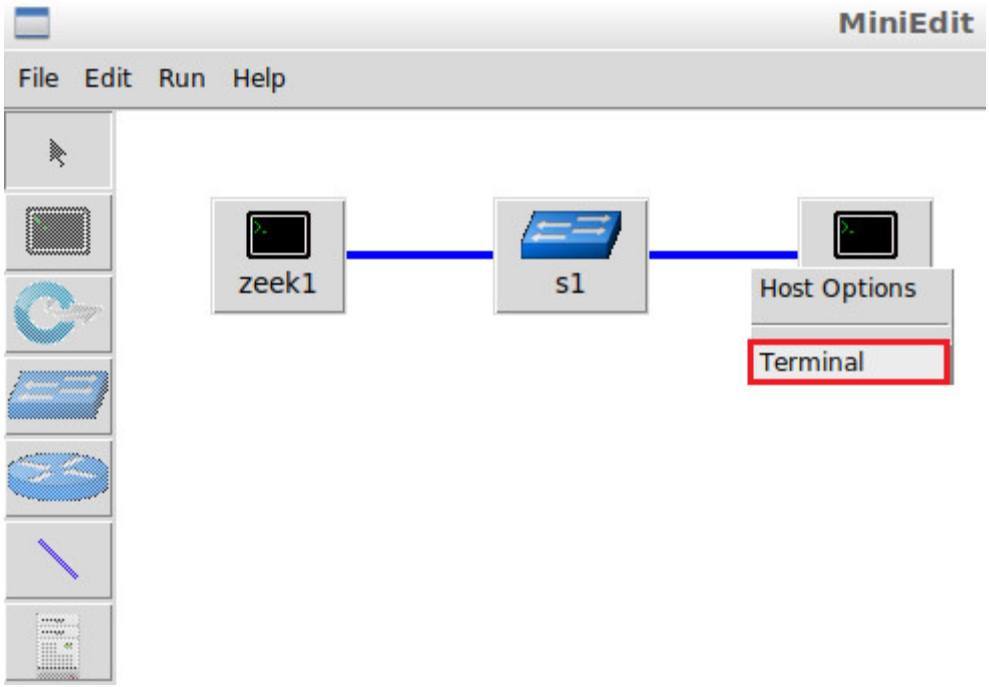


**Step 5.** To begin running the virtual machines, navigate to the `Run` button, found on the bottom left of the Miniedit editor, and select the `Run` button, as seen in the image below.




## 2.3 Setting up the zeek2 virtual machine for live network capture

**Step 1.** Launch the *zeek2* terminal by holding the right mouse button on the desired machine and clicking the `Terminal` button.

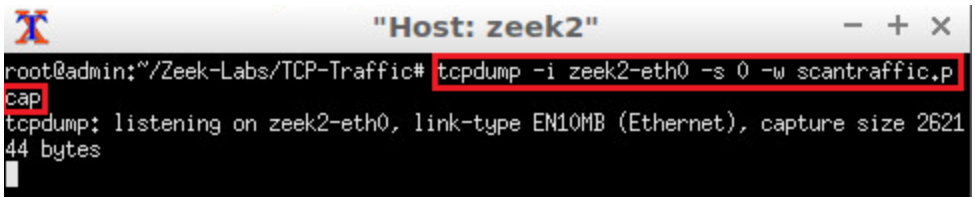



**Step 2.** Navigate to the TCP-Traffic directory.

```
cd Zeek-Labs/TCP-Traffic/
```


**Step 3.** Start live packet capture on interface *zeek2-eth0* and save the output to a file named *scantraffic.pcap*.

```
tcpdump -i zeek2-eth0 -s 0 -w scantraffic.pcap
```




The *zeek2* virtual machine is now ready to begin collecting live network traffic. Next, we will use the *zeek1* machine to generate scan-based network traffic.

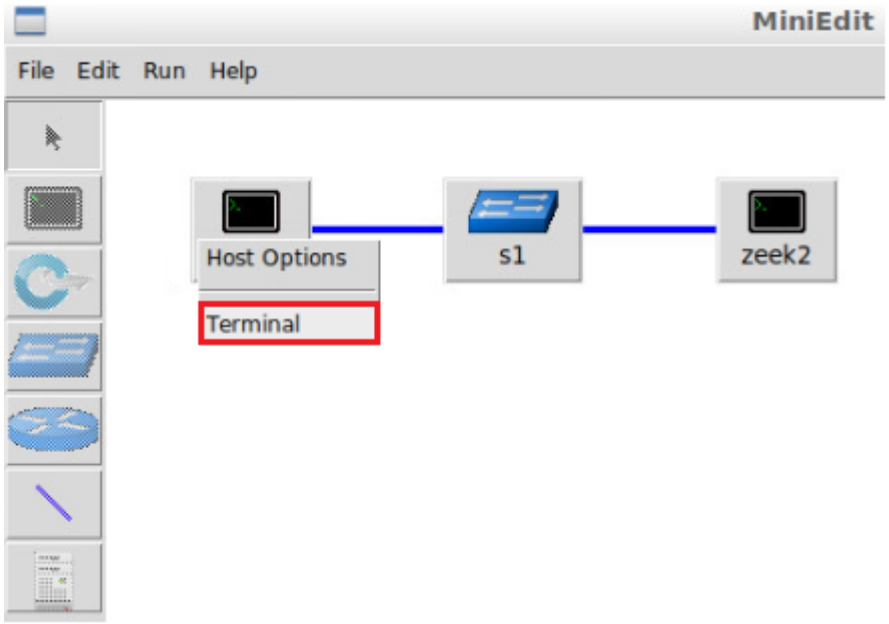
## 2.4 Using the zeek1 virtual machine for network scanning activities

In this section we use the `nmap` software to generate TCP-based scan traffic.

This section introduces two new options for the `nmap` software.

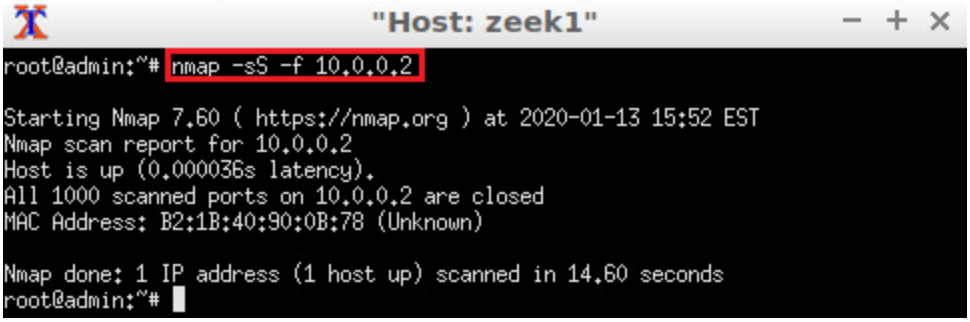
- `-f`: specifies to send packet fragments. By fragmenting packets, a scanner can attempt to bypass firewalls that check for entire packet signatures.
- `-mtu <num>`: specifies the max number of bytes to be sent in a fragmented packet. The number variable must be a multiple of 8.

**Step 1.** Minimize the *zeek2* `Terminal` and open the *zeek1* `Terminal` by following the previous steps. If necessary, right click within the Miniedit editor to activate your cursor.



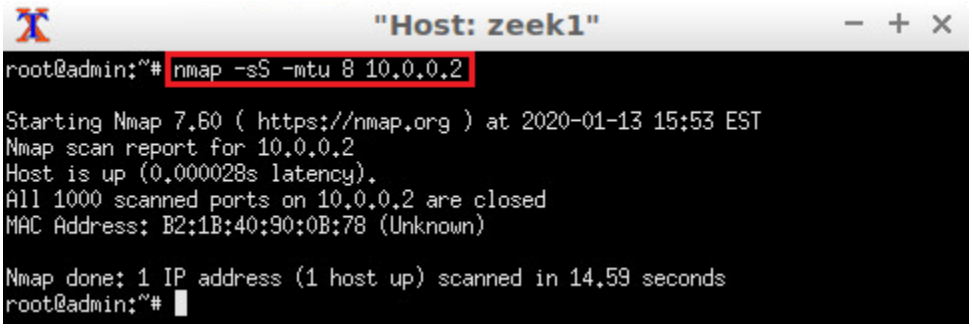

**Step 2.** Launch a fragmented TCP SYN scan against the *zeek2* machine.

```
nmap -sS -f 10.0.0.2
```



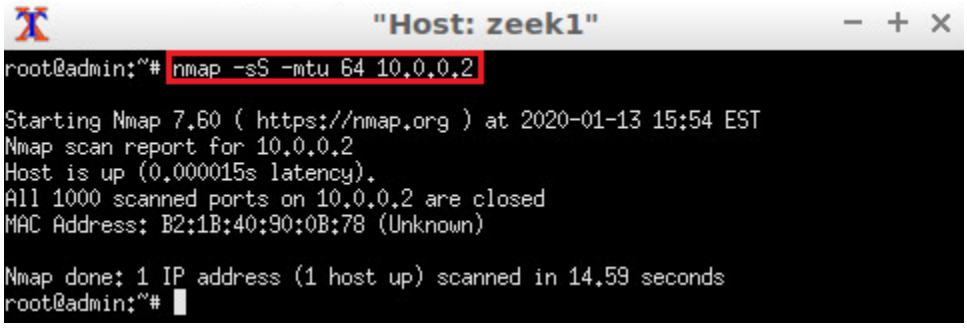

**Step 3.** Launch a fragmented TCP SYN scan with a packet size of 8 bytes against the *zeek2* machine.

```
nmap -sS -mtu 8 10.0.0.2
```




**Step 4.** Launch a fragmented TCP SYN scan with a packet size of 64 bytes against the *zeek2* machine.

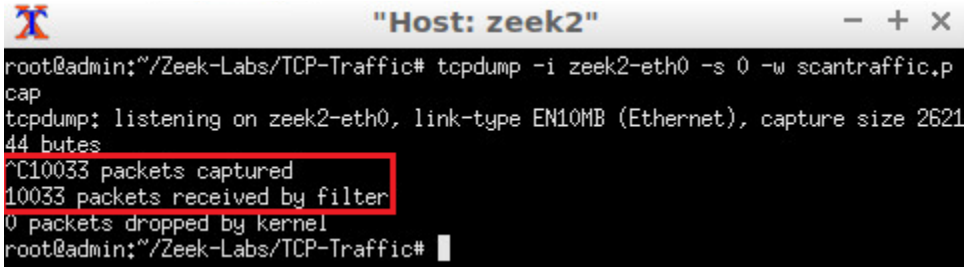
```
nmap -sS -mtu 64 10.0.0.2
```



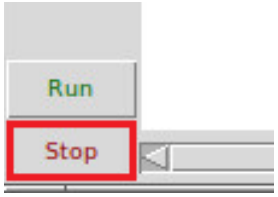

### 2.4.1 Terminating live network capture

**Step 1.** Minimize the *zeek1 Terminal* and open the *zeek2 Terminal* using the navigation bar at the bottom of the screen. If necessary, right click within the Miniedit editor to activate your cursor.

**Step 2**. Use the `Ctrl+c` key combination to stop live traffic capture. Statistics of the capture session will we be displayed. 10,033 packets were recorded by the interface, which were then captured and stored in the new *scantraffic.pcap* file.




**Step 3.** Stop the current Mininet session by clicking the `Stop` button on the bottom left of the MiniEdit editor, and close the MiniEdit editor by clicking the `x` on the top right of the editor.




## 3 Generating and viewing Zeek profiling log files

Now that we have collected fragmented traffic, we can begin processing the packet capture file with Zeek.

### 3.1 Applying the profiling filter

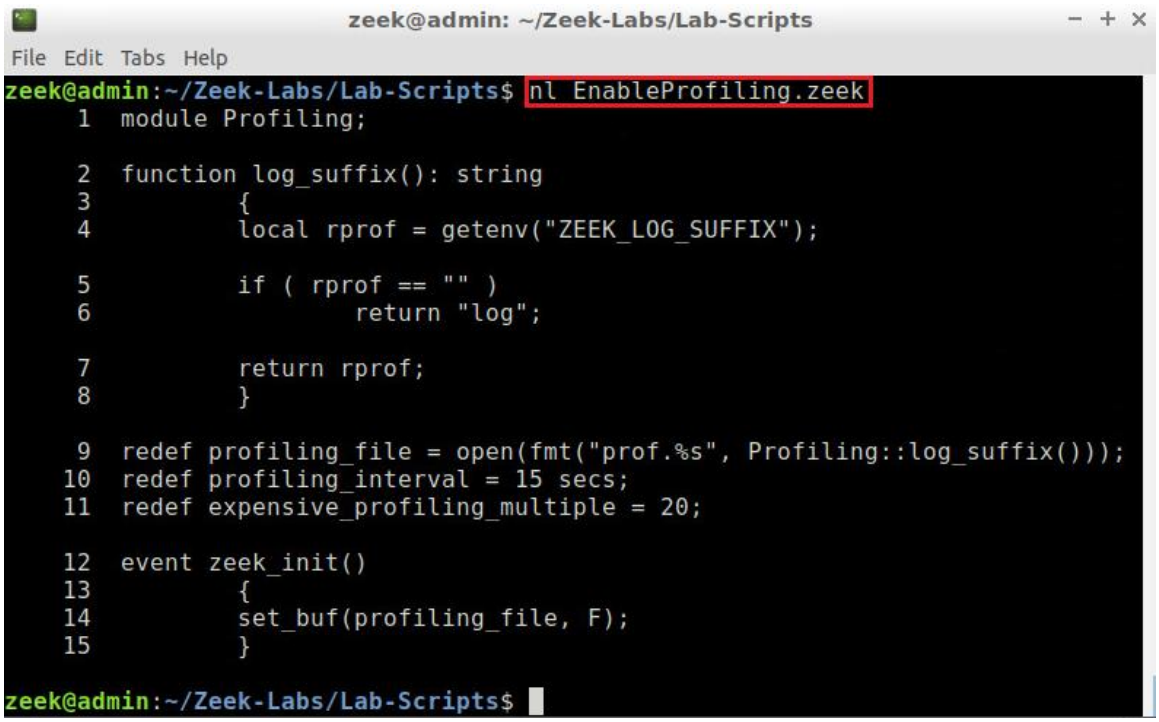
**Step 1.** On the left side of the *Client* desktop, click on the LXTerminal icon as shown below.


**Step 2.** Navigate to the *Lab-Scripts* directory.

```
cd Zeek-Labs/Lab-Scripts/
```


**Step 3.** View the *EnableProfiling.zeek* Zeek script.

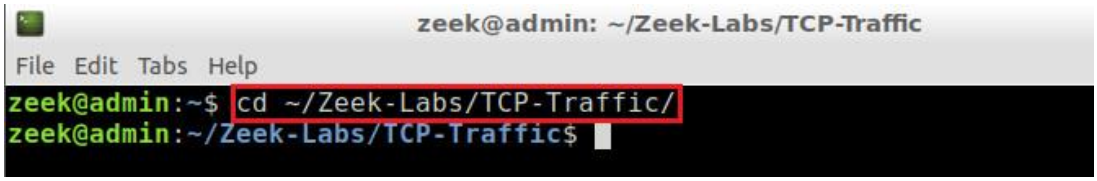
```
nl EnableProfiling.zeek
```




Similar to the example in the introduction, the *EnableProfiling.zeek* Zeek script is used to create a new log file named *Statistics.log* containing Zeek profiling statistics. The script enables the intervals to be 15 seconds apart, with 20 total intervals.

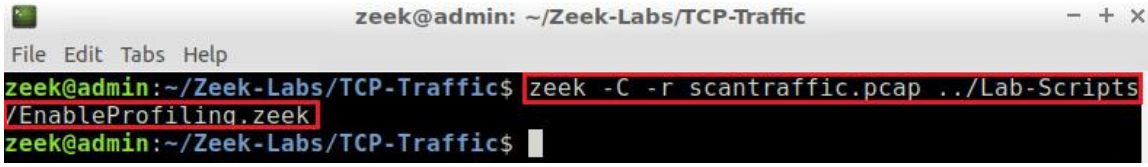
**Step 4.** Navigate to the TCP-Traffic directory.

```
cd ~/Zeek-Labs/TCP-Traffic/
```



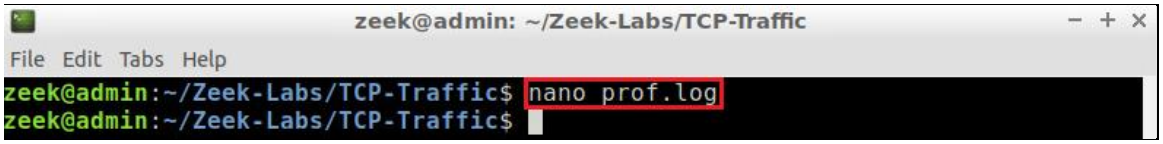

**Step 5.** Process the *ntraffic.pcap* packet capture file.

```
zeek -C -r scantraffic.pcap ../Lab-Scripts/EnableProfiling.zeek
```

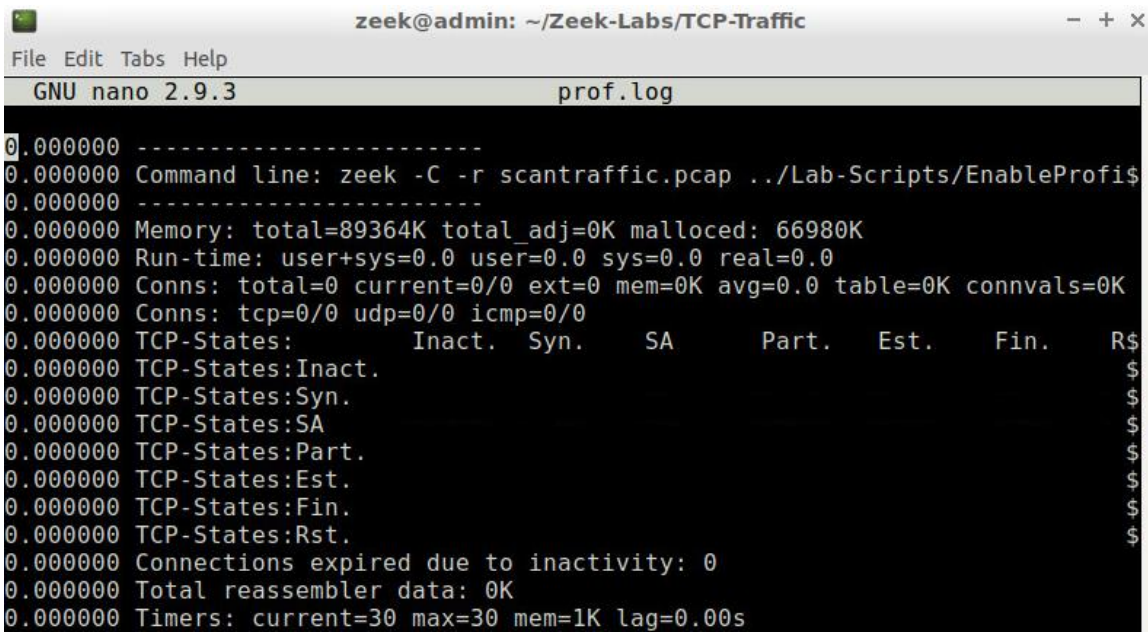



**Step 6.** Display the contents of the *prof.log* file.

```
nano prof.log
```




The *prof.log* file will be displayed.




Viewing the *Statistics.log* file, each profiling_interval will be displayed between a line separator made by dashes `---`.

Within the *prof.log* file, we can see the total memory used while processing the packet capture file, the Run-time, as well as a number of TCP flags, connections and *Triggers*. Within the first iteration of profiling_interval we see that no TCP packet flags have been recorded.

**Step 7.** Go to the next iteration of profiling_interval within the *Statistics.log* file.

```
zeek@admin: ~/Zeek-Labs/TCP-Traffic
File  Edit  Tabs  Help
  GNU nano 2.9.3                    prof.log

1578948752.164306 Memory: total=89364K total_adj=0K malloced: 66984K
1578948752.164306 Run-time: user+sys=0.0 user=0.0 sys=0.0 real=0.0
1578948752.164306 Conns: total=0 current=0/0 ext=0 mem=0K avg=0.0 table=0K co$
1578948752.164306 Conns: tcp=0/0 udp=0/0 icmp=0/0
1578948752.164306 TCP-States:        Inact.  Syn.    SA      Part.   Est.    $
1578948752.164306 TCP-States:Inact.                                          $
1578948752.164306 TCP-States:Syn.                                            $
1578948752.164306 TCP-States:SA                                              $
1578948752.164306 TCP-States:Part.                                           $
1578948752.164306 TCP-States:Est.                                            $
1578948752.164306 TCP-States:Fin.                                            $
1578948752.164306 TCP-States:Rst.                                            $
1578948752.164306 Connections expired due to inactivity: 0
1578948752.164306 Total reassembler data: 0K
1578948752.164306 Timers: current=27 max=30 mem=1K lag=1578948751.16s
1578948752.164306 DNS_Mgr: requests=0 succesful=0 failed=0 pending=0 cached_h$
1578948752.164306 Triggers: total=0 pending=0
1578948752.164306         ScheduleTimer = 2

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell
```

By scrolling through the *prof.log* file, we can see information found in the next iteration of a profiling_interval. We can see the total number of *TCP-States:Syn* has updated multiple parameters, with additional *Triggers* being been included. This includes the total memory usage, displayed towards the bottom of the image.

Zeek profiling is a great tool for generating more detailed session-based statistics while processing packet capture files with Zeek.

**Step 8.** Press `Ctrl + x` to exit out the nano.

## 4 Implementing tools to test Zeek's performance

While Zeek profiling will display the resulting statistics after processing a packet capture file, it is important to monitor Zeek resource consumption during network traffic analysis.

A number of Linux-based software utilities can be used to track system resource consumption in real time.

### 4.1 Using sysstat sar utility

The `sar` command can be used to display a number of system resources over specific time intervals. The following steps will highlight the ways to enable sar resource tracking.

**Step 1.** Launch the `sar` utility to track CPU consumption.

```
sar 2 30
```

- `sar`: calls the `sar` utility, belonging to the `syystat` packages.
- `2`: indicates each iteration of CPU statistics is separated by a 2 second time interval.
- `30`: indicates that a total of 30 iterations of CPU statistics should be displayed.

```
zeek@admin: ~/Zeek-Labs/TCP-Traffic
File  Edit  Tabs  Help
zeek@admin:~/Zeek-Labs/TCP-Traffic$ sar 2 30
Linux 4.15.0-70-generic (admin)         01/13/2020      _x86_64_        (8 CPU
)

04:08:37 PM     CPU     %user     %nice   %system   %iowait    %steal     %idl
e
04:08:39 PM     all      0.13      0.00      0.06      0.00      0.00     99.8
1
04:08:41 PM     all      0.00      0.00      0.00      0.00      0.00    100.0
0
04:08:43 PM     all      0.06      0.00      0.06      0.00      0.00     99.8
8
04:08:45 PM     all      0.06      0.00      0.00      0.00      0.00     99.9
4
04:08:47 PM     all      0.00      0.00      0.00      0.00      0.00    100.0
0
```

Use the `CTRL + C` keyboard combination to terminate the `sar` utility and return to the terminal.

**Step 2.** Launch the `sar` utility to track memory consumption.

```
sar -r 3 25
```

- `sar`: calls the `sar` utility, belonging to the `syystat` packages.
- `-r`: indicates memory consumption in kilobytes.
- `3`: indicates each iteration of memory statistics is seperated by a 3 second time interval.
- `25`: indicates that a total of 25 iterations of memory statistics should be displayed.

```
zeek@admin: ~/Zeek-Labs/TCP-Traffic
File  Edit  Tabs  Help
zeek@admin:~/Zeek-Labs/TCP-Traffic$ sar -r 3 25
Linux 4.15.0-70-generic (admin)         01/13/2020      _x86_64_        (8 CPU
)

04:09:04 PM kbmemfree   kbavail kbmemused  %memused kbbuffers  kbcached  kbcom
mit   %commit  kbactive   kbinact   kbdirty
04:09:07 PM   7266672   7496160    792116      9.83     92692    332116   1172
688     12.99    432692    121120        48
04:09:10 PM   7266672   7496160    792116      9.83     92692    332116   1172
688     12.99    432692    121120        48
04:09:13 PM   7266672   7496160    792116      9.83     92692    332116   1172
688     12.99    432692    121120         0
```

Use the `CTRL + C` keyboard combination to terminate the `sar` utility and return to the terminal.

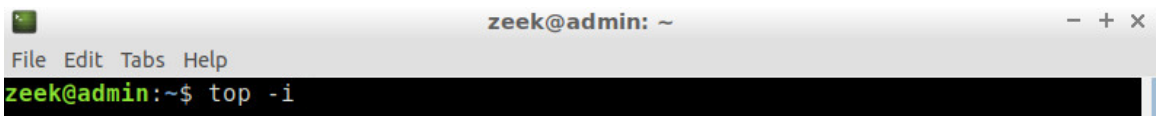
## 4.2 Using the top utility

Alternative to the syystat sar utility, the top utility can be used to display the resource consumption of every active process.

**Step 1.** Launch the top utility to track resource consumption.

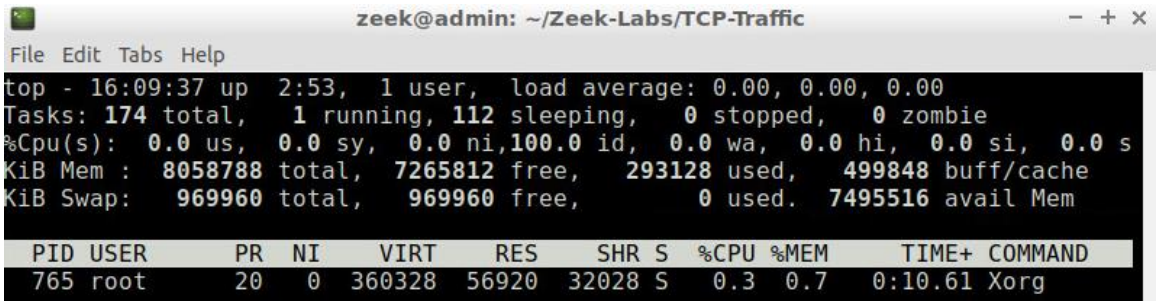
```
top -i
```

- `top`: calls the `top` utility.
- `-i`: toggles idle processes off, so that only active processes will be displayed.



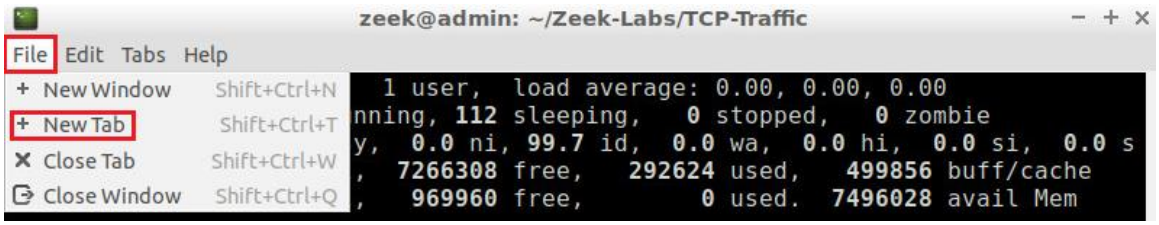

After entering the command, the Terminal will display the resource consumption.

Each row will belong to a unique process and display the related CPU and memory resource usage.



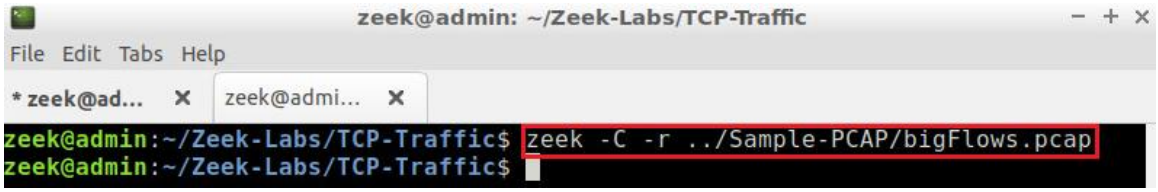

## 4.3 Viewing the resource consumption of Zeek

**Step 1.** Using the File drop down options, create a *New Tab* within the Terminal.

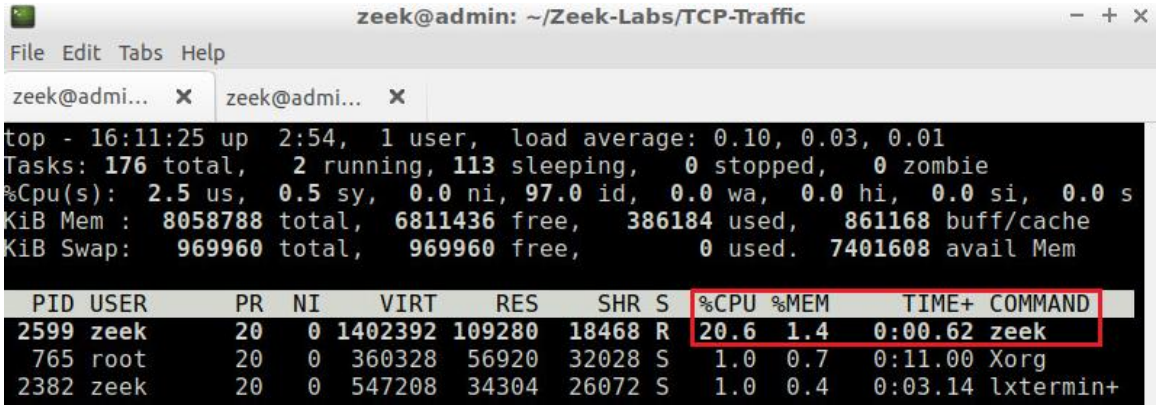



**Step 2.** In the second tab, begin packet capture file processing of the *bigFlows.pcap* file using Zeek.

```
zeek -C -r ../Sample-PCAP/bigFlows.pcap
```




**Step 3.** Return to the first Terminal tab and view the active processes.



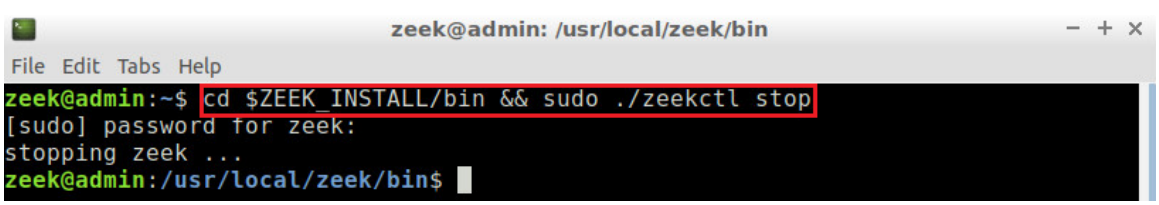

Use the `CTRL + C` keyboard combination to terminate the `top` utility and return to the terminal.

## 4.4 Closing the current instance of Zeek

After you have finished the lab, it is necessary to terminate the currently active instance of Zeek. Shutting down a computer while an active instance persists will cause Zeek to shut down improperly and may cause errors in future instances.

**Step 1.** Stop Zeek by entering the following command on the terminal. If required, type `password` as the password. If the Terminal session has not been terminated or closed, you may not be prompted to enter a password. To type capital letters, it is recommended to hold the *Shift* key while typing rather than using the *Caps* key.

```
cd $ZEEK_INSTALL/bin && sudo ./zeekctl stop
```




Concluding this lab, we introduced Zeek's profiling capabilities and generated fragmented traffic to be processed into a profiling log file. Lastly, we introduced Terminal utilities that

can be used to track Zeek's resource consumption per process. Regular checking of Zeek profiling and resource consumption is necessary to ensure the IDS is working optimally in a real-time environment.

Furthermore, we have concluded introducing Zeek's capabilities as an IDS. The remaining labs within this series will focus on further processing Zeek log files for advanced analysis.

## References

1. "Profiling", Zeek user manual, [Online], Available: https://docs.zeek.org/en/stable/scripts/policy/misc/profiling.zeek.html