The Cyber Center for Security and Analytics

ZEEK INTRUSION DETECTION SERIES

Lab 1: Introduction to the Capabilities of Zeek

Document Version: 03-13-2020

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

Contents

Overview

This lab introduces Zeek, an open-source network analysis framework primarily used in security monitoring and traffic analysis. The primary focus of this lab is to explain Zeek's layered architecture while demonstrating Zeek's capabilities towards performing network traffic analysis.

Objectives

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

1. Understand Zeek's layered architecture.
2. Start and manage a Zeek instance using the *ZeekControl* utility.
3. Use Zeek to process packet captures files.
4. Generate and analyze live network traffic in Zeek.

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.

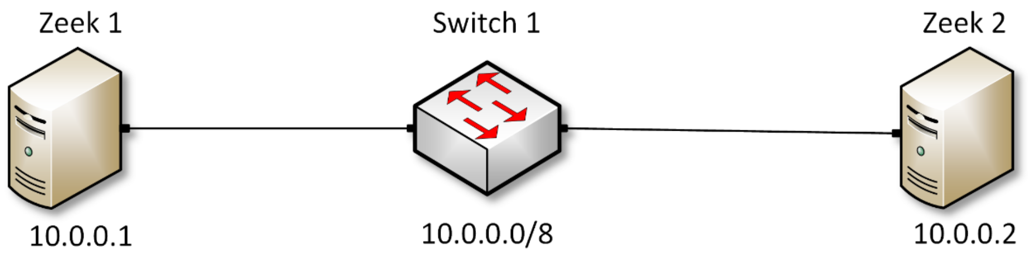


Figure 1. Lab topology.

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.
2. Section 2: Using *ZeekControl* to update the status of Zeek.
3. Section 3: Introduction to Zeek's traffic analysis capabilities.

1 Introduction to Zeek

Zeek is a passive, open-source network traffic analyzer. It is primarily used as a security monitor that inspects all traffic on a network link for signs of suspicious activity[1]. It can run on commodity hardware with standard UNIX-based systems and can be used as a passive network monitoring tool.

Setting Zeek as a node with an assigned IP address on the monitored network is not mandatory. Figure 2 shows Zeek's layered architecture. Once Zeek receives packets, its *event engine* converts them into *events*. The *events* are then forwarded to the policy script interpreter, which generates logs, notifications, and/or actions.

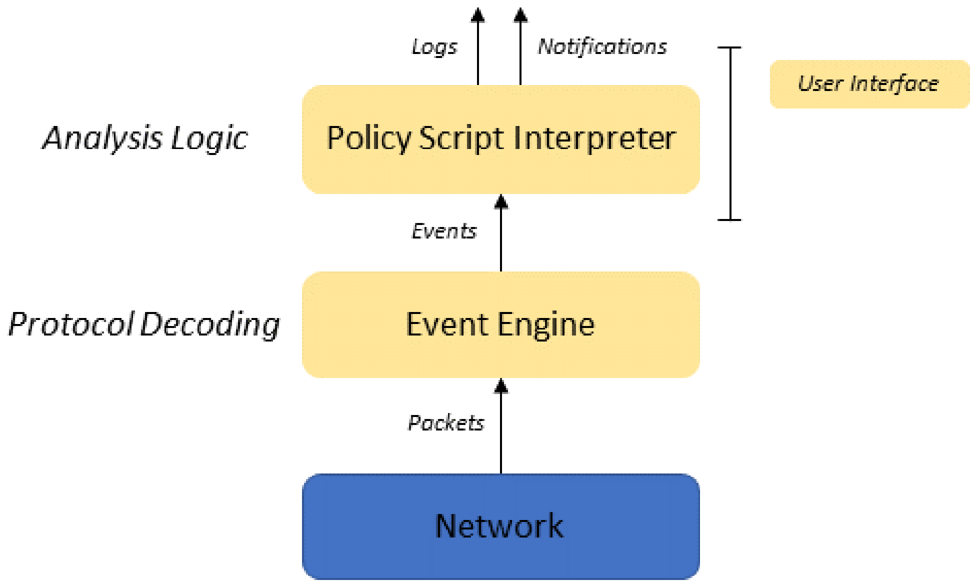


Figure 2. Zeek's architecture.

Zeek uses the standard `libpcap` library for capturing packets to be used in network monitoring and analysis.

1.1 The Zeek event engine

The event engine layer performs low-level network packets analysis. It receives raw packets from the network layer (packet capture), sorts them by connection, reassembles data streams, and decodes application layer protocols. Whenever it encounters something potentially relevant to the policy layer, it generates an event.

The event engine consists of several analyzers responsible for well-defined tasks. Typical tasks include decoding a specific protocol, performing signature-matching, identifying backdoors, etc. Usually, an analyzer is accompanied by a default script which implements some general policy adjustable to the local environment. The event engine can be divided into four major parts.

1.1.1 State management

Zeek's main data structure is a connection which follows typical flow identification mechanisms, such as 5-tuple approaches. The 5-tuple structure consists of the source IP address/port number, destination IP address/port number, and the protocol in use. For a connection-oriented protocol like TCP, the definition of a connection is more clear-cut, however for others such as UDP and ICMP, Zeek implements a flow-like abstraction to aggregate packets. Each packet belongs to exactly one connection.

1.1.2 Transport layer analyzers

On the transport layer, Zeek analyzes TCP, UDP packets. In TCP, Zeek's associated analyzer closely follows the various state changes, keeps track of acknowledgments, handles retransmissions and much more.

1.1.3 Application layer analyzers

The analysis of the application layer data of a connection depends on the service. There are analyzers for a wide variety of different protocols, e.g. HTTP, SMTP or DNS, that generally conduct detailed analysis of the data stream.

1.1.4 Infrastructure

The general infrastructure of Zeek includes the event and timer management components, the script interpreter, and data structures.

1.2 The Zeek policy script interpreter

While the event engine itself is policy-neutral, the top layer of Zeek defines the environment-specific network security policy. By writing handlers for events that may be raised by the event engine, the user can precisely define the constraints within the given network. If a security breach is detected, the policy layer generates an alert.

New event handlers can be created in Zeek's own scripting language. While providing all expected convenience of a powerful scripting language, it has been designed with network intrusion detection in mind. While it is expected that additional policy scripts are written by the user, there are nevertheless several default scripts included with the initial installation of Zeek. These default scripts already perform a wide range of analyses and are easily customizable.

1.3 Zeek analyzers

The majority of Zeek's analyzers are in its event engine with accompanying policy scripts that can be customized by the user. Sometimes, however, the analyzer is just a policy script implementing multiple event handlers. The analyzers perform application layer decoding, anomaly detection, signature matching and connection analysis. Zeek has been designed so that it is easy to add additional analyzers.

1.4 Signatures

Most network intrusion detection systems (NIDS) match a large set of signatures against the network traffic. Here, a signature is a pattern of bytes that the NIDS tries to locate in the payload of network packets. As soon as a match is found, the system generates an alert.

A well-known IDS system is *Snort*; conversely, Zeek's general approach to intrusion detection has a much broader scope than traditional signature-matching, yet still contains a signature engine providing a functionality that is similar to that of other systems. Furthermore, while Zeek implements its own flexible signature language, there exists a converter which directly translates Snort's signatures into Zeek's syntax, as shown below:

```
alert tcp any any -> [a.b.0.0/16,c.d.e.0/24] 80
  ( msg:"WEB-ATTACKS conf/httpd.conf attempt";
    nocase; sid:1373; flow:to_server,established;
    content:"conf/httpd.conf"; [...] )
```

(a) Snort

```
signature sid-1373 {
  ip-proto == tcp
  dst-ip == a.b.0.0/16,c.d.e.0/24
  dst-port == 80
  # The payload below is actually generated in a
  # case-insensitive format, which we omit here
  # for clarity.
  payload /.*conf\/httpd\.conf/
  tcp-state established,originator
  event "WEB-ATTACKS conf/httpd.conf attempt"
}%
```

(b) Zeek

Figure 3. Example of signature conversion[1]. (a) Snort's signature. (b) Zeek's signature.

1.5 ZeekControl

ZeekControl, formerly known as *BroControl*, is an interactive shell for easily operating and managing Zeek installations on a single system or across multiple systems in a traffic-monitoring cluster.

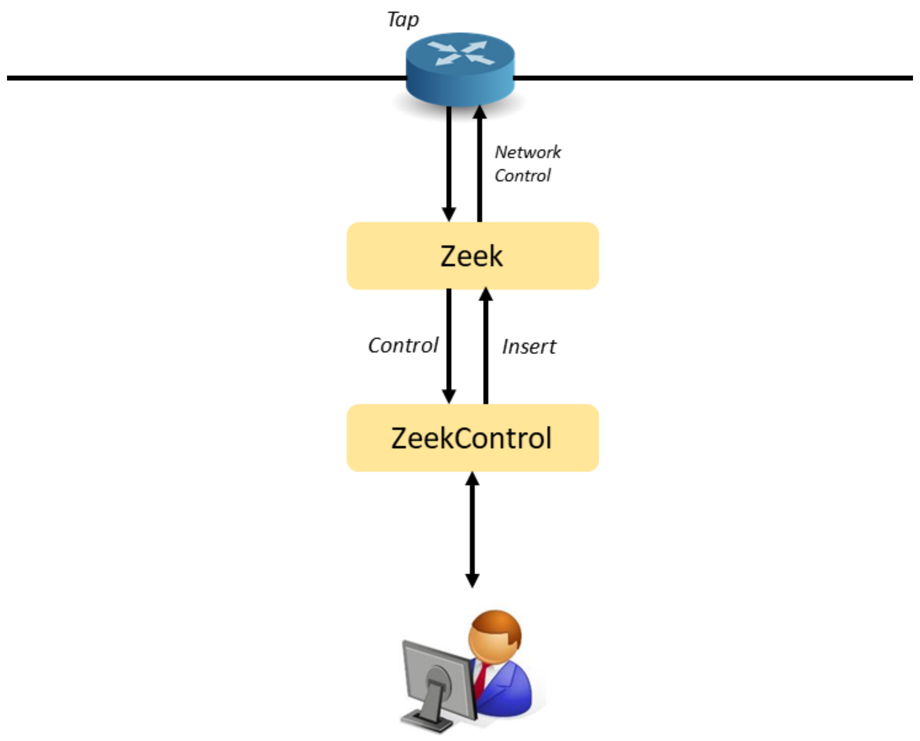
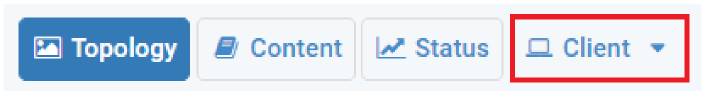


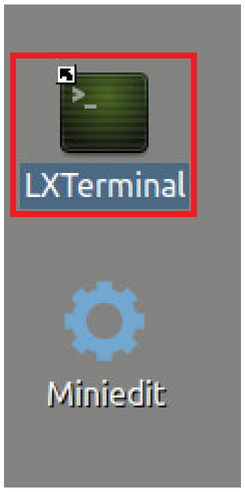
Figure 4. *ZeekControl* scheme.

2 Using ZeekControl to update the status of Zeek

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

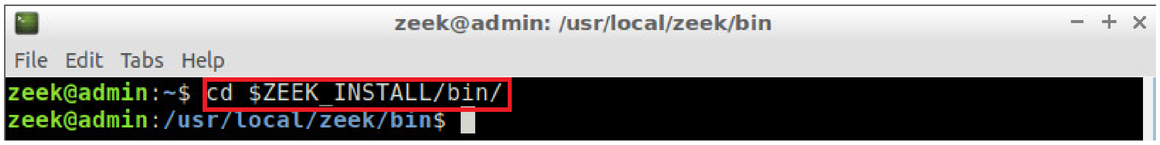


Step 2. The *Client* machine will now open, and the desktop will be displayed. On the left side of the screen, double click on the LXTerminal icon as shown below.



Step 3. Using the Terminal, input the following command to enter the *ZeekControl* directory. To type capital letters, it is recommended to hold the *Shift* key while typing rather than using the *Caps* key.

```
cd $ZEEK_INSTALL/bin/
```

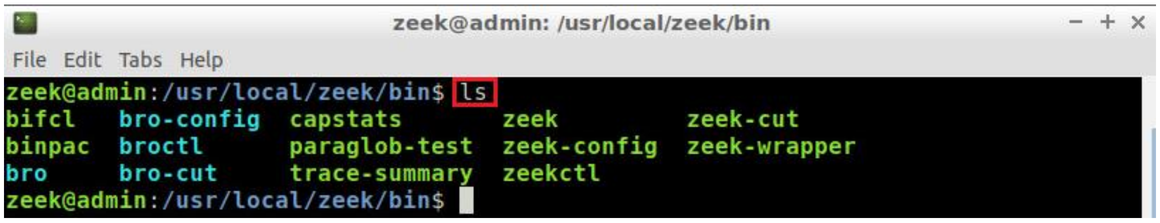


```
zeek@admin:~$ cd $ZEEK_INSTALL/bin/
zeek@admin:/usr/local/zeek/bin$
```

The active directory will change, as seen on the second line of the Terminal. Note that $ZEEK_INSTALL variable was substituted by its value (*/usr/local/zeek*) listed in Table 2.

Step 4. Use the following command to view the contents of the active directory.

```
ls
```

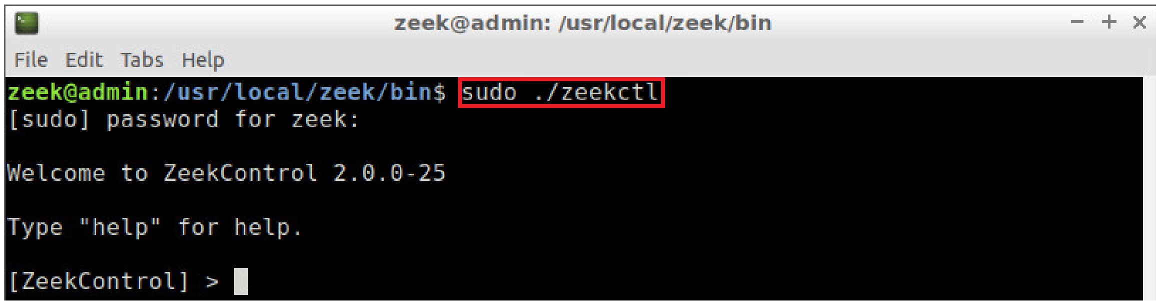


```
zeek@admin:/usr/local/zeek/bin$ ls
bifcl    bro-config  capstats       zeek         zeek-cut
binpac   broctl      paraglob-test  zeek-config  zeek-wrapper
bro      bro-cut     trace-summary  zeekctl
zeek@admin:/usr/local/zeek/bin$
```

The directory contents will be displayed. The green file name portrays an executable file.

Step 5. Use the following command to launch the *ZeekControl* tool. When prompted for a password, type `password` and hit *Enter*.

```
sudo ./zeekctl
```



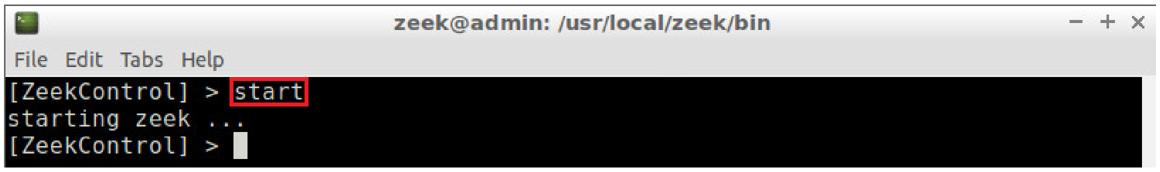
```
zeek@admin:/usr/local/zeek/bin$ sudo ./zeekctl
[sudo] password for zeek:

Welcome to ZeekControl 2.0.0-25

Type "help" for help.

[ZeekControl] >
```

Once active, the *ZeekControl* prompt will be displayed within the Terminal. The `help` command will display additional information regarding *ZeekControl*.

2.1 Starting a new instance of Zeek

Step 1. To initialize Zeek, enter the following command into the *ZeekControl* prompt.

```
start
```

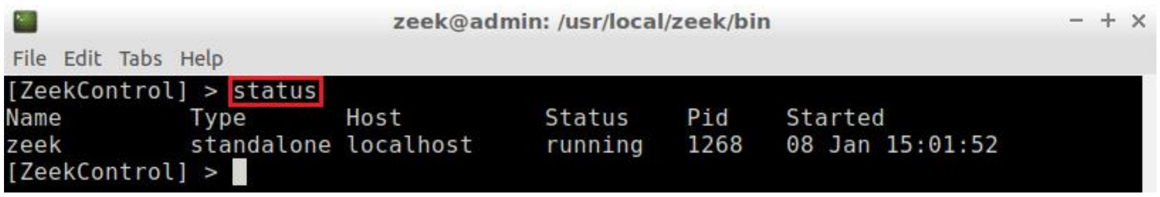


```
[ZeekControl] > start
starting zeek ...
[ZeekControl] >
```

If you see error messages during the new Zeek instance initializing process, please ignore it.

Step 2. Use the following command to view the status of the currently active Zeek instance to ensure that it is active.

```
status
```



```
[ZeekControl] > status
Name         Type       Host          Status    Pid    Started
zeek         standalone localhost     running   1268   08 Jan 15:01:52
[ZeekControl] >
```

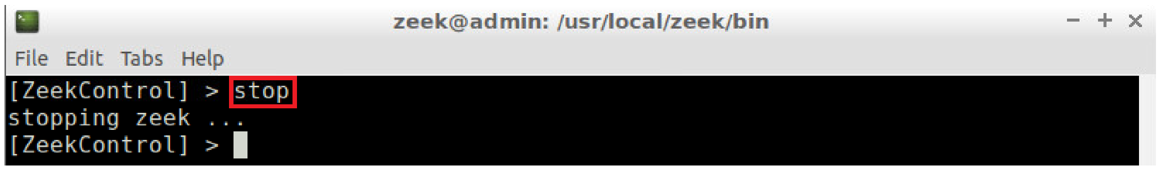
The ***running*** status indicates that Zeek is currently active and functioning properly. The output of the `status` command includes other useful parameters:

- `Name`: the name of the Zeek instance.
- `Type`: the type of the instance (standalone in our case).
- `Host`: the hostname (localhost).
- `Pid`: the process ID. This ID can be used with other tools like `kill` to send a signal to the process.
- `Started`: the starting date and time of the instance.

2.2 Stopping the active instance of Zeek

Step 1. To stop Zeek, enter the following command into the *ZeekControl* prompt.

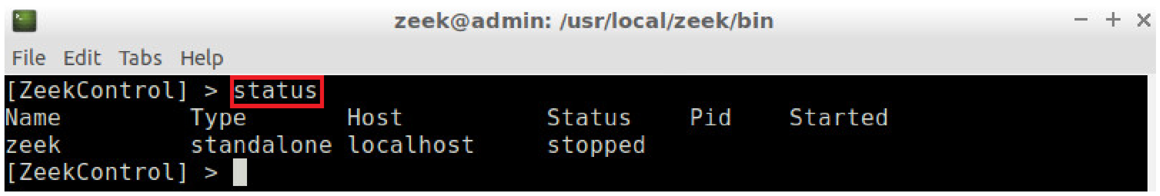
```
stop
```



```
[ZeekControl] > stop
stopping zeek ...
[ZeekControl] >
```

Step 2. Use the following command to verify the exit status of Zeek.

```
status
```

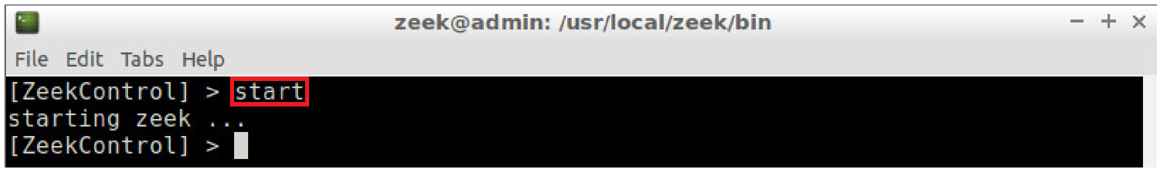


```
[ZeekControl] > status
Name          Type       Host          Status    Pid    Started
zeek          standalone localhost     stopped
[ZeekControl] >
```

The *stopped* status indicates that Zeek is currently stopped.

Step 3. To restart Zeek, enter the following command into the *ZeekControl* prompt.

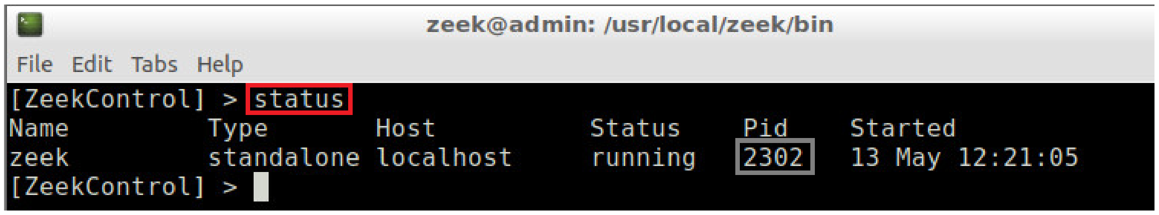
```
start
```



```
[ZeekControl] > start
starting zeek ...
[ZeekControl] >
```

Step 4. Type the following command to check the Zeek restarted. You will verify that Zeek is running.

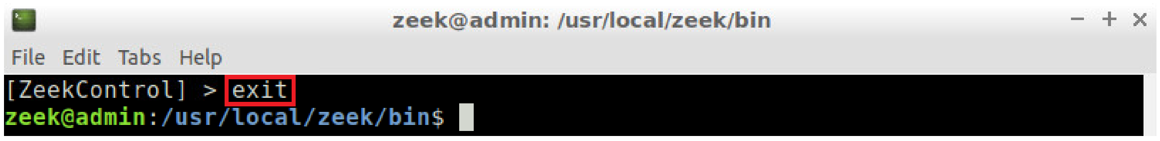
```
status
```



```
[ZeekControl] > status
Name         Type       Host          Status    Pid    Started
zeek         standalone localhost     running   2302   13 May 12:21:05
[ZeekControl] >
```

Step 5. To exit from *ZeekControl* type following command.

```
exit
```



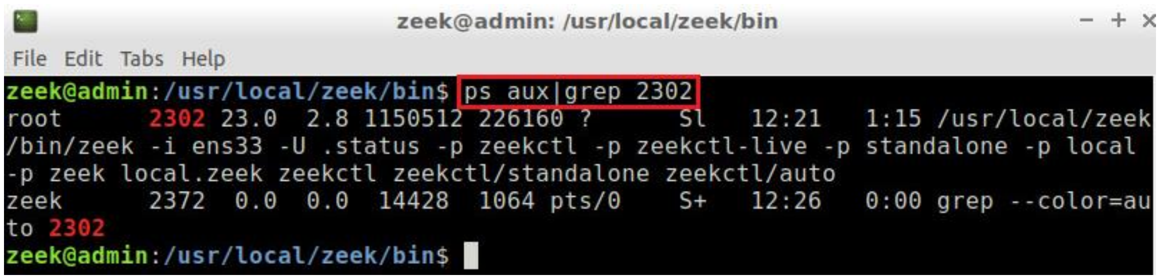
```
[ZeekControl] > exit
zeek@admin:/usr/local/zeek/bin$
```

Note that exiting the *ZeekControl* tool does not stop Zeek. Zeek is only stopped by explicitly using the `stop` command in the ZeekControl prompt.

Step 6. To verify that zeek control is not stopped type the following command:

```
ps aux | grep <PID_number>
```

where *<PID_number>* is the number inside the grey box depicted in step 4.



```
zeek@admin:/usr/local/zeek/bin$ ps aux|grep 2302
root      2302 23.0  2.8 1150512 226160 ?      Sl   12:21   1:15 /usr/local/zeek
/bin/zeek -i ens33 -U .status -p zeekctl -p zeekctl-live -p standalone -p local
-p zeek local.zeek zeekctl zeekctl/standalone zeekctl/auto
zeek      2372  0.0  0.0  14428  1064 pts/0    S+   12:26   0:00 grep --color=au
to 2302
zeek@admin:/usr/local/zeek/bin$
```

Notice that the <PID_number> may differ than the figure above.

3 Introduction to Zeek's traffic analysis capabilities

Zeek's broad range of traffic analysis capabilities makes it an exceptional intrusion detection system (IDS) and network analysis framework. Zeek is proficient in processing packet capture (pcap) files and logging traffic on a given network interface.

3.1 Processing offline packet capture files

Linux-based systems process packet capture (pcap) files using the `libpcap` library. In Zeek, it is possible to capture live traffic and analyze trace files. In the following example, we analyze a pcap file using a premade script that detects brute force attacks.

3.1.1 Command format for processing packet capture files

The general format for initializing offline packet capture analysis is as follows:

```
zeek -r <pcap_file_location> <script_location>
```

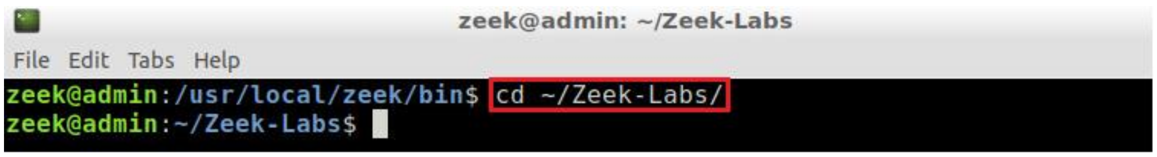
- `zeek`: command to invoke Zeek.
- `-r`: option signifies to Zeek that it will be reading from an offline file.
- `<pcap_file_location>`: indicates the pcap file location.
- `<script_location>`: indicates the script location.

3.1.2 Leveraging a script to detect brute force attacks present in a pcap file

Zeek installs a number of default scripts and trace files that can be used for testing purposes. In this section, we use the *bruteforce.pcap* as the input packet capture file and *ZeekBruteforceDetection.zeek* as the detection script. The packet capture file contains network traffic of a brute force password attack, while the script defines the brute forcing event for the Zeek event engine.

Step 1. Enter the lab workspace directory. To type capital letters, it is recommended to hold the `Shift` key while typing rather than using the `Caps` key.

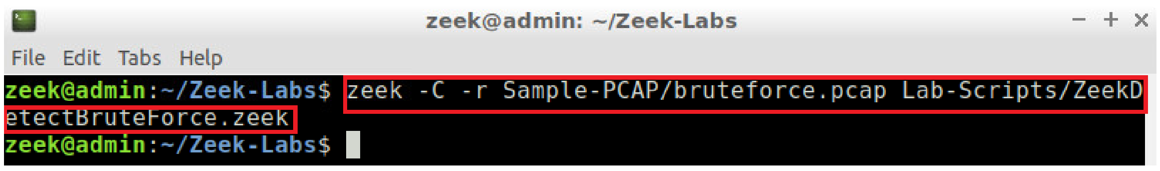
```
cd ~/Zeek-Labs/
```



```
zeek@admin:/usr/local/zeek/bin$ cd ~/Zeek-Labs/
zeek@admin:~/Zeek-Labs$
```

Step 2. Initialize Zeek offline packet parsing on the packet capture file. It is possible to use the `tab` key to autocomplete the longer paths.

```
zeek -C -r Sample-PCAP/bruteforce.pcap Lab-Scripts/ZeekDetectBruteForce.zeek
```

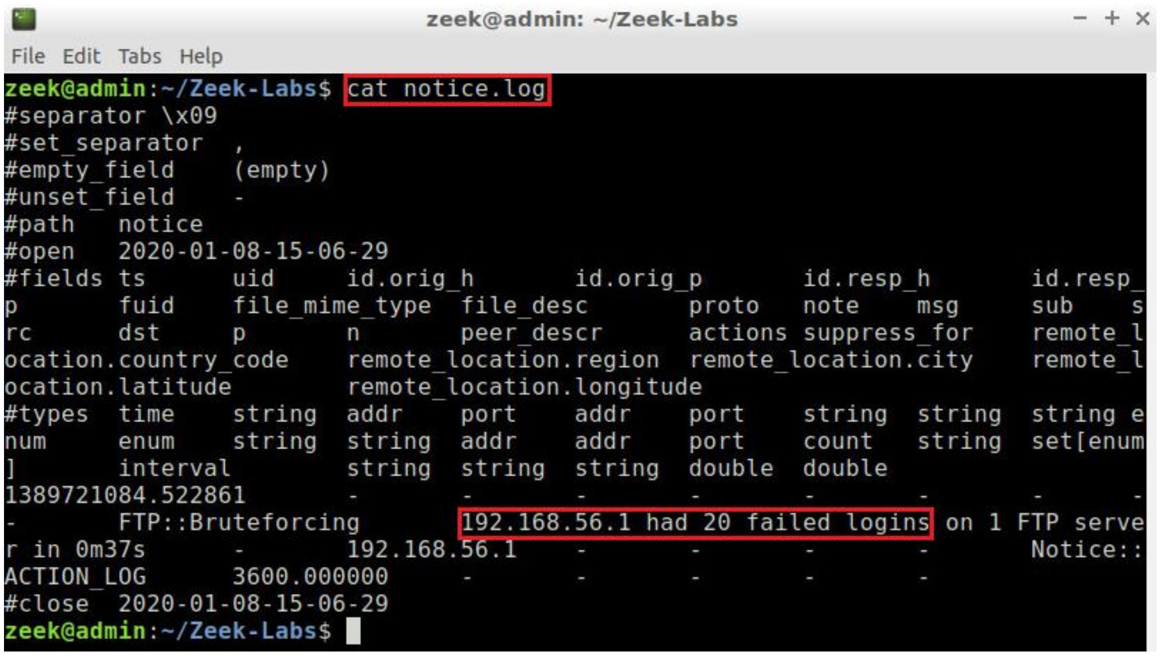


```
zeek@admin:~/Zeek-Labs$ zeek -C -r Sample-PCAP/bruteforce.pcap Lab-Scripts/ZeekD
etectBruteForce.zeek
zeek@admin:~/Zeek-Labs$
```

The `-C` option is included to prevent Zeek from displaying specific warnings.

Step 3. After running the command, if a brute forcing attack was found, it will be logged in the *notice.log* output log file. We will use the `cat` command to view the file.

```
cat notice.log
```



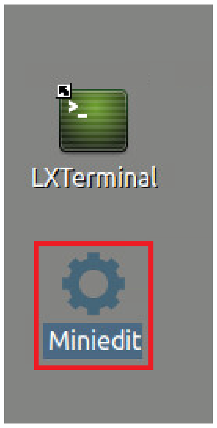
```
zeek@admin:~/Zeek-Labs$ cat notice.log
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   notice
#open   2020-01-08-15-06-29
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_
p       fuid    file_mime_type  file_desc       proto   note    msg     sub     s
rc      dst     p       n       peer_descr      actions suppress_for    remote_l
ocation.country_code    remote_location.region  remote_location.city    remote_l
ocation.latitude        remote_location.longitude
#types  time    string  addr    port    addr    port    string  string  string e
num     enum    string  string  addr    addr    port    count   string  set[enum
]       interval        string  string  string  double  double
1389721084.522861       -       -       -       -       -       -       -       -
-       FTP::Bruteforcing       192.168.56.1 had 20 failed logins on 1 FTP serve
r in 0m37s      -       192.168.56.1    -       -       -       -       Notice::
ACTION_LOG      3600.000000     -       -       -       -       -
#close  2020-01-08-15-06-29
zeek@admin:~/Zeek-Labs$
```

Examining the proceeding image, a possible brute force attack was detected. The log file shows that the IP address *192.168.56.1* had 20 or more failed login attempts on the hosted FTP server.

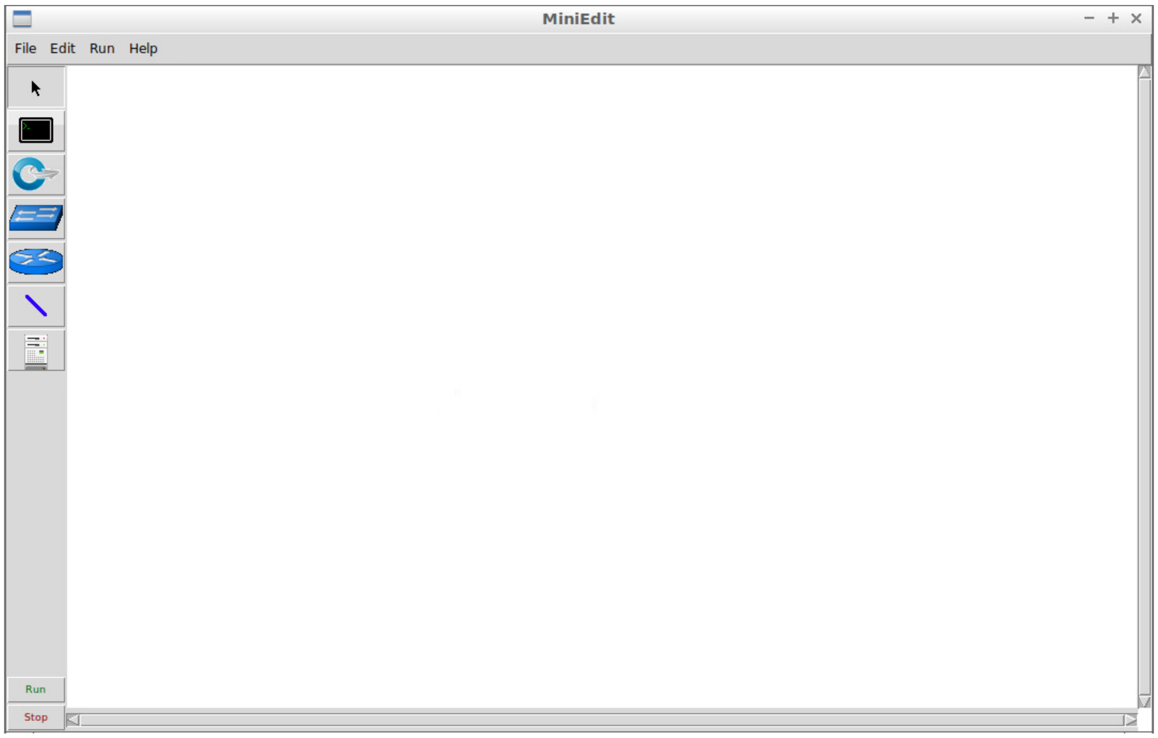
3.2 Launching Mininet

Mininet is a network emulator that creates a network topology consisting of virtual hosts, switches, controllers, and links. Within the Zeek lab series, we will be leveraging Mininet to generate and capture network traffic.

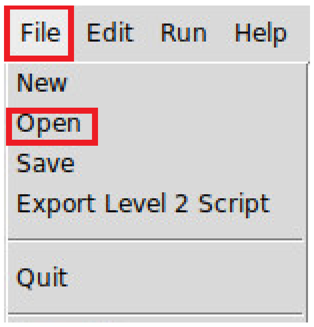
Step 1. From the *Client* machine's desktop, on the left side of the screen, double click on the MiniEdit icon as shown below. When prompted for a password, type `password` and hit `Enter`. The MiniEdit editor will now launch.



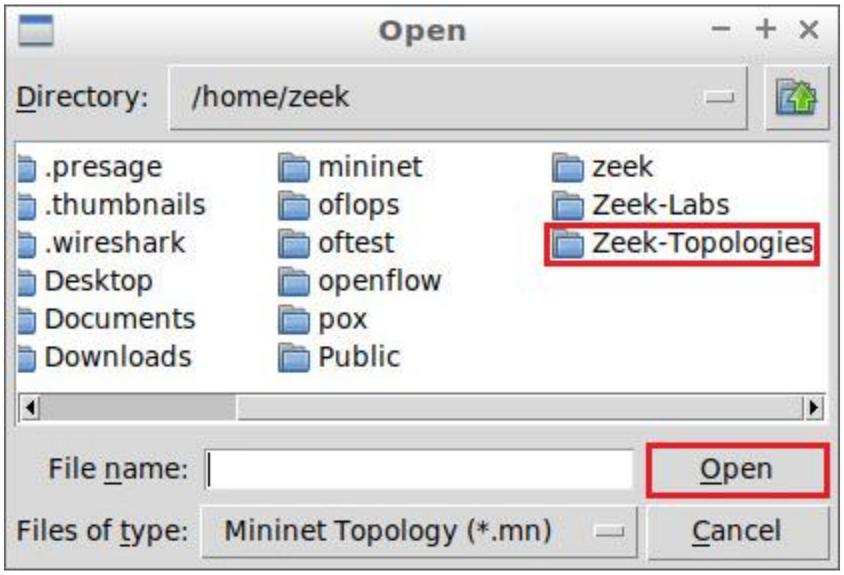
Step 2. The MiniEdit editor will now launch and allow for the creation of new, virtualized lab topologies. The image below shows the default MiniEdit display.



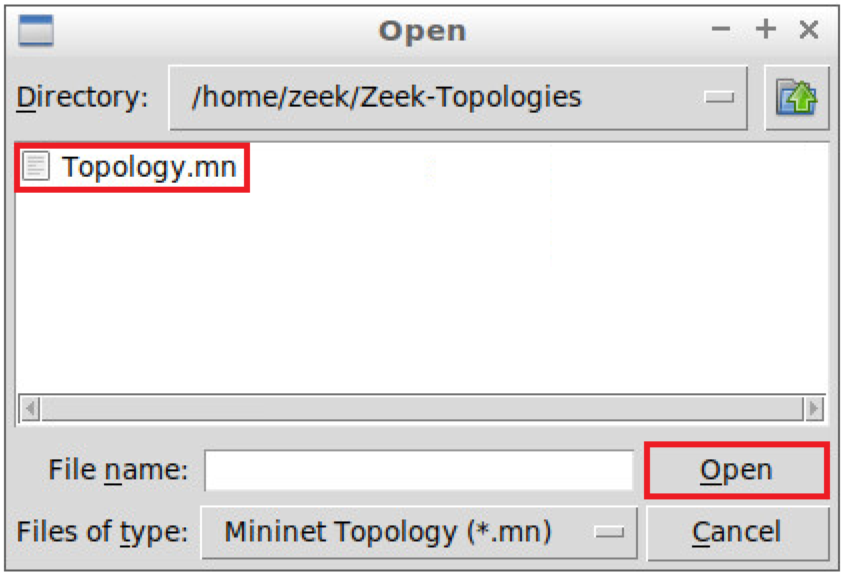
Step 3. A premade topology has already been created for this lab series. To load the correct topology, begin by clicking the `Open` button within the `File` tab on the top left of the MiniEdit editor.



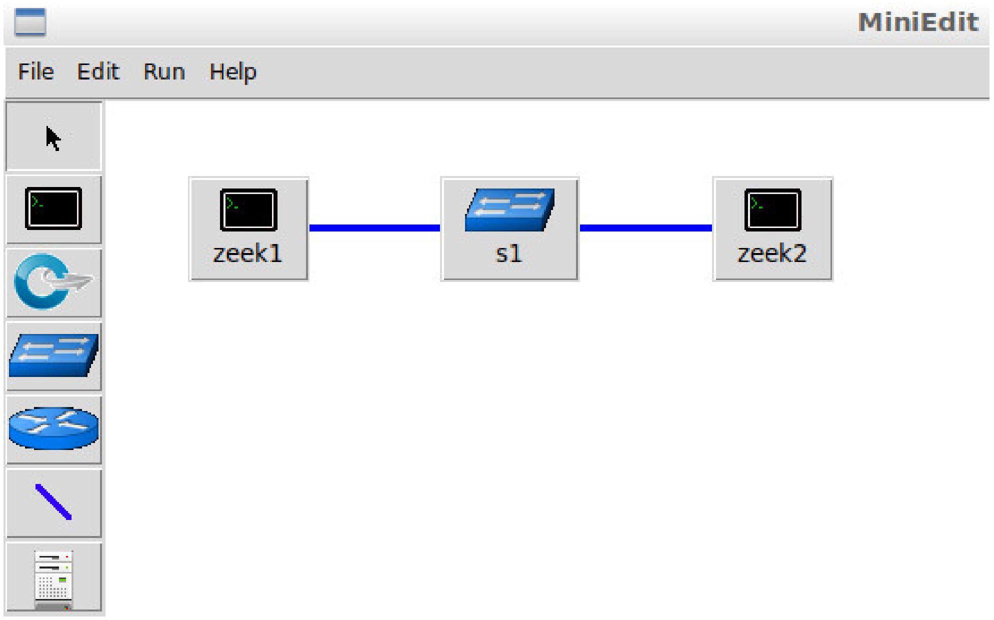
Step 4. Navigate to the Zeek-Topologies directory by scrolling to the right of the active directories and double clicking the Zeek-Topolgies icon, or by clicking the `Open` button.



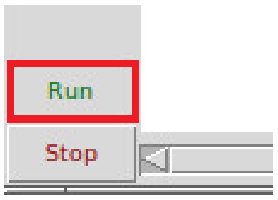
Step 5. Select the *Topology.mn* file by double clicking the *Topolgies.mn* icon, or by clicking the *Open* button.


Open
Directory: /home/zeek/Zeek-Topologies
Topology.mn
File name:
Open
Files of type: Mininet Topology (*.mn)
Cancel

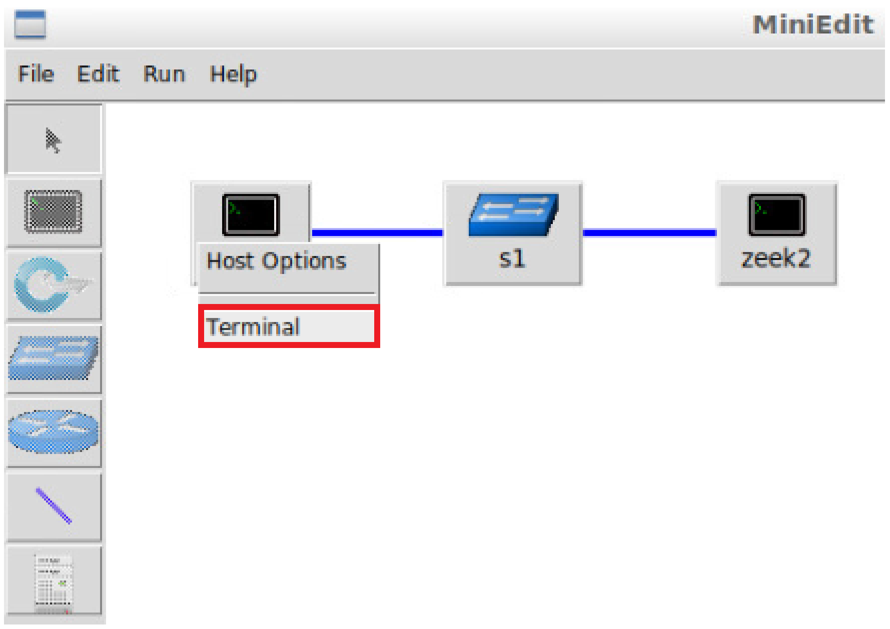
Step 6. The lab topology will contain two virtual machines – *zeek1* and *zeek2*, which are able to connect and communicate with one another through the *s1* switch, as seen in the image below.



Step 7. 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.



Step 8. To access either the *zeek1* or *zeek2* terminals for subsequent steps, hold the right mouse button on the desired machine, which will then display a *Terminal* button. Drag the cursor to the *Terminal* button to launch the terminal, as seen in the image below.



With the Mininet lab topology loaded, we can now begin to generate and analyze live network traffic capture.

3.3 Generating and analyzing live network traffic capture

The `tcpdump` command utility is a famous network packet analyzing tool that is used to display TCP/IP and other network packets being transmitted over the network[4].

3.3.1 Leveraging the Tcpdump command utility

The general format for `tcpdump` is the following command:

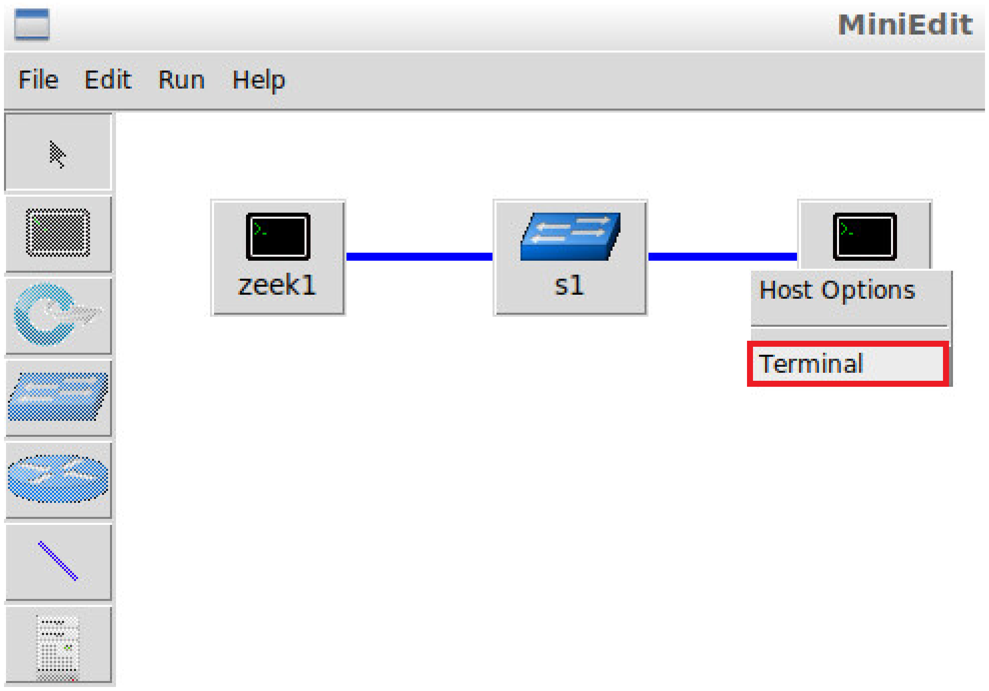
```
sudo tcpdump -i <interface_name> -s <num> -w <pcap_file_location>
```

- `sudo`: option to enable higher level privileges.
- `tcpdump`: program for capturing live network traffic.
- `-i`: option used to specify a network interface.
- `<interface_name>:` denotes the interface name.
- `-s`: option used to specify number of packets to capture.
- `<num>`: denotes the number of packets to capture. 0 equals infinite.
- `-w`: option used to specify that we will be writing to a new file.
- `<pcap_file_location>`: indicates the file location.

3.3.2 Capturing live network traffic

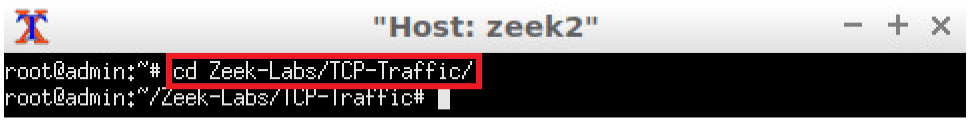
The *zeek2* machine will be used to capture live network traffic, while the *zeek1* machine will be used to generate live network traffic.

Step 1. Open the *zeek2* Terminal by hold the right mouse button on the desired machine, which will then display a *Terminal* button. Drag the cursor to the the *Terminal* button to launch the terminal, as seen in the image below.



Step 2. Navigate to the TCP-Traffic directory. To type capital letters, it is recommended to hold the *Shift* key while typing rather than using the *Caps* key.

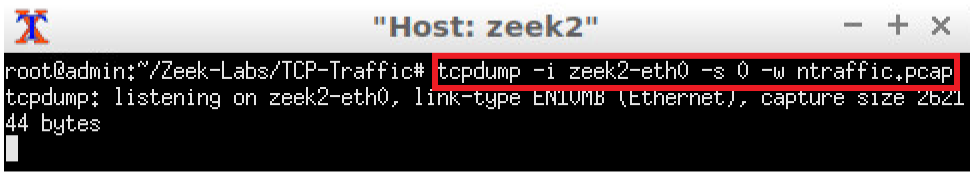
```
cd Zeek-Labs/TCP-Traffic/
```



```
root@admin:~# cd Zeek-Labs/TCP-Traffic/
root@admin:~/Zeek-Labs/TCP-Traffic#
```

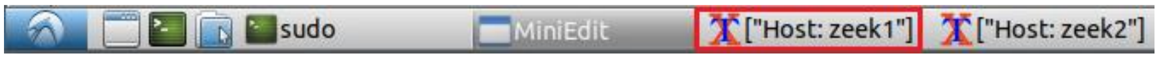
Step 3. Use the following command to begin live packet capture. If the Terminal session has not been terminated or closed, you may not be prompted to enter the password. If prompted for a password, type `password` and hit `Enter`. Live packet capture will start on interface *zeek2-eth0*.

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



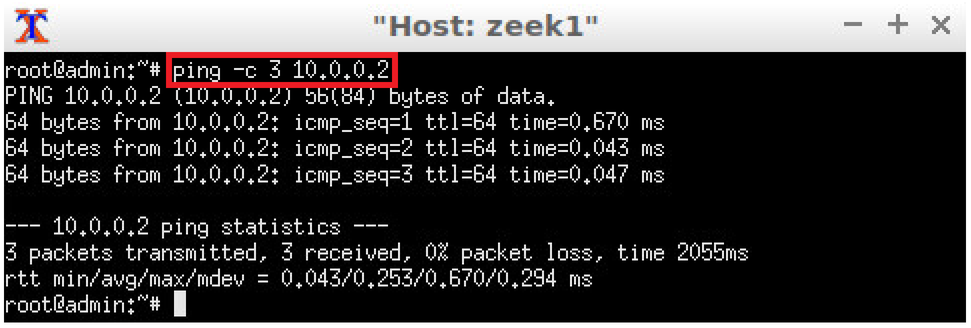
```
root@admin:~/Zeek-Labs/TCP-Traffic# tcpdump -i zeek2-eth0 -s 0 -w ntraffic.pcap
tcpdump: listening on zeek2-eth0, link-type EN10MB (Ethernet), capture size 2621
44 bytes
```

Step 4. Minimize the *zeek2 Terminal* and open the *zeek1 Terminal*. If necessary, right click within the Miniedit editor to activate your cursor.



Step 5. Generate traffic by using the `ping` utility. `Ping` operates by sending Internet Control Message Protocol (ICMP) echo request packets to the target host and waiting for an ICMP echo reply. Issue the following command on the newly opened *zeek1* Terminal.

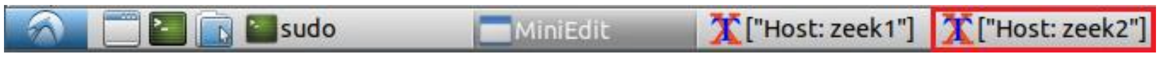
```
ping -c 3 10.0.0.2
```



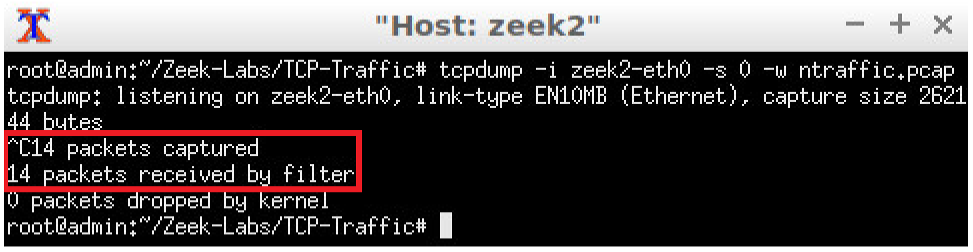
```
root@admin:~# ping -c 3 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.670 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.047 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2055ms
rtt min/avg/max/mdev = 0.043/0.253/0.670/0.294 ms
root@admin:~#
```

The `-c` option is used to indicate the number of packets to send – in this example, 3 packets.

Step 6. 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.

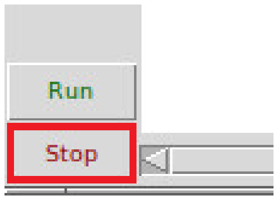


Step 6. Use the `Ctrl+c` key combination to stop live traffic capture. Statistics of the capture session will we be displayed. 14 packets were recorded by the interface, which were then captured and stored in the new *ntraffic.pcap* file.



```
root@admin:~/Zeek-Labs/TCP-Traffic# tcpdump -i zeek2-eth0 -s 0 -w ntraffic.pcap
tcpdump: listening on zeek2-eth0, link-type EN10MB (Ethernet), capture size 2621
44 bytes
^C14 packets captured
14 packets received by filter
0 packets dropped by kernel
root@admin:~/Zeek-Labs/TCP-Traffic#
```

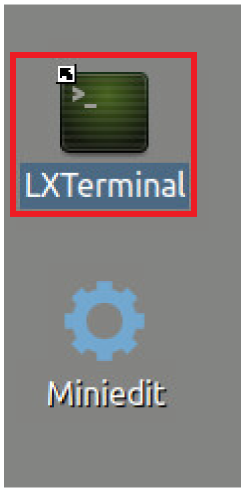
Step 7. Stop the current Mininet session by clicking the *Stop* button on the bottom left of the MiniEdit editor and close the MiniEdit editor by clicking the `x` on the top right of the editor.



We will now return to the *Client* machine to process and analyze the newly generated network traffic.

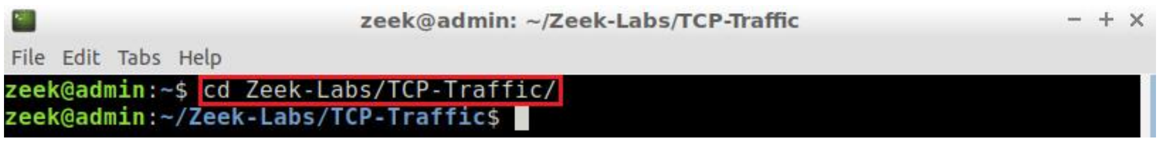
3.3.3 Analyzing the newly captured network traffic

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



Step 2. Navigate to the *TCP-Traffic* directory to find the *ntraffic.pcap* file. To type capital letters, it is recommended to hold the *Shift* key while typing rather than using the *Caps* key.

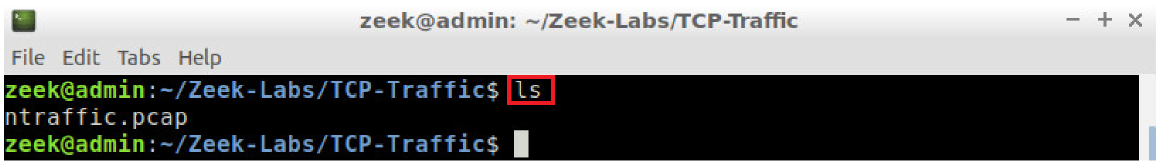
```
cd Zeek-Labs/TCP-Traffic/
```



```
zeek@admin:~$ cd Zeek-Labs/TCP-Traffic/
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

Step 3. View the file contents of the *TCP-Traffic* directory.

```
ls
```

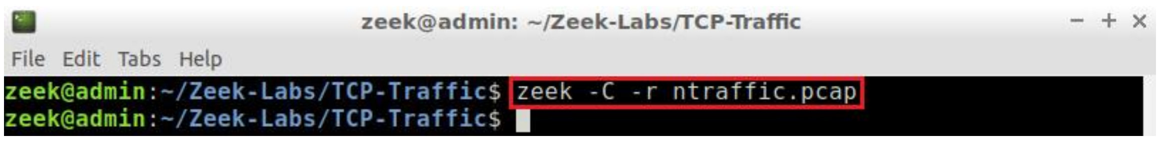


```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ ls
ntraffic.pcap
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

We can see the *ntraffic.pcap* file that was generated by the *zeek2* machine is now accessible.

Step 4. Initialize Zeek offline packet parsing on the packet capture file. The `-r` option is used to read from a given pcap file, and the `-C` option is used to disable checksums validation.

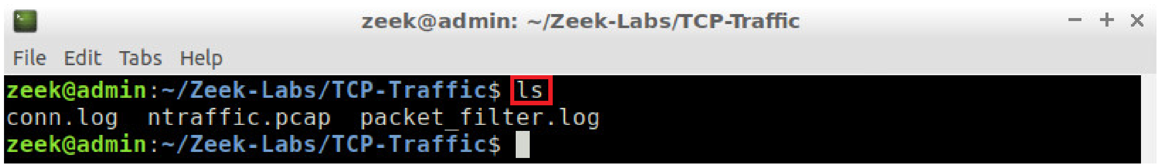
```
zeek -C -r ntraffic.pcap
```



```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ zeek -C -r ntraffic.pcap
zeek@admin:~/Zeek-Labs/TCP-Traffic$ 
```

Step 5. View the newly generated Zeek log files.

```
ls
```

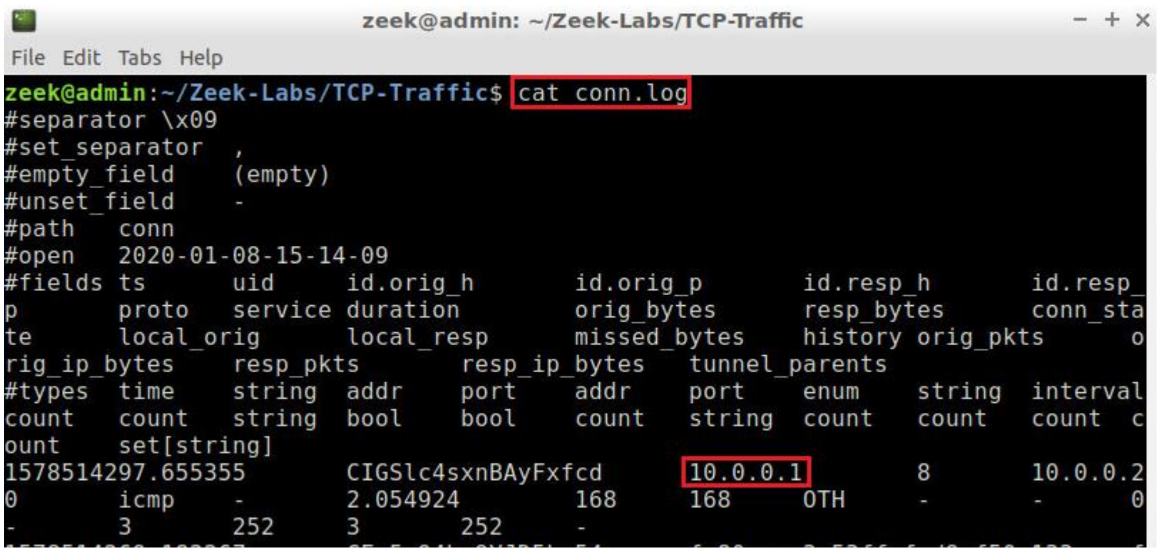


```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ ls
conn.log  ntraffic.pcap  packet_filter.log
zeek@admin:~/Zeek-Labs/TCP-Traffic$ 
```

The generated log files will contain important information regarding the network traffic. For instance, the *conn.log* file will contain connection-based information, specifically the hosts communicating, their IP addresses, protocols and ports. The following labs will offer in-depth insight and examples towards understanding these Zeek log files.

Step 6. Viewing the *conn.log* connection-based log file with the `cat` command, we can see that the IP address *10.0.0.1* was detected to generate the captured traffic, corresponding to the *zeek1* virtual machine.

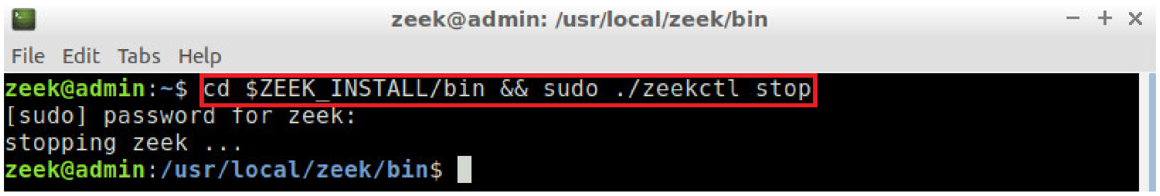
```
cat conn.log
```



```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ cat conn.log
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   conn
#open   2020-01-08-15-14-09
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_
p       proto   service duration        orig_bytes      resp_bytes      conn_sta
te      local_orig      local_resp      missed_bytes    history orig_pkts       o
rig_ip_bytes    resp_pkts       resp_ip_bytes   tunnel_parents
#types  time    string  addr    port    addr    port    enum    string  interval
count   count   string  bool    bool    count   string  count   count   count   c
ount    set[string]
1578514297.655355       CIGSlc4sxnBAyFxfcd      10.0.0.1        8       10.0.0.2
0       icmp    -       2.054924        168     168     OTH     -       -       0
-       3       252     3       252     -
```

Step 7. 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
```



```
zeek@admin:~$ cd $ZEEK_INSTALL/bin && sudo ./zeekctl stop
[sudo] password for zeek:
stopping zeek ...
zeek@admin:/usr/local/zeek/bin$
```

The above command navigates to Zeek's installation directory and executes the stop command in *zeekctl*.

If you see error messages during the new Zeek instance initializing process, please ignore it.

Concluding this lab, we have reviewed Zeek's architecture and event-based engine, as well as introduced both offline and live network traffic capture.

References

1. "Zeek documentation", [Online]. Available: https://docs.zeek.org/en/stable/intro/index.html
2. Sommer, Robin, and Vern Paxson, "Enhancing byte-level network intrusion detection signatures with context," *In Proceedings of the 10th ACM conference on Computer and communications security*, pp. 262-271. ACM, 2003.

3. "Signature-based intrusion detection", [Online]. Available: http://www.cs.unc.edu/~jeffay/courses/nidsS05/slides/6-Sig-based Detection.pdf.
4. Joseph, D. A., Paxson, V., & Kim, S. (2006). tcpdump Tutorial. University of California, EE122 Fall.