The University of Texas at San Antonio™

The Cyber Center for Security and Analytics

UNIVERSITY OF SOUTH CAROLINA

ZEEK INSTRUSION DETECTION SERIES

Lab 2: An Overview of Zeek Logs

Document Version: 03-13-2020

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

Contents

Overview

This lab covers Zeek's logging files. Zeek's event-based engine will generate log files based on signatures or events found during network traffic analysis. The focus in this lab is on explaining each logging file and introducing some basic analytic functions and tools.

Objectives

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

1. Generate Zeek log files.
2. Use Linux Terminal tools combined with Zeek's *zeek-cut* utility to customize the output of logs for analysis.

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.

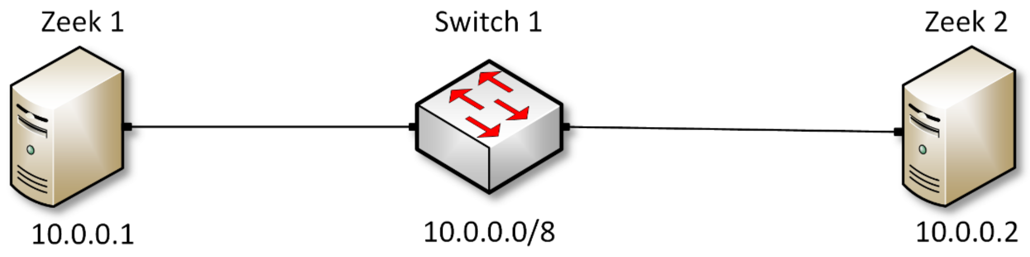


Figure 1. Lab topology.

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 logs.
2. Section 2: Starting a new instance of Zeek.
3. Section 3: Parsing packet capture files into Zeek log files.
4. Section 4: Analyzing Zeek log files.

1 Introduction to Zeek Logs

Zeek's generated log files include a comprehensive record of every connection seen on the wire; this includes application-layer protocols and fields (e.g., Hyper-Text Transfer Protocol (HTTP) sessions, Uniform Resource Locator (URL), key headers, Multi-Purpose Internet Mail Extensions (MIME) types, server responses, etc.), Domain Name Server (DNS) requests and responses, Secure Socket Layer (SSL) certificates, key content of Simple Mail Transfer Protocol (SMTP) sessions, and others.

1.1 Zeek Logs generated by packet analysis

A Zeek log is a stream of high-level entries that correspond to network activities, such as a login to SSH or an email sent using SMTP. In Zeek, each event stream has a dedicated file with its own set of features, fields, or columns.

During capture or analysis, Zeek generates a log determined by the protocol type. Due to this architecture, a Session Initiation Protocol (SIP) log for instance, does not contain any other protocols' packets information like HTTP. Furthermore, each log file contains case-relative fields (e.g., *from* and *subject* fields in an SMTP log). Some of these log files are large and contain entries that can be either benign or malicious, whereas others are smaller and contain more actionable information.

1.2 Zeek Logs generated by recurrent network analysis

With every session of packet analysis, either through live packet analysis or the parsing of an offline packet capture file, Zeek generates session-specific log files. In addition to these session-based log files, Zeek creates network-reliant log files as well. These network-reliant files are continually generated and updated when a new session is initialized and started.

The following Zeek log files are updated daily:

- *known_hosts.log*: Log file containing information for hosts that completed TCP handshakes.
- *known_services.log*: Log file containing a list of services running on hosts.
- *known_certs*.log: Log file containing a list of Secure Socket Layer (SSL) certificates.
- *software.log*: Log file containing information about Software being used on the network.

Additionally, a list of detection-based log files is created during each session. The log files relevant to this lab are:

- *notice.log* (Zeek notices): When Zeek detects an anomaly, a corresponding notice will be raised in this file.
- *intel.log* (Intelligence data matches): When Zeek detects traffic flagged with known malicious indicators, a corresponding reference will be logged in this file.
- *signatures.log* (Signature matches): When Zeek detects traffic flagged with known malicious or faulty packet signatures, a corresponding reference will be logged in this file.

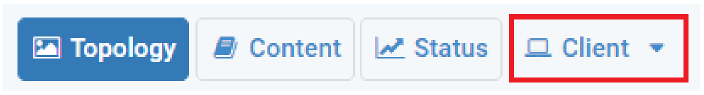
1.3 Typical uses of Zeek Logs

By default, Zeek logs all information into well-structured, tab-separated text files suitable for postprocessing. Users can also choose from a set of alternative output formats and backends such as external databases.

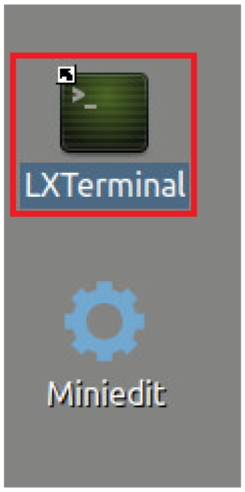
The Zeek-native `zeek-cut` utility can be leveraged to further specify and parse the information within the generated log files.

2 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.

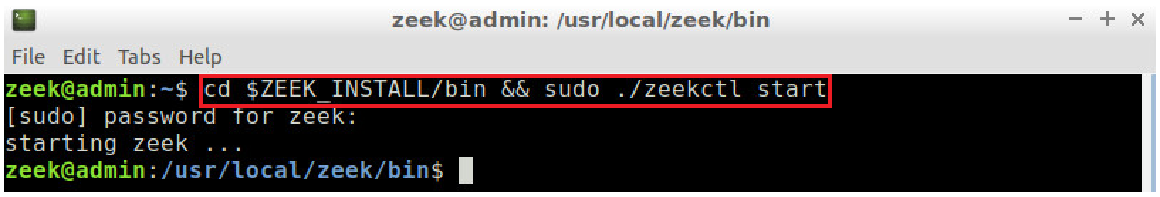


Step 2. The *Client* machine will now open, and the desktop will be displayed. On the left side of the screen, double click on the LXTerminal icon as shown below.



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. When prompted for a password, type `password` and hit `Enter`. 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 start
```



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

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

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

3 Parsing packet capture files into Zeek log files

In this section we introduce Zeek's capability of generating and viewing log files. Packet capture files used in this lab are preloaded onto the *Client* machine, and can be found with the following path:

```
~/Zeek-Labs/Sample-PCAP/
```

These packet capture files were downloaded from `Tcpreplay's` sample capture collection. To access the following link, users must have access to an external computer connected to the Internet, because the Zeek Lab topology does not have an active Internet connection.

```
http://tcpreplay.appneta.com/wiki/captures.html
```

`Tcpreplay` is a suite of free Open Source utilities for editing and replaying previously captured network traffic and can be used to test transmissions and network health.

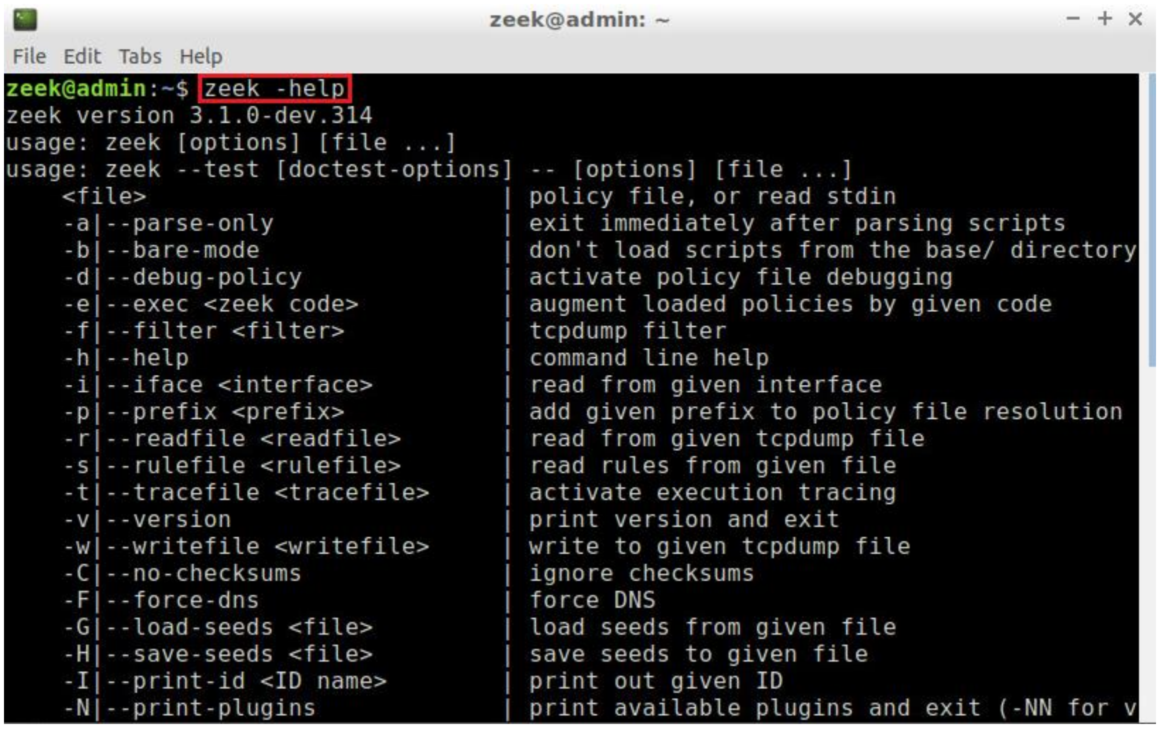
3.1 Overview of Zeek command options

When using Zeek, the user specifies a running state option. In this lab, three primarily options are used:

- `-C`: specifies to ignore checksum warnings, specifically to avoid redundancy since we are focusing on TCP traffic only.
- `-r`: specifies offline packet capture file analysis.
- `-w`: specifies live network capture.

Additional Zeek options can be found by passing the `-help` option to the `zeek` command:

```
zeek -help
```



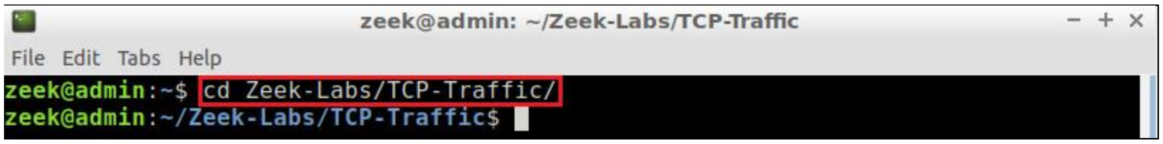
```
zeek@admin:~$ zeek -help
zeek version 3.1.0-dev.314
usage: zeek [options] [file ...]
usage: zeek --test [doctest-options] -- [options] [file ...]
    <file>                          | policy file, or read stdin
    -a|--parse-only                 | exit immediately after parsing scripts
    -b|--bare-mode                  | don't load scripts from the base/ directory
    -d|--debug-policy               | activate policy file debugging
    -e|--exec <zeek code>           | augment loaded policies by given code
    -f|--filter <filter>            | tcpdump filter
    -h|--help                       | command line help
    -i|--iface <interface>          | read from given interface
    -p|--prefix <prefix>            | add given prefix to policy file resolution
    -r|--readfile <readfile>        | read from given tcpdump file
    -s|--rulefile <rulefile>        | read rules from given file
    -t|--tracefile <tracefile>      | activate execution tracing
    -v|--version                    | print version and exit
    -w|--writefile <writefile>      | write to given tcpdump file
    -C|--no-checksums               | ignore checksums
    -F|--force-dns                  | force DNS
    -G|--load-seeds <file>          | load seeds from given file
    -H|--save-seeds <file>          | save seeds to given file
    -I|--print-id <ID name>         | print out given ID
    -N|--print-plugins              | print available plugins and exit (-NN for v
```

3.2 Using Zeek to process offline packet capture files

In this subsection we will use Zeek to process the existing offline packet capture file *smallFlows.pcap*. By specifying the `-r` option and the directory path to the pcap file, Zeek can generate the corresponding log files.

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

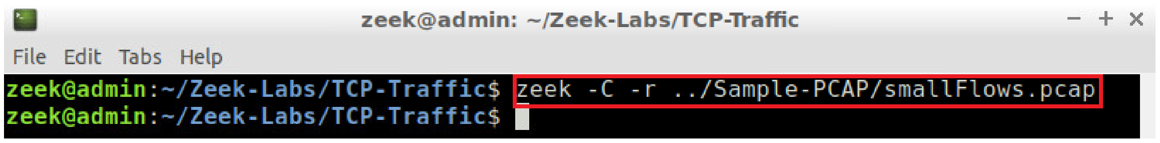
```
cd Zeek-Labs/TCP-Traffic/
```



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

Step 2. Use the following command to process the *smallFlows.pcap* file. It is possible to use the `tab` key to autocomplete the longer paths.

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

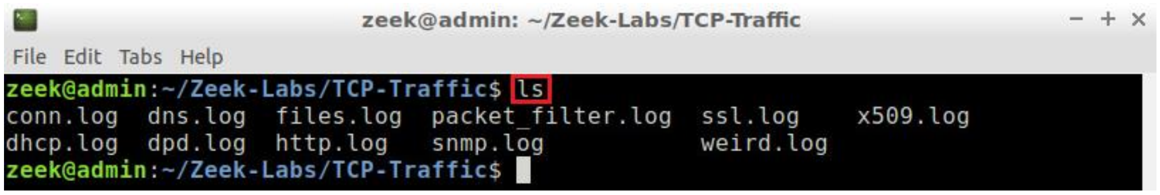


```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ zeek -C -r ../Sample-PCAP/smallFlows.pcap
zeek@admin:~/Zeek-Labs/TCP-Traffic$ 
```

After Zeek finishes processing the packet capture file, it will generate a number of log files.

Step 3. Use the following command to list the generated log files.

```
ls
```



```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ ls
conn.log  dns.log  files.log  packet_filter.log  ssl.log    x509.log
dhcp.log  dpd.log  http.log   snmp.log           weird.log
zeek@admin:~/Zeek-Labs/TCP-Traffic$ 
```

3.3 Understanding Zeek log files

Zeek's generated log files can be summarized as follows:

- *conn.log*: A file containing information pertaining to all TCP/UDP/ICMP connections, this file contains most of the information gathered from the packet capture.

- *files.log*: A file consisting of analytic results of packets' counts and sessions' durations.
- *packet_filter.log*: A file listing the active filters applied to Zeek upon reading the packet capture file.
- *x509.log*: A file containing public key certificates used by protocols.
- *weird.log*: A file containing packet data non-conformant with standard protocols. It also contains packets with possibly corrupted or damaged packet header fields.
- *(protocol).log* (*dns.log, dhcp.log, http.log, snmp.log*): These are files containing information for packets found in each respective protocol. For instance, *dns.log* will only contain information generated by Domain Name Service (DNS) packets.

More information regarding log files is available in the Zeek official documentation, which can be viewed online using an external Internet-connected machine through this link:

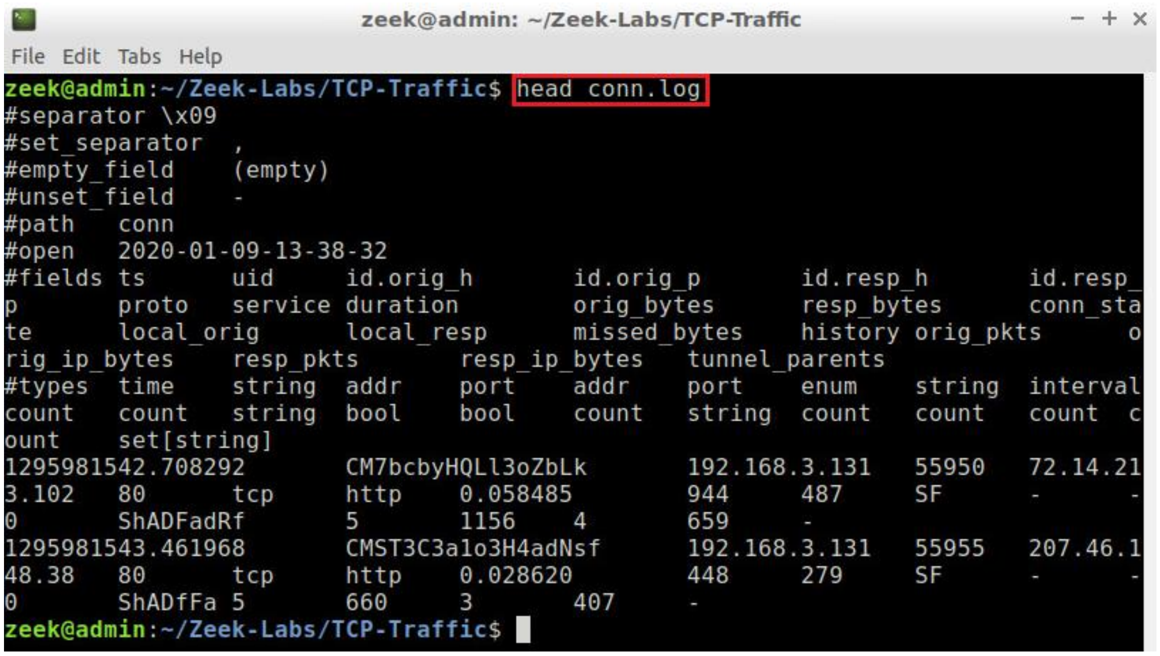
```
https://docs.zeek.org/en/stable/script-reference/log-files.html
```

3.4 Basic viewing of Zeek logs

In this subsection we examine the generated log files and their contents.

Step 1. Use the following command to display the contents of the *conn.log* file using the `head` command.

```
head conn.log
```



```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ head conn.log
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   conn
#open   2020-01-09-13-38-32
#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]
1295981542.708292       CM7bcbyHQLl3oZbLk       192.168.3.131   55950   72.14.21
3.102   80      tcp     http    0.058485        944     487     SF      -       -
0       ShADFadRf       5       1156    4       659     -
1295981543.461968       CMST3C3a1o3H4adNsf      192.168.3.131   55955   207.46.1
48.38   80      tcp     http    0.028620        448     279     SF      -       -
0       ShADfFa 5       660     3       407     -
zeek@admin:~/Zeek-Labs/TCP-Traffic$ 
```

The topmost rows within the *conn.log* file will be displayed in the Terminal; however, the current formatting wraps around multiple lines, making it unclear and hard to understand. In the following section we introduce the `zeek-cut` utility for enhancing the output of these log files.

4 Analyzing Zeek log files

In this section, we review the utilities that help in displaying log files with well-formatted outputs, as well as saving output to text files.

4.1 Leveraging zeek-cut for a more refined view of log files

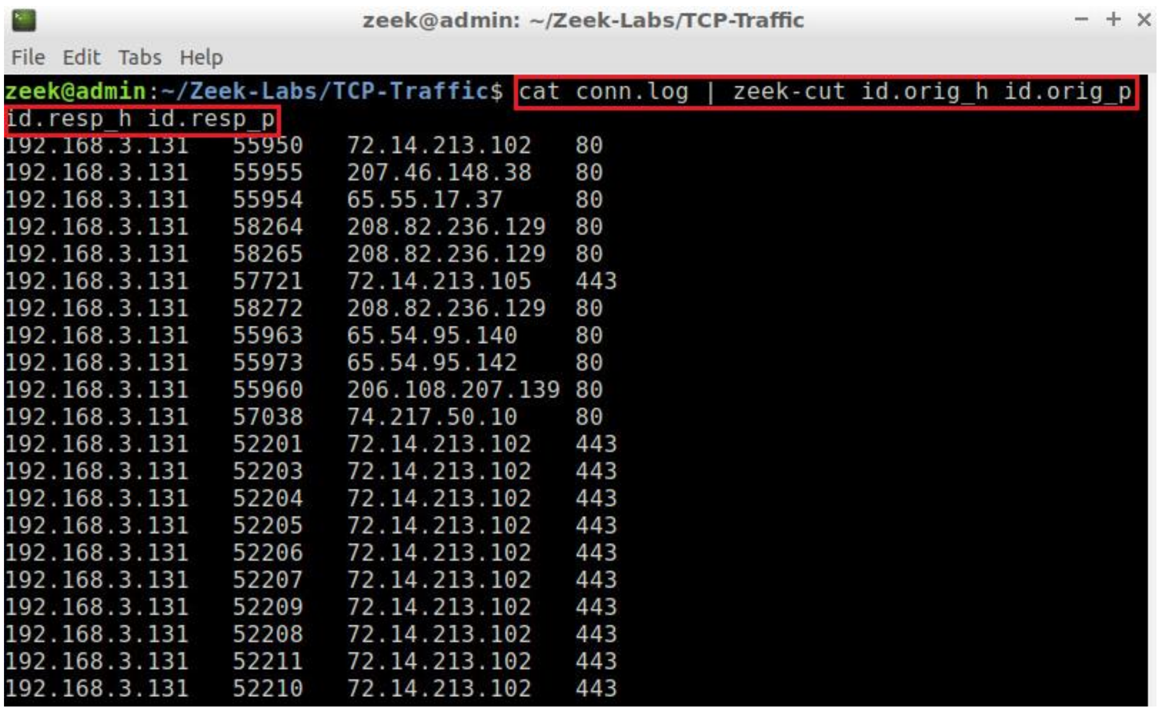
Although the produced log file is tab delimited, it is difficult to visualize and parse information from the terminal. The `zeek-cut` utility can be used to parse the log files by specifying which column data to be displayed in a more organized output.

4.1.1 Using zeek-cut in conjunction with cat and head command utilities

Generally, the `zeek-cut` utility is typically coupled with `cat` using the `pipe |` command. In Linux, the `pipe` command sends the output of one command as input to another. Essentially, the output of the left command is passed as input to that on its right, and multiple commands can be chained together.

Step 1. Use the following command to pipe the contents of `cat` into `zeek-cut`.

```
cat conn.log | zeek-cut id.orig_h id.orig_p id.resp_h id.resp_p
```



```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ cat conn.log | zeek-cut id.orig_h id.orig_p
id.resp_h id.resp_p
192.168.3.131   55950   72.14.213.102   80
192.168.3.131   55955   207.46.148.38   80
192.168.3.131   55954   65.55.17.37     80
192.168.3.131   58264   208.82.236.129  80
192.168.3.131   58265   208.82.236.129  80
192.168.3.131   57721   72.14.213.105   443
192.168.3.131   58272   208.82.236.129  80
192.168.3.131   55963   65.54.95.140    80
192.168.3.131   55973   65.54.95.142    80
192.168.3.131   55960   206.108.207.139 80
192.168.3.131   57038   74.217.50.10    80
192.168.3.131   52201   72.14.213.102   443
192.168.3.131   52203   72.14.213.102   443
192.168.3.131   52204   72.14.213.102   443
192.168.3.131   52205   72.14.213.102   443
192.168.3.131   52206   72.14.213.102   443
192.168.3.131   52207   72.14.213.102   443
192.168.3.131   52209   72.14.213.102   443
192.168.3.131   52208   72.14.213.102   443
192.168.3.131   52211   72.14.213.102   443
192.168.3.131   52210   72.14.213.102   443
```

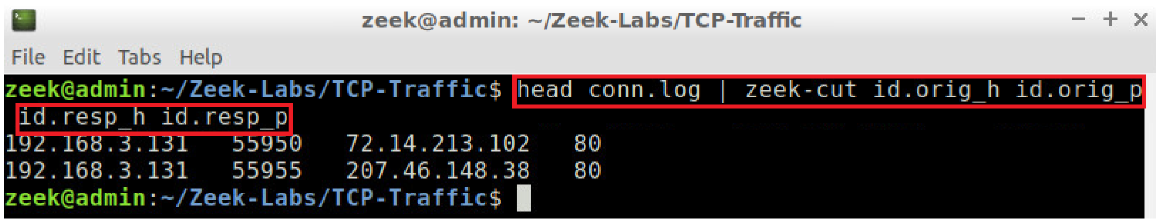
The options passed into the `zeek-cut` utility represent the column headers to be extracted from the log file:

- `id.orig_h`: Column containing the source IP address.
- `id.orig_p`: Column containing the source port.
- `id.resp_h`: Column containing the destination IP address.
- `id.resp_p`: Column containing the destination port.

Alternatively, instead of using the `cat` command, the `head` command can be used to display the topmost rows of the log file, which can be very useful to view a large file's contents.

Step 2. Use the following command to pipe the contents of `head` into `zeek-cut`.

```
head conn.log | zeek-cut id.orig_h id.orig_p id.resp_h id.resp_p
```



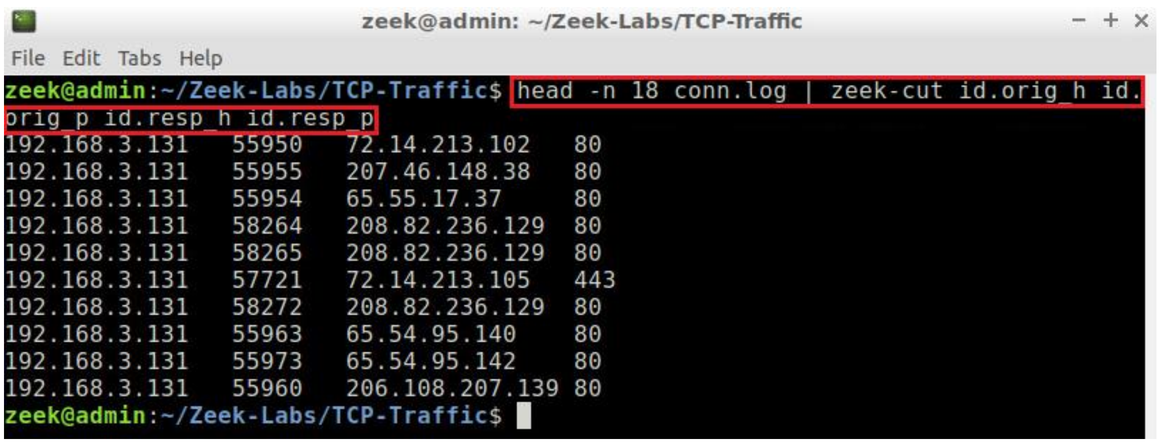
```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ head conn.log | zeek-cut id.orig_h id.orig_p
 id.resp_h id.resp_p
192.168.3.131   55950   72.14.213.102   80
192.168.3.131   55955   207.46.148.38   80
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

Notice that only two records are shown. This is caused by the `head` command taking the 10 topmost rows of *conn.log*, regardless of what that entails, and passing it as input to `zeek-cut`.

Since the log file contains 8 lines of header padding used for displaying the file's format, we will have to specify the first 18 rows of file in order to succesfully display the first 10 packets of the log file.

Step 3. Use the following command to pipe the contents of `head` into `zeek-cut`.

```
head -n 18 conn.log | zeek-cut id.orig_h id.orig_p id.resp_h id.resp_p
```


zeek@admin: ~/Zeek-Labs/TCP-Traffic

File Edit Tabs Help

```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ head -n 18 conn.log | zeek-cut id.orig_h id.orig_p id.resp_h id.resp_p
192.168.3.131   55950   72.14.213.102   80
192.168.3.131   55955   207.46.148.38   80
192.168.3.131   55954   65.55.17.37     80
192.168.3.131   58264   208.82.236.129  80
192.168.3.131   58265   208.82.236.129  80
192.168.3.131   57721   72.14.213.105   443
192.168.3.131   58272   208.82.236.129  80
192.168.3.131   55963   65.54.95.140    80
192.168.3.131   55973   65.54.95.142    80
192.168.3.131   55960   206.108.207.139 80
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

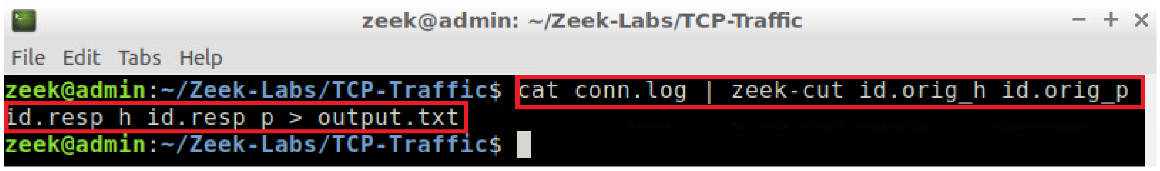
The `-n` option can be passed to the `head` utility to specify the desired number of rows.

4.1.2 Printing the output of zeek-cut to a text file

While the results displayed in the Terminal after using the `zeek-cut` utility can be easily viewed for smaller datasets, it is often necessary to save the output into a separate file. Using the `>` character, we can send the output to a new file for further processing by other applications.

Step 1. Use the following command to change the output location of `zeek-cut`.

```
cat conn.log | zeek-cut id.orig_h id.orig_p id.resp_h id.resp_p > output.txt
```


zeek@admin: ~/Zeek-Labs/TCP-Traffic

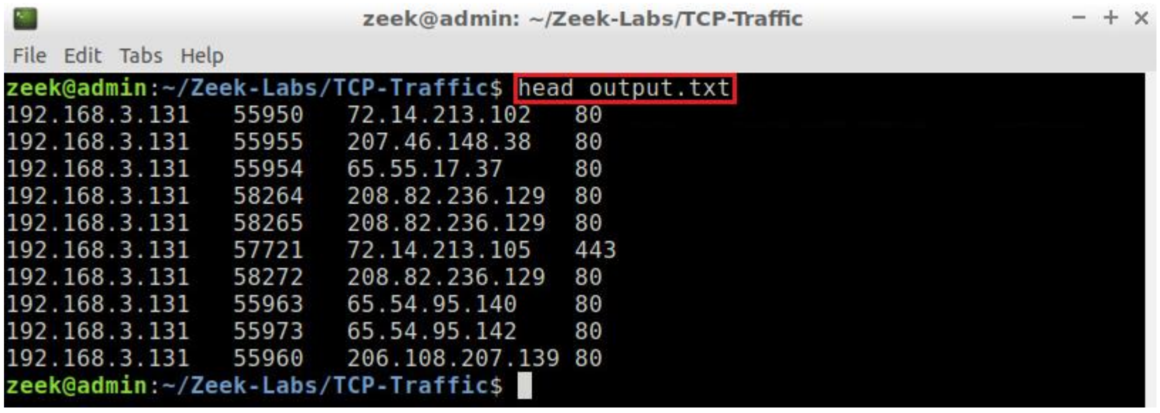
File Edit Tabs Help

```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ cat conn.log | zeek-cut id.orig_h id.orig_p id.resp_h id.resp_p > output.txt
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

By including the file extension in *output.txt*, we are choosing to print the output into a plain text file.

Step 2. We can display the topmost contents of the new *output.txt* file by using the `head` command.

```
head output.txt
```



```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ head output.txt
192.168.3.131   55950   72.14.213.102   80
192.168.3.131   55955   207.46.148.38   80
192.168.3.131   55954   65.55.17.37     80
192.168.3.131   58264   208.82.236.129  80
192.168.3.131   58265   208.82.236.129  80
192.168.3.131   57721   72.14.213.105   443
192.168.3.131   58272   208.82.236.129  80
192.168.3.131   55963   65.54.95.140    80
192.168.3.131   55973   65.54.95.142    80
192.168.3.131   55960   206.108.207.139 80
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

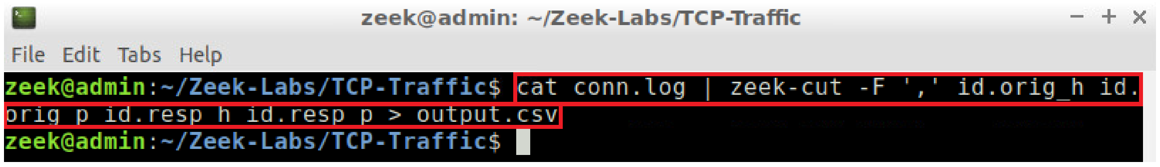
The *output.txt* file contains the same tab-delimited format as shown in previous `zeek-cut` examples.

4.1.3 Printing the output of zeek-cut to a csv file

In some situations, it is helpful to save the output of `zeek-cut` in a csv file. In a csv file, data may be imported into other applications, such as databases or machine learning classifiers.

Step 1. The exported output file by `zeek-cut` is tab-delimited due to the default `zeek-cut` settings. To export a file with another delimiter, the `-F` option is used.

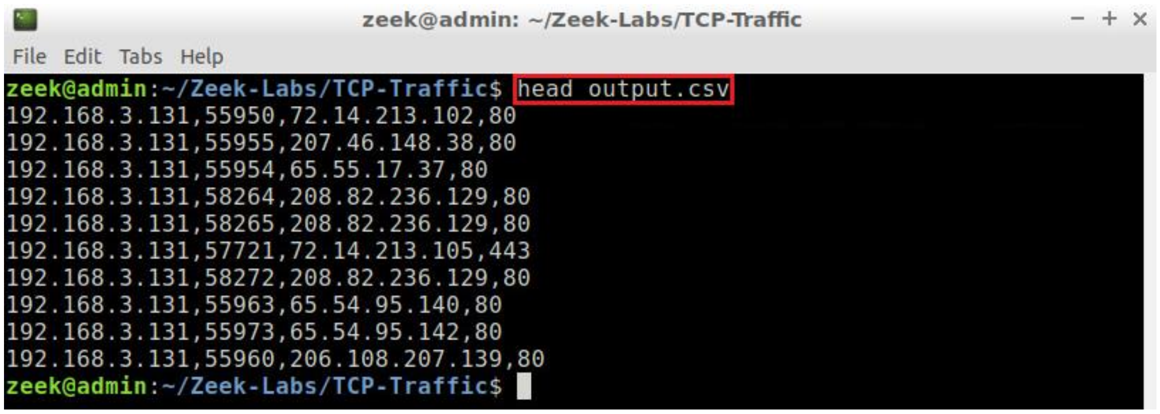
```
cat conn.log | zeek-cut -F ‘,’ id.orig_h id.orig_p id.resp_h id.resp_p >
output.csv
```



```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ cat conn.log | zeek-cut -F ',' id.orig_h id.
orig_p id.resp_h id.resp_p > output.csv
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

Step 2. We can now display the topmost contents of the *output.csv* file.

```
head output.csv
```



```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ head output.csv
192.168.3.131,55950,72.14.213.102,80
192.168.3.131,55955,207.46.148.38,80
192.168.3.131,55954,65.55.17.37,80
192.168.3.131,58264,208.82.236.129,80
192.168.3.131,58265,208.82.236.129,80
192.168.3.131,57721,72.14.213.105,443
192.168.3.131,58272,208.82.236.129,80
192.168.3.131,55963,65.54.95.140,80
192.168.3.131,55973,65.54.95.142,80
192.168.3.131,55960,206.108.207.139,80
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

As shown in the image, the *output.csv* file is in a comma-delimited format, rather than the previous tab-delimited format.

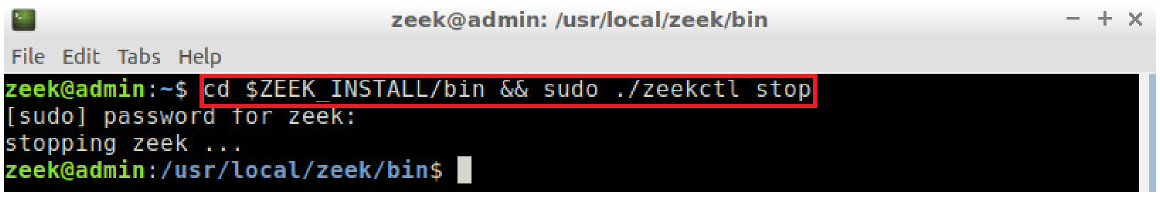
In conclusion, `zeek-cut` is a flexible tool that can be called to format Zeek log files depending on the user's needs. The `zeek-cut` utility can be utilized with more advanced commands to further increase customization.

4.2 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
```



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

Concluding this lab, we have reviewed Zeek's output log files in more depth while introducing some of the more relevant network-based log files and introduced some basic utilities to view these log files.

References

1. “Log files”, Zeek user manual, [Online]. Available: Zeek, docs.zeek.org/en/stable/script-reference/log-files.html.
2. “Sample captures”, *Tcpreplay*, [Online]. Available: tcpreplay.appneta.com/wiki/captures.html