The University of Texas at San Antonio™

The Cyber Center for Security and Analytics

UNIVERSITY OF SOUTH CAROLINA

ZEEK INSTRUSION DETECTION

Lab 3: Parsing, Reading and Organizing Zeek Log Files

Document Version: 03-13-2020

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

Contents

Overview

This lab explains how to format and organize Zeek's log files by combining zeek-cut utility with basic Linux shell commands. Utilities and tools introduced in this lab provide practical examples for logs customization in a real network environment.

Objectives

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

1. Use Linux tools and commands for text files processing.
2. Practice Linux shell scripts and the AWK scripting language.
3. Incorporate AWK with zeek-cut to provide formatted logs.

Lab topology

Figure 1 shows the lab topology. The topology uses 10.0.0.0/8 which is the default network assigned by Mininet.

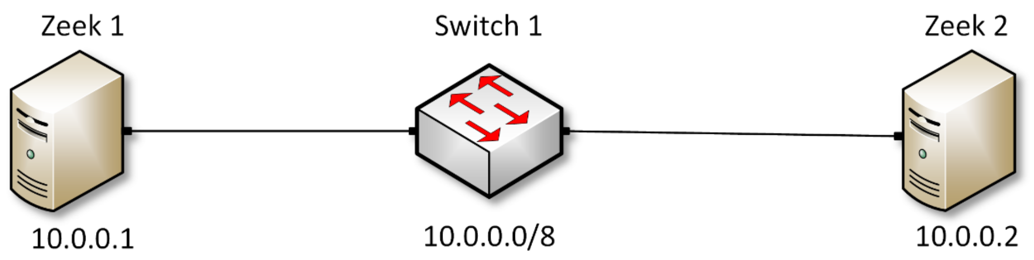


Figure 1. Lab topology.

Lab settings

The information (case-sensitive) in the table below provides the credentials 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 shell scripts.
2. Section 2: Advanced zeek-cut log file analysis.
3. Section 3: Incorporating the AWK scripting language for log file analysis.

1 Introduction to shell scripts

A shell script is a text file containing commands to be executed by the Unix command-line interpreter. Shell scripts provide a convenient way to manipulate files and automate programs' executions. Selection and repetition are incorporated into scripts to branch control based on conditioning and looping statements. Running a shell script can immensely save time and prevent manually entering repetitive commands in recurrent tasks.

1.1 Ubuntu Linux text editors

Linux-based distributions include pre-installed text editors like `nano`, `vi`, `vim`, `gedit`, etc. `nano` is a keyboard-oriented lightweight text editor with a simple Command Line Interface (CLI). Other editors such as `vi` and `vim` are highly customizable and extensible, making them attractive for users that demand a large amount of control and flexibility over their text editing environment. Alternatively, the Graphical User Interface (GUI) text editor `gedit` can be used to visually work outside of the terminal. More information on these text editors can be found on the Ubuntu help pages. To access the following links, users must have access to an external computer connected to the Internet, because the Zeek Lab topology does not have an active Internet connection.

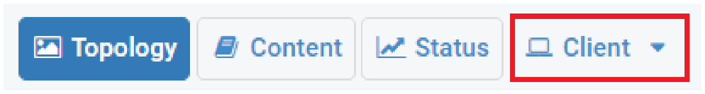
- `Nano` – `https://help.ubuntu.com/community/Nano`
- `Vim` – `https://help.ubuntu.com/community/VimHowto`
- `Gedit` – `https://help.ubuntu.com/community/gedit`

For simplicity, in this lab we use `nano` text editor to view, create and edit text files.

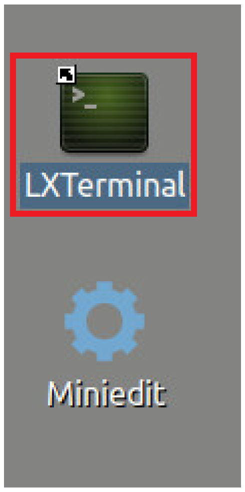
1.2 Creating a shell script

Shell scripts are effective in executing repetitive terminal commands. Unlike executing commands manually in the terminal, scripts can be saved and executed whenever needed simple by invoking their names. We will begin this lab by writing some basic shell scripts.

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



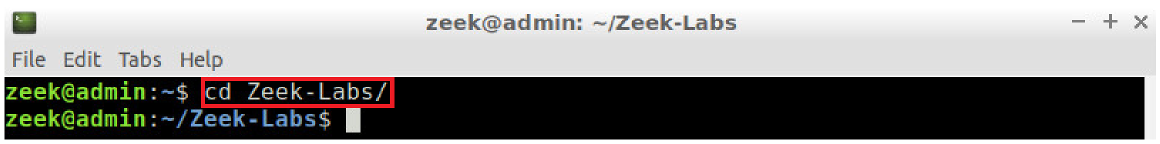
Step 2. The *Client* machine will now open, and the desktop will be displayed. On the left side of the screen, click on the LXTerminal icon as shown below.



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

Step 3. In the Linux terminal, navigate to the lab workspace directory by typing the following command:

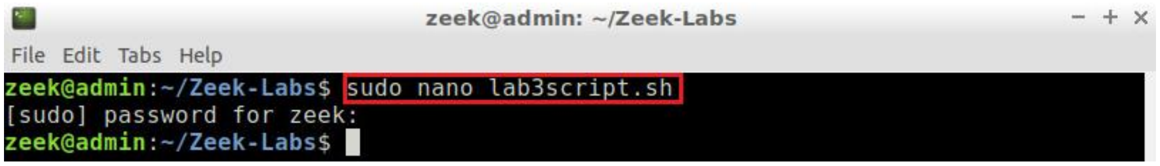
```
cd Zeek-Labs/
```



```
zeek@admin: ~/Zeek-Labs
File Edit Tabs Help
zeek@admin:~$ cd Zeek-Labs/
zeek@admin:~/Zeek-Labs$
```

Step 4. Use the `nano` text editor to create the *lab3script.sh* file.

```
sudo nano lab3script.sh
```

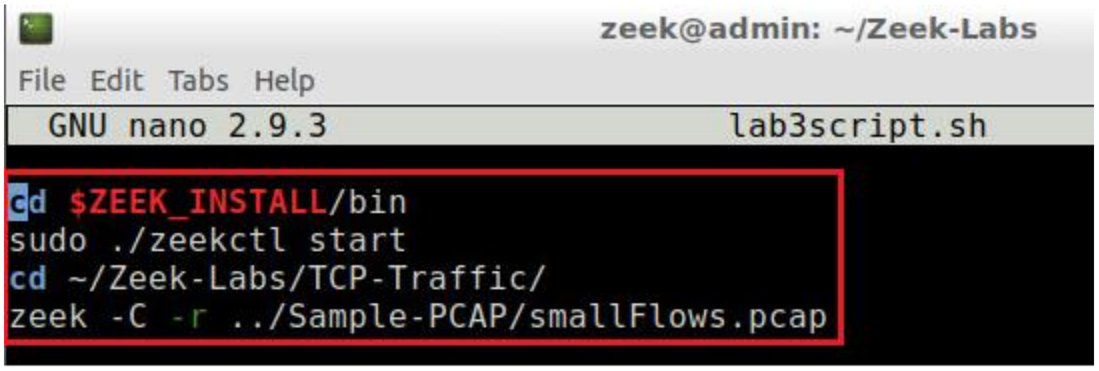


```
zeek@admin:~/Zeek-Labs$ sudo nano lab3script.sh
[sudo] password for zeek:
zeek@admin:~/Zeek-Labs$
```

Step 5. Edit the *lab3script.sh* file contents.

Once the text editor has opened, we will be able to enter the following commands. Each new line will denote a new Terminal command being passed. 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
cd ~/Zeek-Labs/TCP-Traffic/
zeek -C -r ../Sample-PCAP/smallFlows.pcap
```



```
  GNU nano 2.9.3                        lab3script.sh

cd $ZEEK_INSTALL/bin
sudo ./zeekctl start
cd ~/Zeek-Labs/TCP-Traffic/
zeek -C -r ../Sample-PCAP/smallFlows.pcap
```

The file's content is explained as follows:

- Line 1: changes the current directory to the Zeek's installation directory.
- Line 2: starts a new instance of Zeek through `zeekctl`.
- Line 3: changes the current directory to the lab workspace.
- Line 4: invokes the `zeek` command with the `-r` option to begin processing the *smallFlows.pcap* capture file located in the *Sample-PCAP* directory.

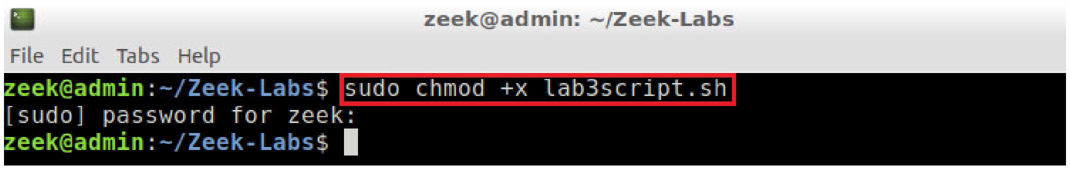
Step 6. When using `nano`, the following keyboard shortcuts are used to save a file and then exit the workspace.

- `CTRL + o` – save the file
- `CTRL + x` – save and exit the file, return to terminal

After completing Step 6 and adding the correct commands with proper formatting, we will save and exit the text editor. Press `CTRL + o` and hit `Enter` to save the file's contents, then `CTRL + x` to exit `nano` and return to the terminal.

Step 7. Use the following command to modify the permissions of the script file to make it executable. When prompted for a password, type `password` and hit `Enter`.

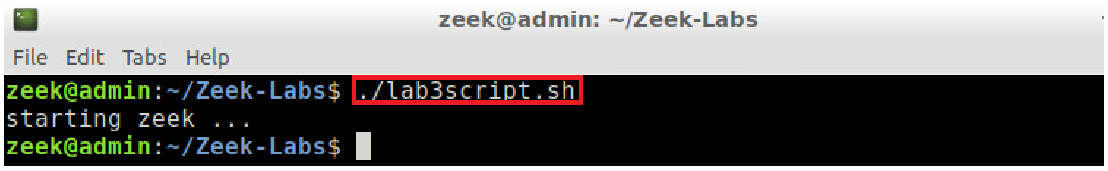
```
sudo chmod +x lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ sudo chmod +x lab3script.sh
[sudo] password for zeek:
zeek@admin:~/Zeek-Labs$
```

Step 8. Execute the *lab3script.sh* shell script by typing the following command.

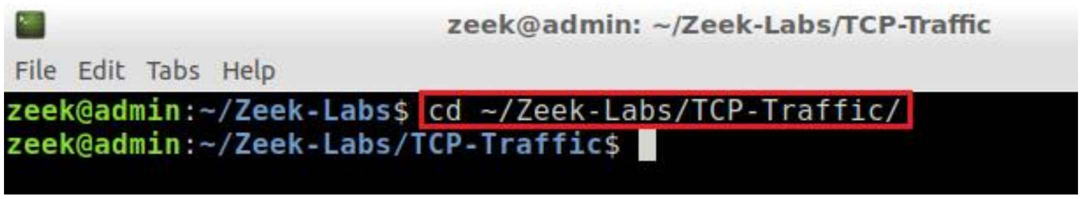
```
./lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ ./lab3script.sh
starting zeek ...
zeek@admin:~/Zeek-Labs$
```

Step 9. Navigate to the lab workspace directory.

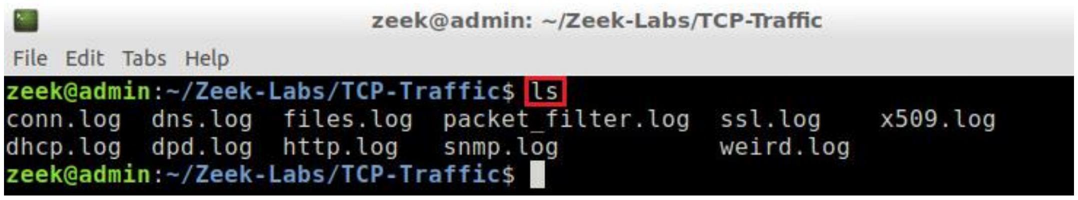
```
cd ~/Zeek-Labs/TCP-Traffic/
```



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

Step 10. Verify that the *smallFlows.pcap* file was processed successfully.

```
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$
```

The above output shows the list of log files generated by Zeek's processing, verifying that the script executed without errors.

2 Advanced zeek-cut log file analysis

This section introduces more advanced `zeek-cut` functionality to analyze packet capture statistics. These statistics can be used for planning and anomaly analysis. For instance, if a single port has been targeted and received a large number of network traffic, it may highlight a possible vulnerability. We can use the `zeek-cut` utility to determine if a host sends an abnormal number of packets to a specific destination and further analyze this event.

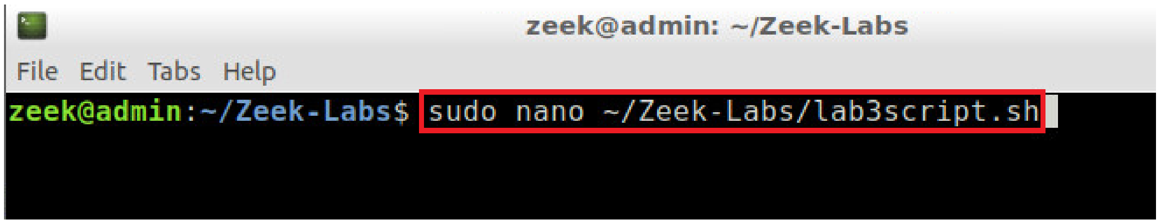
2.1 Example 1

Example 1: Show the 10 source IP addresses that generated the most network traffic, organized in descending order.

To solve this example, we will be looking at the `id.orig_h` column because it contains the source IP addresses from the packet capture file.

Step 1. Open the *lab3script.sh* file with `nano` text editor.

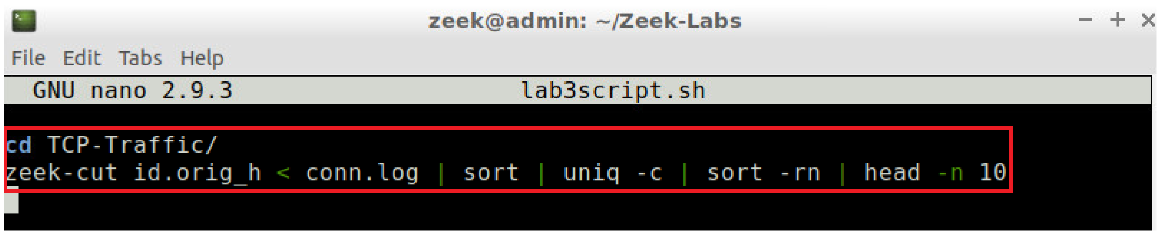
```
sudo nano ~/Zeek-Labs/lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ sudo nano ~/Zeek-Labs/lab3script.sh
```

Step 2. Modify the script file's contents. Delete all the previous content and type the following command:

```
cd TCP-Traffic/
zeek-cut id.orig_h < conn.log | sort | uniq -c | sort -rn | head -n 10
```



```
  GNU nano 2.9.3                    lab3script.sh

cd TCP-Traffic/
zeek-cut id.orig_h < conn.log | sort | uniq -c | sort -rn | head -n 10
```

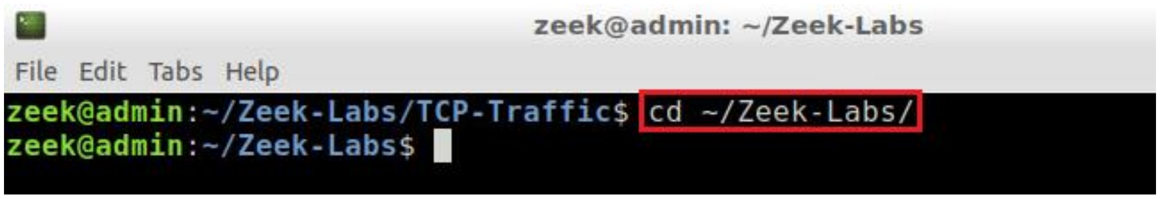
Press `CTRL + o` and hit `Enter` to save the file's contents, then `CTRL + x` to exit `nano` and return to the terminal. The above command is explained as follows:

- `zeek-cut id.orig_h < conn.log`: selects the `id.orig_h` column from the *conn.log* file.
- `| sort`: uses the `sort` command to organize the rows in alphabetical order.
- `| uniq -c`: uses the `uniq` command with the `-c` option to remove duplicates while returning unique instances and their counts.
- `| sort -rn`: uses the `sort` command with the `-rn` option to organize the rows in reverse numerical order.

- `| head -n 10`: uses the `head` command with the `-n` option to display the 10 topmost values.

Step 3. Navigate into *Zeek-Labs* folder by issuing the following command:

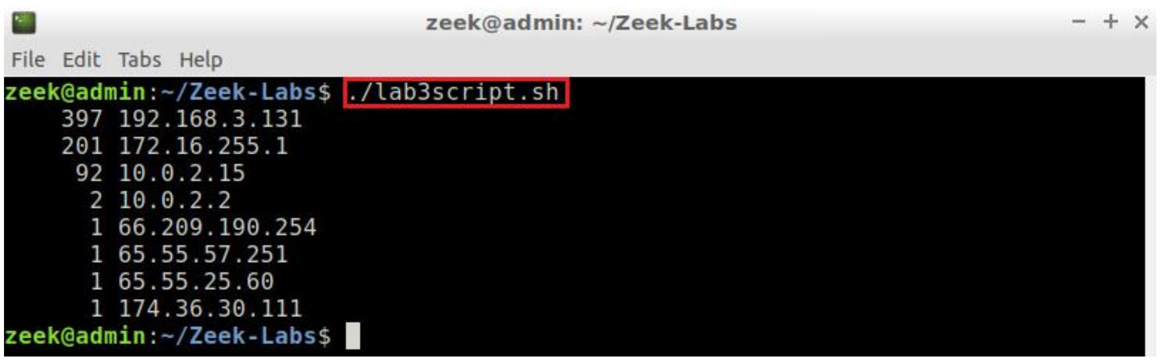
```
cd ..
```



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

Step 4. Execute the modified shell script.

```
./lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ ./lab3script.sh
    397 192.168.3.131
    201 172.16.255.1
     92 10.0.2.15
      2 10.0.2.2
      1 66.209.190.254
      1 65.55.57.251
      1 65.55.25.60
      1 174.36.30.111
zeek@admin:~/Zeek-Labs$
```

The number of duplicates is seen in the left column, while the matching source IP address is seen in the right column. Only 8 unique source addresses were found, and each was returned. From this output, we can conclude that the majority of network traffic was generated by the top 3 source IP addresses.

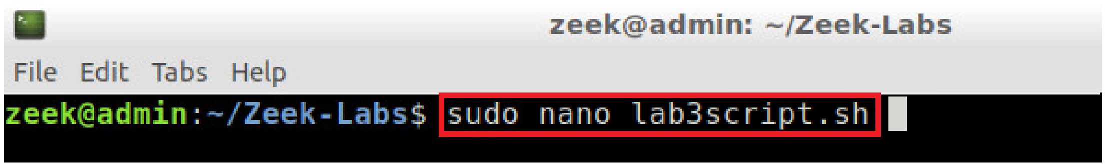
2.2 Example 2

Example 2: Show the 10 destination ports that received the most network traffic, organized in descending order.

To solve this example, we will be looking at the `id.resp_p` column because it contains the destination ports from the packet capture file.

Step 1. Open the *lab3script.sh* file with `nano` text editor.

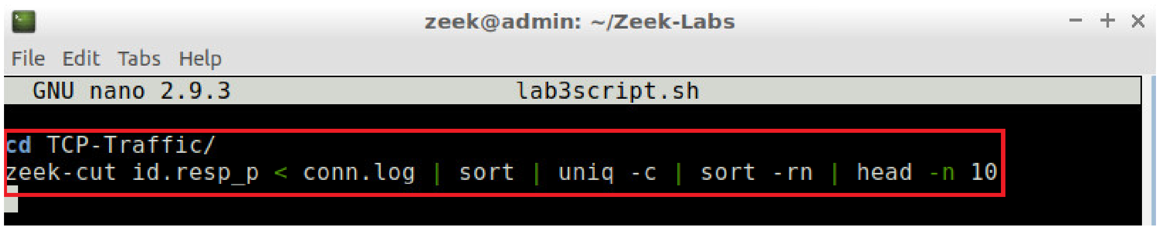
```
sudo nano lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ sudo nano lab3script.sh
```

Step 2. Modify the script file's contents. Delete all the previous content and type the following command:

```
cd TCP-Traffic/
zeek-cut id.resp_p < conn.log | sort | uniq -c | sort -rn | head -n 10
```



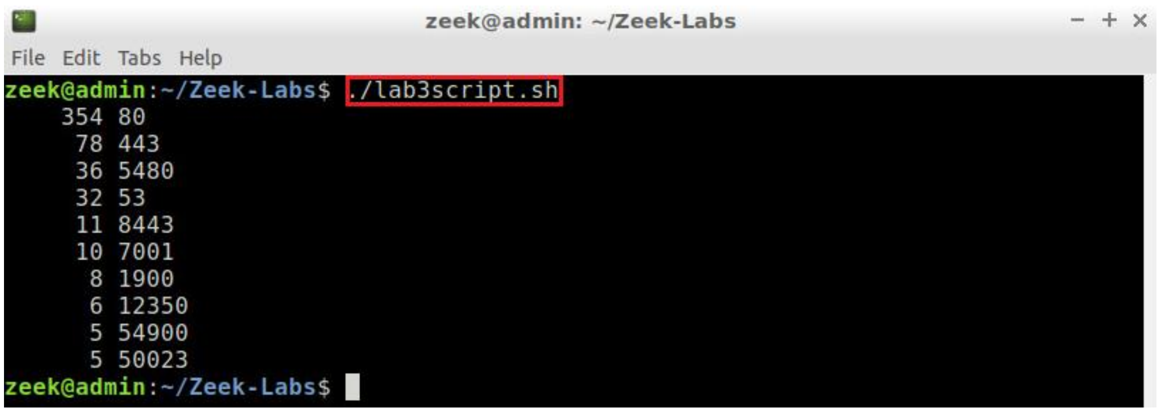
```
cd TCP-Traffic/
zeek-cut id.resp_p < conn.log | sort | uniq -c | sort -rn | head -n 10
```

Press `CTRL + o` and hit `Enter` to save the file's contents, then `CTRL + x` to exit `nano` and return to the terminal. The above command is explained as follows:

- `zeek-cut id.resp_p < conn.log`: selects the `id.resp_p` column from the *conn.log* file.
- `| sort`: uses the `sort` command to organize the rows in alphabetical order.
- `| uniq -c`: uses the `uniq` command with the `-c` option to remove duplicates while returning unique instances and their counts.
- `| sort -rn`: uses the `sort` command with the `-rn` option to organize the rows in reverse numerical order.
- `| head -n 10`: uses the `head` command with the `-n` option to display the 10 topmost values.

Step 3. Execute the modified shell script.

```
./lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ ./lab3script.sh
    354 80
     78 443
     36 5480
     32 53
     11 8443
     10 7001
      8 1900
      6 12350
      5 54900
      5 50023
zeek@admin:~/Zeek-Labs$
```

The number of duplicates is seen in the left column, while the matching destination port is seen in the right column. More than 10 unique destination ports were found, so only the top 10 were returned. From this output we can conclude that port 80 received the most traffic.

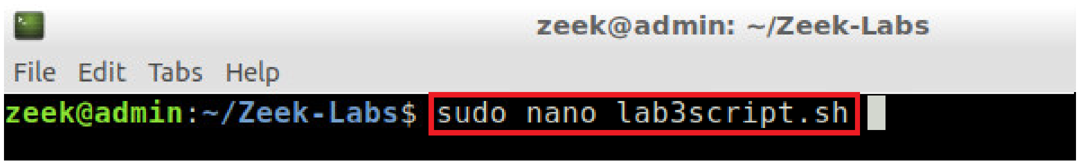
2.3 Example 3

Example 3: Show the number of connections per protocol service.

To solve this example, we will be looking at the `service` column because it contains the destination ports from the packet capture file.

Step 1. Open the *lab3script.sh* file with `nano` text editor.

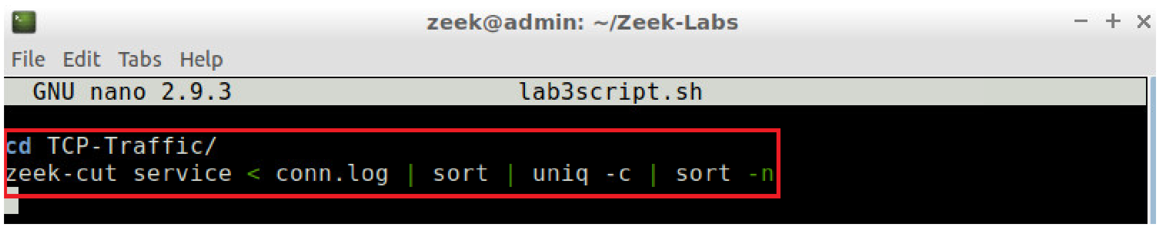
```
sudo nano lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ sudo nano lab3script.sh
```

Step 2. Modify the script file's contents. Delete all the previous content and type the following command:

```
cd TCP-Traffic/
zeek-cut service < conn.log | sort | uniq -c | sort -n
```



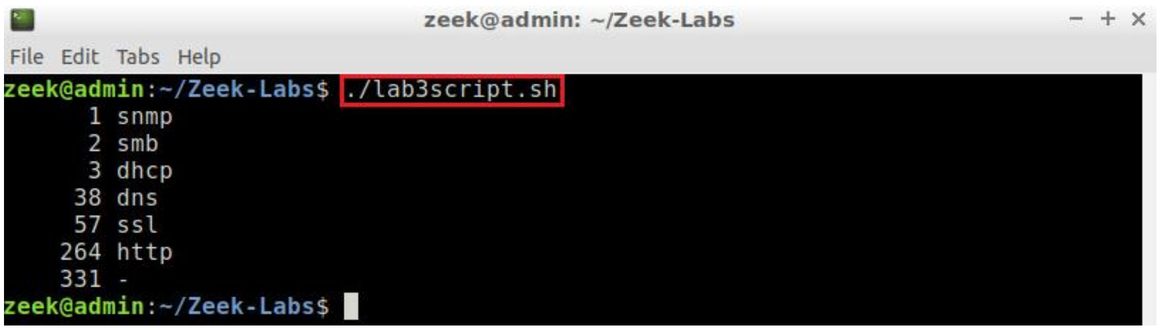
```
GNU nano 2.9.3                    lab3script.sh

cd TCP-Traffic/
zeek-cut service < conn.log | sort | uniq -c | sort -n
```

Press `CTRL + o` and hit `Enter` to save the file's contents, then `CTRL + x` to exit `nano` and return to the terminal. The above command is explained as follows:

- `zeek-cut service < conn.log`: selects the `service` column from the *conn.log* file.
- `| sort`: uses the `sort` command to organize the rows in alphabetical order.
- `| uniq -c`: uses the `uniq` command with the `-c` option to remove duplicates while returning unique instances and their counts.
- `| sort -n`: uses the `sort` command with the `-n` option to organize the rows in numerical order.

Step 3. Execute the modified shell script.

```
./lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ ./lab3script.sh
      1 snmp
      2 smb
      3 dhcp
     38 dns
     57 ssl
    264 http
    331 -
zeek@admin:~/Zeek-Labs$
```

The number of duplicates is seen in the left column, while the matching destination port is seen in the right column. From this output we can see that 331 packets did not have a marked protocol. This can be caused by a number of anomalies and is an example of how you can use the `zeek-cut` utility to return anomalies that require further identification.

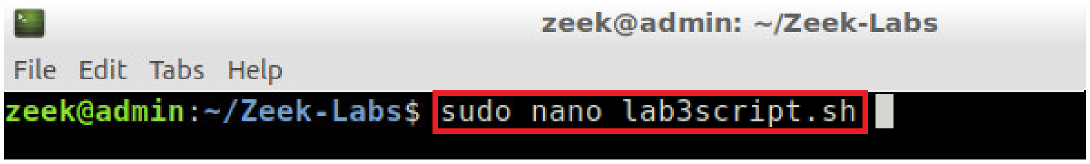
2.4 Example 4

Example 4: Print the distinct browsers used by the hosts in this packet capture file to a separate file.

To solve this example, we will be looking at the `user_agent` column because it contains the browser and connection-related information from the packet capture file.

Step 1. Open the *lab3script.sh* file with `nano` text editor.

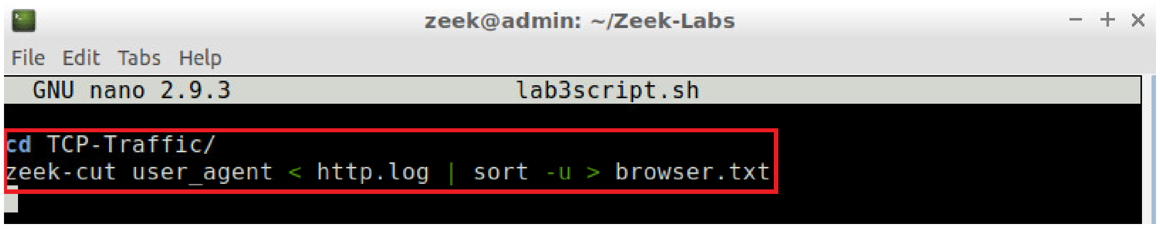
```
sudo nano lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ sudo nano lab3script.sh
```

Step 2. Modify the script file's contents.

```
cd TCP-Traffic/
zeek-cut user_agent < http.log | sort -u > browser.txt
```



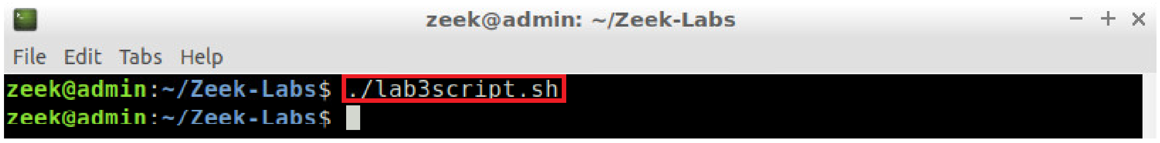
```
  GNU nano 2.9.3                    lab3script.sh

cd TCP-Traffic/
zeek-cut user_agent < http.log | sort -u > browser.txt
```

Press `CTRL + o` and hit `Enter` to save the file's contents, then `CTRL + x` to exit `nano` and return to the terminal. The above command is explained as follows:

- `zeek-cut user_agent < http.log` selects the `user_agent` column from the *http.log* file.
- `| sort -u > browser.txt` uses the `sort` command to sort the lines in the file and the `-u` option checks for strict ordering. The output is then saved into the *browser.txt* file.

Step 3. Execute the modified shell script.

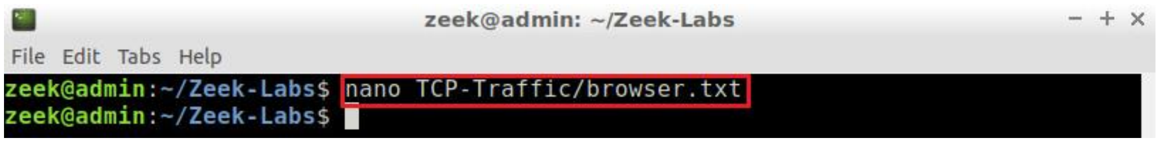
```
./lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ ./lab3script.sh
zeek@admin:~/Zeek-Labs$
```

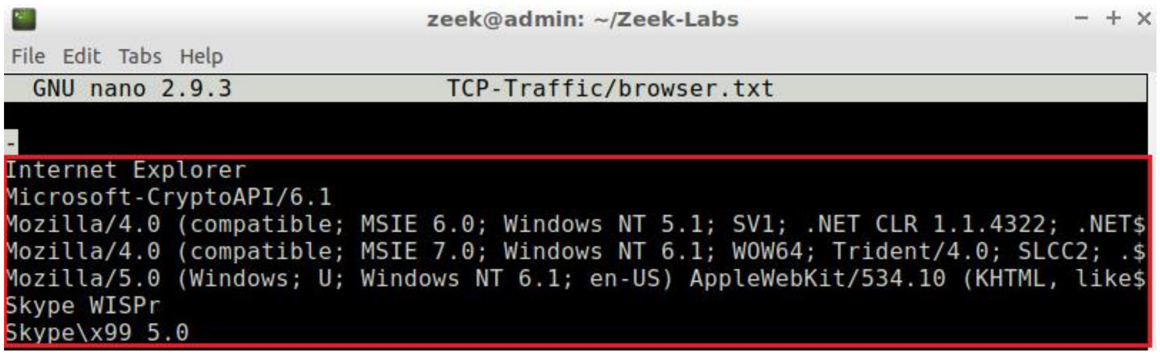
Step 4. Use a text editor to view the contents of the *browser.txt* file.

```
nano TCP-Traffic/browser.txt
```



```
zeek@admin:~/Zeek-Labs$ nano TCP-Traffic/browser.txt
zeek@admin:~/Zeek-Labs$
```

Step 5. View the distinct browser information.



```
  GNU nano 2.9.3                 TCP-Traffic/browser.txt

-
Internet Explorer
Microsoft-CryptoAPI/6.1
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET$
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .$
Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.10 (KHTML, like$
Skype WISPr
Skype\x99 5.0
```

Each browser found within the packet capture file is printed with related information extracted from the traffic by Zeek.

3 Incorporating the AWK scripting language for log file analysis

AWK is a terminal scripting language used to parse, filter and modify text files. AWK is specifically useful when processing rows and columns found in a Comma Separated Value (CSV) file. Additionally, AWK's integrated string manipulation functions allow for the searching and modifying of specific output.

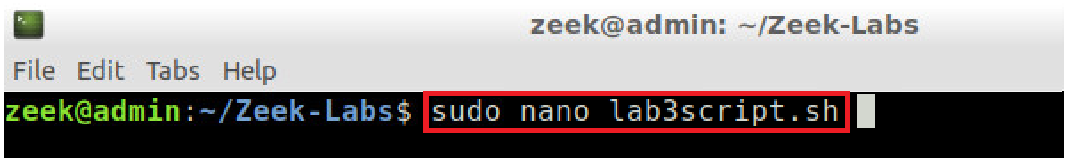
Like `cat` and `head` commands, AWK output can be piped into the `zeek-cut` utility, allowing more advanced parsing and formatting options. AWK reads each column in a file through its position. The first input column is accessed using $1 while the second column is accessed using $2 and so on. AWK also allows creating simple variables to store and read script values. AWK reads the input data as a loop, starting from the top of the file and finishing at the end of the file. Each row is considered an instance within the script.

3.1 Example 1

Example 1: Find the source and destination IP address of all UDP and TCP connections that lasted more than one minute.

Step 1. Open the *lab3script.sh* file with `nano` text editor.

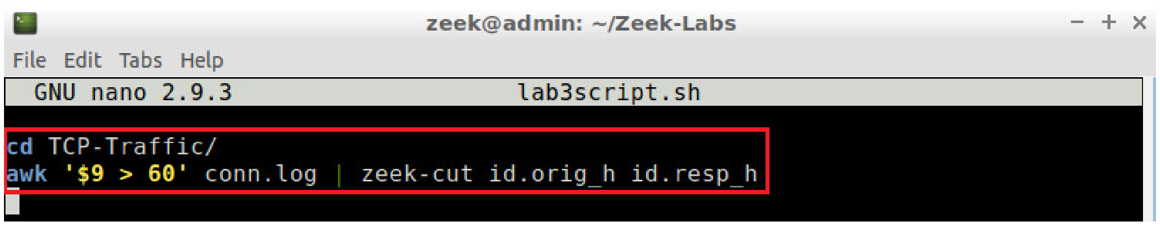
```
sudo nano lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ sudo nano lab3script.sh
```

Step 2. Modify the script file's contents. Delete all the previous content and type the following command:

```
cd TCP-Traffic/
awk '$9 > 60' conn.log | zeek-cut id.orig_h id.resp_h
```



```
  GNU nano 2.9.3                        lab3script.sh

cd TCP-Traffic/
awk '$9 > 60' conn.log | zeek-cut id.orig_h id.resp_h
```

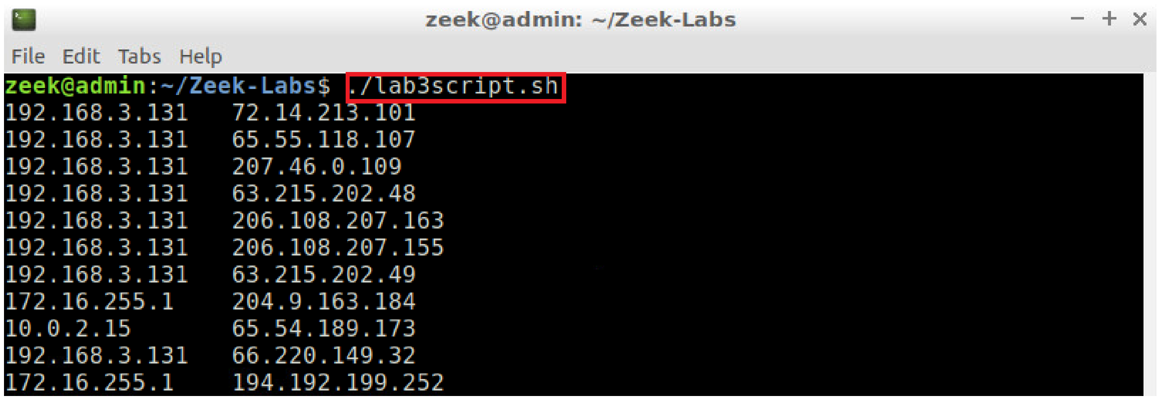
Press `CTRL + o` and hit `Enter` to save the file's contents, then `CTRL + x` to exit `nano` and return to the terminal. The above command is explained as follows:

- `awk '$9 > 60' conn.log` selects the rows that have their 9th column value greater than 60 from the *conn.log* file. The 9th field represents the connection duration, and we are checking if the value is greater than 60 seconds (or 1 minute).

- `| zeek-cut id.orig_h id.resp_h` returns the source and destination IP addresses.

Step 3. Execute the modified shell script.

```
./lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ ./lab3script.sh
192.168.3.131   72.14.213.101
192.168.3.131   65.55.118.107
192.168.3.131   207.46.0.109
192.168.3.131   63.215.202.48
192.168.3.131   206.108.207.163
192.168.3.131   206.108.207.155
192.168.3.131   63.215.202.49
172.16.255.1    204.9.163.184
10.0.2.15       65.54.189.173
192.168.3.131   66.220.149.32
172.16.255.1    194.192.199.252
```

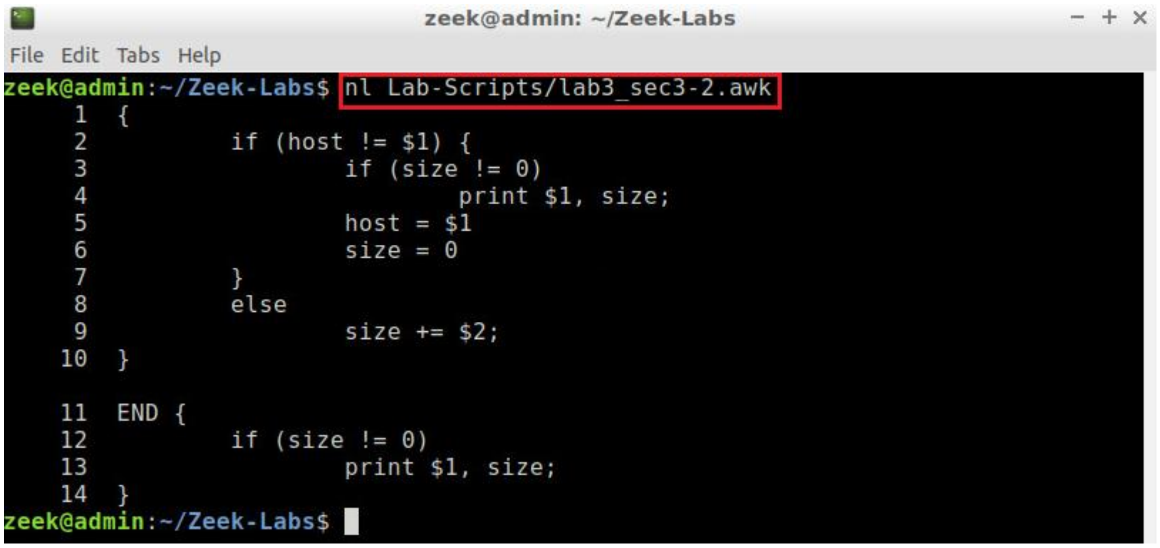
The source IP address is seen in the left column, while the matching destination IP address is seen in the right column. The pairs will only be displayed if the connection lasted at least one minute.

3.2 Example 2

Example 2: Show the top source host addresses in terms of total traffic (in bytes) sent in descending order.

The *Lab-Scripts* directory contains an AWK script named *lab3_sec3-2.awk* that can be viewed with the following command:

```
nl Lab-Scripts/lab3_sec3-2.awk
```



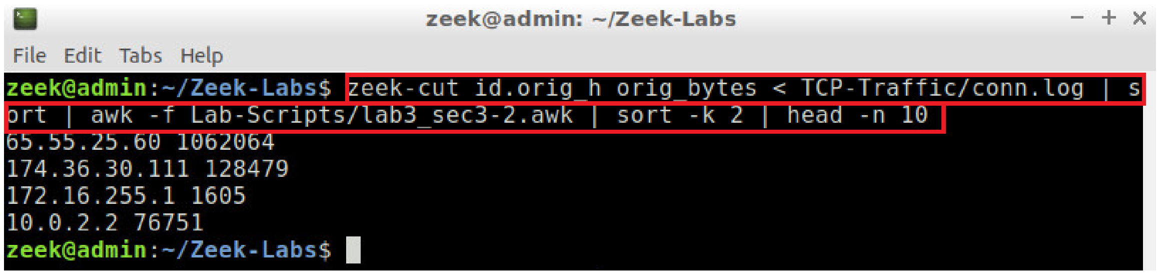
```
zeek@admin:~/Zeek-Labs$ nl Lab-Scripts/lab3_sec3-2.awk
     1  {
     2          if (host != $1) {
     3                  if (size != 0)
     4                          print $1, size;
     5                  host = $1
     6                  size = 0
     7          }
     8          else
     9                  size += $2;
    10  }

    11  END {
    12          if (size != 0)
    13                  print $1, size;
    14  }
zeek@admin:~/Zeek-Labs$
```

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

1. The `{` character is used to begin nested statements. This instance is the main functionality of the script.
2. The host variable, which will be used to store the source IP addresses found in the first column ($1), is checked against the current data entry in the column. If it is not equal, we will enter the next statement. Because we only want one instance of each source IP address, but the summed value of bytes sent, we will use this check to prevent duplicate entries.
3. This line contains a check to make sure the current packet is not empty and does contain a payload. If the current packet contains a payload of more than 0 bytes, we will proceed to line 4.
4. The current source IP address and its byte payload will be printed or returned to the next statements.
5. Now that we know the current source IP address is not yet stored in the host variable, we will create a new entry into the variable.
6. The size variable is reset back to zero
7. The `}` character is used to end nested statements. Therefore, the first case of a source IP address not being contained in host is complete.
8. If the host variable contains the current data entry, we will proceed to line 9.
9. Here we will sum the unique source IP address' total bytes by adding the payload from the second column ($2).
10. The `}` character is used to end nested statements. This is the ending of the main functionality of the script.
11. The `END` statement denotes what the script will do once it has reached the end of the file, and there are no more input data rows to be read.
12. If a source IP address contains a total payload of more than 0 bytes, we will proceed to line 13.
13. AWK will return the source IP address found in the first column, as well as the size variable, containing the total payload in relation to that source IP address.

Step 1. Input the following command.

```
zeek-cut id.orig_h orig_bytes < TCP-Traffic/conn.log | sort | awk -f Lab-
Scripts/lab3_sec3-2.awk   | sort -k 2 | head -n 10
```



```
zeek@admin:~/Zeek-Labs$ zeek-cut id.orig_h orig_bytes < TCP-Traffic/conn.log | s
ort | awk -f Lab-Scripts/lab3_sec3-2.awk | sort -k 2 | head -n 10
65.55.25.60 1062064
174.36.30.111 128479
172.16.255.1 1605
10.0.2.2 76751
zeek@admin:~/Zeek-Labs$
```

- `zeek-cut id.orig_h orig_bytes < conn.log`: selects the `id.orig_h` and `orig_bytes` columns from the *conn.log* file.
- `| sort`: uses the `sort` command to organize the rows in alphabetical order.
- `| awk -f lab3_sec3-2.awk`: will execute awk with the `-f` option to denote using the script find within the *lab3_sec3-2.awk* file.
- `| sort -k 2`: uses the `sort` command with the `-k` option to organize the rows based on the values found in the second column – the total number of bytes.
- `| head -n 10`: uses the `head` command with the `-n` option to display the 10 topmost values.

The left column contains the source IP address, while the right column contains the number of bytes produced by the paired source IP address.

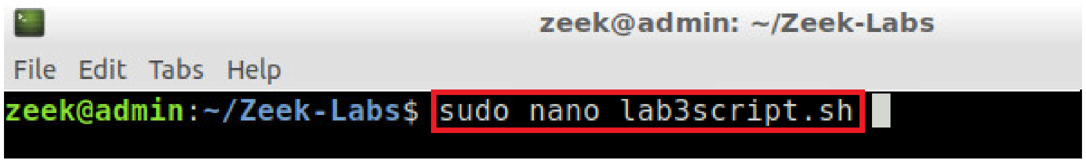
3.3 Example 3

Example 3: Are there any web servers operating on non-standardized ports?

To solve this example, we will be looking at the `service` column to view the packets using the Hyper Text Transport Protocol (HTTP) protocol. The standard ports for the HTTP protocol are 80 and 8080, so we will be searching for the network traffic that does not reach those ports.

Step 1. Open the *lab3script.sh* file with `nano` text editor.

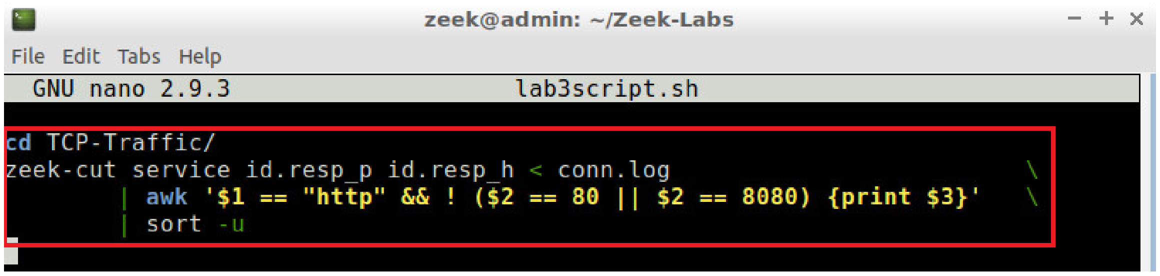
```
sudo nano lab3script.sh
```



```
zeek@admin:~/Zeek-Labs$ sudo nano lab3script.sh
```

Step 2. Modify the script file's contents. Delete all the previous content and type the following command:

```
cd TCP-Traffic/
zeek-cut service id.resp_p id.resp_h < conn.log                              \
      | awk '$1 == "http" && ! ($2 == 80 || $2 == 8080) {print $3}'         \
      | sort -u
```



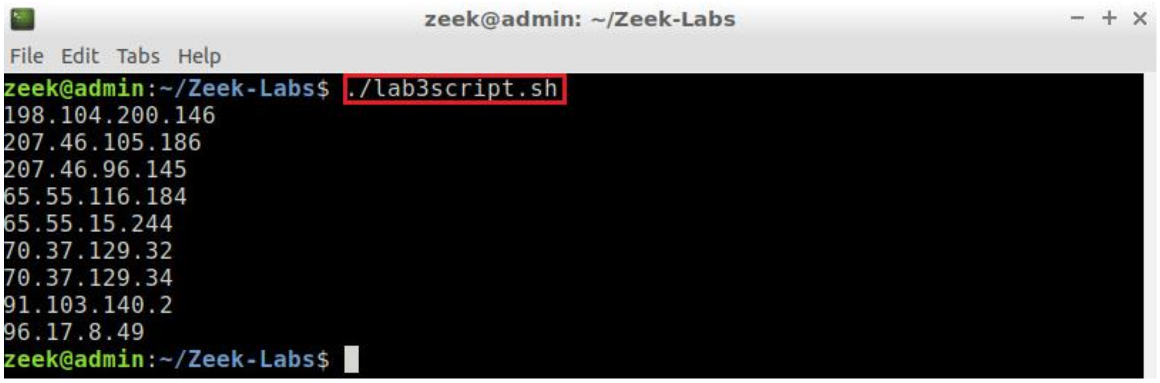
```
zeek@admin: ~/Zeek-Labs
File  Edit  Tabs  Help
  GNU nano 2.9.3                    lab3script.sh

cd TCP-Traffic/
zeek-cut service id.resp_p id.resp_h < conn.log                        \
        | awk '$1 == "http" && ! ($2 == 80 || $2 == 8080) {print $3}'   \
        | sort -u
```

Press `CTRL + o` and hit `Enter` to save the file's contents, then `CTRL + x` to exit `nano` and return to the terminal. The above command is explained as follows:

- `zeek-cut service id.resp_p id.resp_h < conn.log`: selects the `service`, `id.resp_p` and `id.resp_h` columns from the *conn.log* file.
- `| awk`: passes the input into the following AWK command:
    - `$1 == "http"`: performs a check on the first column to make sure the active data entry is running on the http service.
    - `&& ! ($2 == 80 || $2 == 8080)`: performs a second check if the first check is successfully passed. The ports will be checked and if they are not equal to either of the standard http ports (80 and 8080), they will be passed to the print statement
    - `{print $3}`: prints the destination IP address of any host that passes both of the previous checks.
- `| sort -u`: uses the `sort` command to sort the lines in the file and the `-u` option checks for strict ordering.

Step 3. Execute the modified shell script.

```
./lab3script.sh
```



```
zeek@admin: ~/Zeek-Labs
File  Edit  Tabs  Help
zeek@admin:~/Zeek-Labs$ ./lab3script.sh
198.104.200.146
207.46.105.186
207.46.96.145
65.55.116.184
65.55.15.244
70.37.129.32
70.37.129.34
91.103.140.2
96.17.8.49
zeek@admin:~/Zeek-Labs$
```

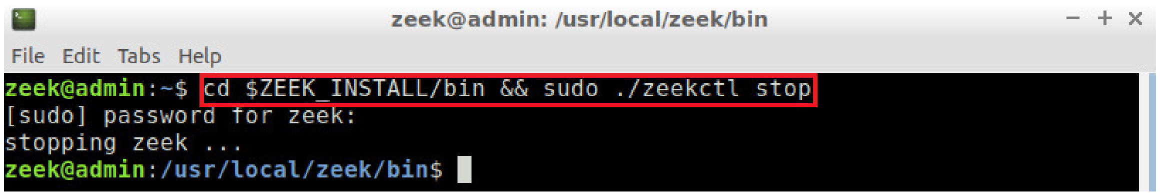
The destination IP addresses that received traffic on non-standardized ports are displayed.

3.4 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
```



```
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 the process of creating shell scripts to be used for network analysis. We introduced more complex commands for the `zeek-control` utility, as well as used the AWK scripting language to retrieve information from Zeek log files.

References

1. "Logging", Zeek user manual, [Online], Available: docs.zeek.org/en/stable/examples/logs.
2. "Exercise: understanding and examining bro logs", Zeek user manual, [Online], Available: https://www.zeek.org/bro-workshop-2011/solutions/logs/index.html.