The University of Texas at San Antonio™

The Cyber Center for Security and Analytics

UNIVERSITY OF SOUTH CAROLINA

# ZEEK INSTRUSION DETECTION SERIES

## Lab 7: Introduction to Zeek Signatures

**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 signature framework language. It introduces what network traffic signatures are and how they are matched to identify specific network events. This lab then reviews premade signature files and provides example usage for analysis.

## Objectives

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

1. Develop signatures using Zeek's signature framework.
2. Analyze processed log files using Zeek signatures.
3. Modify log streams for creating additional events and notices based on signatures.

## 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.

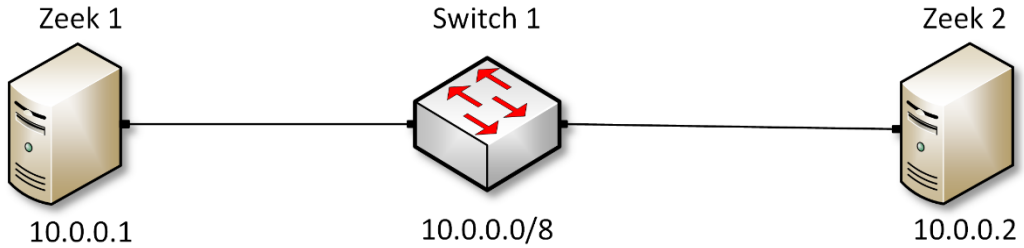



Figure 1. Lab topology.

## 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 signatures.
2. Section 2: Log file analysis using Zeek signatures.
3. Section 3: Modifying Zeek signatures for advanced pattern matching.

## 1 Introduction to Zeek signatures

Following the introduction of developing and implementing basic Zeek scripts, we can now begin generating Zeek signatures. Introduced in the beginning of this lab series, the Zeek event-based engine is the primary architecture for running Zeek as an efficient intrusion detection system. The Zeek event-based engine predominantly utilizes the extensive scripting language to develop policies in order to define the steps and notifications necessary to handle anomalies and exceptions.

However, oftentimes it is simpler to create a predetermined string, known as a signature, and parse packet capture files for the specific signature. Because signatures are used for low-level pattern matching, the Zeek signature framework does not provide the same in-depth functionality as the Zeek scripting language for its event-based engine. Zeek signatures are used to quickly aggregate related network packets through signature matching before analysts can perform further, in-depth analysis on such traffic.

It is important to understand and be familiar with signatures due to their widespread usage across many related Intrusion Detection Systems and application-level firewalls. Separate from Zeek, many alternative IDS, such as the popular *Snort*, rely on signature-based pattern matching for anomaly and malicious event detection. Therefore, in operational cybersecurity environments that analyze network traffic to mitigate and prevent malicious events, understanding Zeek's signature framework adds an additional tool for developing a comprehensive IDS.

This lab will begin by introducing Zeek signatures, detailing their unique file type, how to load them into the Zeek event-based engine, and include a number of examples of leveraging signature matching for log file analysis.

## 1.1 Zeek signature format

The signature below depicts a basic network traffic signature. Depending on their usage, signatures can either include stricter requirements, or be more lax to encompass a larger portion of the processed data.

```
1 signature HTTP-sig {
2     ip-proto == tcp
3     dst-port == 80
4     payload /POST/
5     event "Found HTTP POST!"
6 }
```

1. This line defines a new *signature* object, with the name *HTTP-sig.*
2. Defines the desired match's transport protocol to be TCP.
3. Defines the desired match's destination port to be 80.
4. Defines the desired match's payload to contain the regular expression equivalent to 'POST'.
5. Defines an event if the match is found. Currently, the event will post a "HTTP Packet Found!" message; however, these events can be developed with a more complex functionality if the need arises.

This signature can be loaded into the Zeek signature framework during network traffic analysis, in which Zeek will attempt to match packets with the signature's details. While each individual packet can only be matched one-time, multiple signatures can be applied to any arbitrary data.

Additional signatures and their included variables are outlined and explained in Zeek's official documentation. 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.

```
https://docs.zeek.org/en/current/frameworks/signatures.html
```

## 1.2 Creating and using Zeek signatures

Similar to Zeek's policy scripting framework, Zeek signatures are saved in separate files denoted by the `.sig` file extension. There are three ways to initialize Zeek for network traffic analysis while leveraging the Zeek signature framework:

1. When initializing Zeek from the terminal, include the additional `-s` option:

```
zeek -r <pcap_file_location> -s <signature_file_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.
- `-s`: option signifies to Zeek that the next file contains signatures.
- `<script_location>`: indicates the script location.

2. When creating a Zeek policy script, include the `@load-sigs` directive:

```
@load-sigs

module ZeekScript;

export{
    /* Append and define new log stream parameters */
}
```

3. When creating a Zeek policy script, extend the Zeek global `signature_files` variable by appending the += operator followed by the signature file:

```
@load-sigs

module ZeekScript;

redef signature_files += "signature_file_path.sig"
```

### 1.3 Zeek's default signature framework

This section introduces the default Zeek signature file that is compiled and included after Zeek has been installed.

While this default Zeek script includes scan-based detection, it will not correctly identify every unique anomaly that may be encountered. However, it does provide a comprehensive starter code that can be reviewed and customized to understand the Zeek signature framework.

The default Zeek signature file is named *main.zeek*. More information on this script can be found in Zeek's documentation pages. 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.

```
https://docs.zeek.org/en/current/scripts/base/frameworks/signatures/main.zeek.html
```

The file has been copied into the Zeek lab workspace directory and renamed to *ZeekSignatureFramework.zeek* for ease of access and name-reference clarity.

```
1  type Action: enum {
2      ## Ignore this signature completely (even for scan detection).
3      ## Don't write to the signatures logging stream.
4      SIG_IGNORE,
5      ## Process through the various aggregate techniques, but don't
6      ## report individually and don't write to the signatures logging
7      ## stream.
8      SIG_QUIET,
9      ## Generate a notice.
10     SIG_LOG,
11     ## The same as :zeek:enum:`Signatures::SIG_LOG`, but ignore for
12     ## aggregate/scan processing.
13     SIG_FILE_BUT_NO_SCAN,
14     ## Generate a notice and set it to be alarmed upon.
15     SIG_ALARM,
16     ## Alarm once per originator.
17     SIG_ALARM_PER_ORIG,
18     ## Alarm once and then never again.
19     SIG_ALARM_ONCE,
20     ## Count signatures per responder host and alarm with the
21     ## :zeek:enum:`Signatures::Count_Signature` notice if a threshold
22     ## defined by :zeek:id:`Signatures::count_thresholds` is reached.
23     SIG_COUNT_PER_RESP,
24     ## Don't alarm, but generate per-orig summary.
25     SIG_SUMMARY,
26 };
```

The figure above shows the options for signature match events within the *ZeekSignatureFramework.zeek* file. The options are explained as follows. Each number represents the respective line number:

4. `SIG_IGNORE`: if a signature is matched, do not write to the logging stream.
8. `SIG_QUIET`: if a signature is matched, process the included events but do not write to the logging stream.
10. `SIG_LOG`: if a signature is matched, generate a notice.
13. `SIG_FILE_BUT_NO_SCAN`: if a signature is matched and does not meet scan thresholds, write to the logging stream.
15. `SIG_ALARM`: if a signature is matched, generate a notice and set an alarm.
17. `SIG_ALARM_PER_ORIG`: if a signature is matched, generate a notice and set an alarm once per host that triggered the match.
19. `SIG_ALARM_ONCE`: if a signature is matched, generate a notice and set an alarm only one time, no matter the number of matches.
23. `SIG_COUNT_PER_RESP`: if a signature is matched, create a running count per responder host to compare against developed thresholds to identify and exclude scan traffic.
23. `SIG_SUMMARY`: generate a summary of all matched signatures based on the unique hosts that triggered a signature match.

Additional options and signature-specific events can be created using the Zeek scripting framework. Furthermore, Lab 8 of this series will enumerate upon the aforementioned scan thresholds and how Zeek determines if a host is probing a network.

```
type Info: record {
    ## The network time at which a signature matching type of event
    ## to be logged has occurred.
    ts:         time        &log;
    ## A unique identifier of the connection which triggered the
    ## signature match event.
    uid:        string      &log &optional;
    ## The host which triggered the signature match event.
    src_addr:   addr        &log &optional;
    ## The host port on which the signature-matching activity
    ## occurred.
    src_port:   port        &log &optional;
    ## The destination host which was sent the payload that
    ## triggered the signature match.
    dst_addr:   addr        &log &optional;
    ## The destination host port which was sent the payload that
    ## triggered the signature match.
    dst_port:   port        &log &optional;
    ## Notice associated with signature event.
    note:       Notice::Type &log;
    ## The name of the signature that matched.
    sig_id:     string      &log &optional;
    ## A more descriptive message of the signature-matching event.
    event_msg:  string      &log &optional;
    ## Extracted payload data or extra message.
    sub_msg:    string      &log &optional;
    ## Number of sigs, usually from summary count.
    sig_count:  count       &log &optional;
    ## Number of hosts, from a summary count.
    host_count: count       &log &optional;
};
```

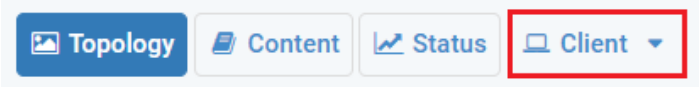
The figure above shows the variables that store signature-specific packet information accessed in the *ZeekSignatureFramework.zeek* file. These variables can be accessed to extract the stored information for notifications and warnings. Furthermore, each variable can be printed to the logging stream, following the Zeek log file format reviewed in previous labs. Each variable is explained by its proceeding comments, denoted by the # character.

## 2 Log file analysis using Zeek signatures

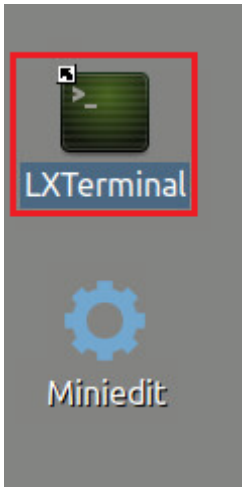
With Zeek's signature framework, we can create specific pattern-based signature filters to be applied during packet capture analysis. This section shows example signatures and their usage for network analysis.

### 2.1 Starting a new instance of Zeek

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

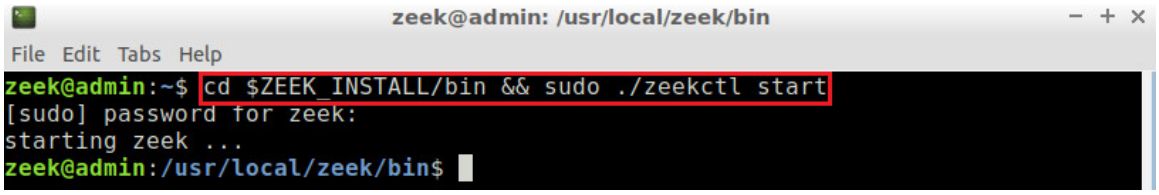



**Step 2.** The *Client* machine will now open, and the desktop will be displayed. On the left side of the screen, click on the LXTerminal icon as shown below.




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

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




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

## 2.2 Viewing a premade Zeek signature file

**Step 1.** Navigate to the *Lab-Scripts* directory.

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

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

**Step 2:** Display the contents of the *lab7_sec2-2.sig* file using `nl`.

```
nl lab7_sec2-2.sig
```

```
zeek@admin: ~/Zeek-Labs/Lab-Scripts
File  Edit  Tabs  Help
zeek@admin:~/Zeek-Labs/Lab-Scripts$ nl lab7_sec2-2.sig
     1  signature HTTP-POST-sig{
     2          ip-proto == tcp
     3          dst-port == 80
     4          payload /POST/
     5          event "Found HTTP Post"
     6  }

     7  signature HTTP-GET-sig{
     8          ip-proto == tcp
     9          dst-port == 80
    10          payload /GET/
    11          event "Found HTTP Request"
    12  }
zeek@admin:~/Zeek-Labs/Lab-Scripts$
```

This signature file contains two signatures to be matched during network traffic analysis and is explained as follows. Each number represents the respective line number:

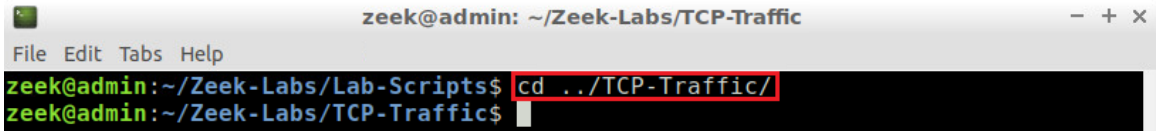
1. This line defines a new *signature* object, with the name *HTTP-POST-sig.*
2. Defines the desired match's transport protocol to be TCP.
3. Defines the desired match's destination port to be 80.
4. Defines the desired match's payload to contain the regular expression equivalent to 'POST'.
5. Defines an event if the match is found. Currently, the event will post a "Found HTTP Post" message.

7. This line defines a new *signature* object, with the name *HTTP-GET-sig.*
8. Defines the desired match's transport protocol to be TCP.
9. Defines the desired match's destination port to be 80.
10. Defines the desired match's payload to contain the regular expression equivalent to 'GET'.
11. Defines an event if the match is found. Currently, the event will post a "Found HTTP Request" message.

## 2.3 Executing the premade Zeek signature file

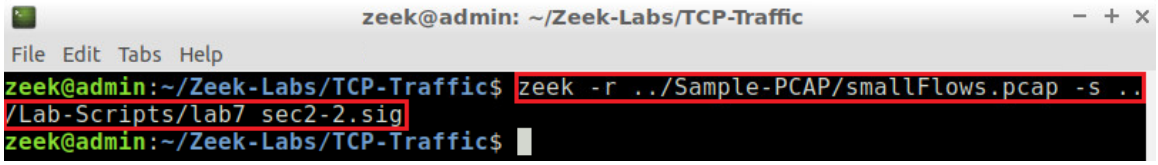
**Step 1.** Navigate to the *TCP-Traffic* directory.

```
cd ../TCP-Traffic/
```



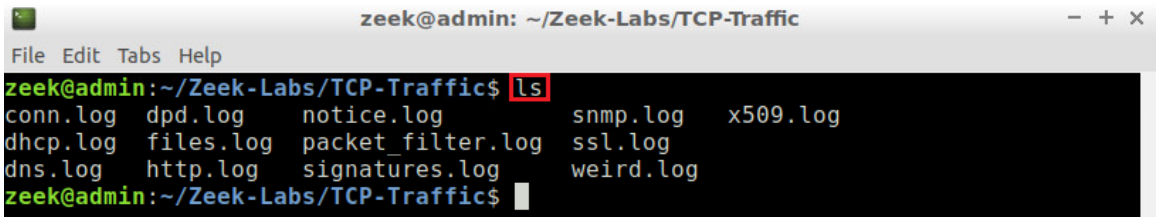

**Step 2.** Process the *smallFlows.pcap* packet capture file using the signature file *lab7_sec2-2.sig*. It is possible to use the `tab` key to autocomplete the longer paths.

```
zeek -r ../Sample-PCAP/smallFlows.pcap -s ../Lab-Scripts/lab7_sec2-2.sig
```




**Step 3.** List the generated log files in the current directory.

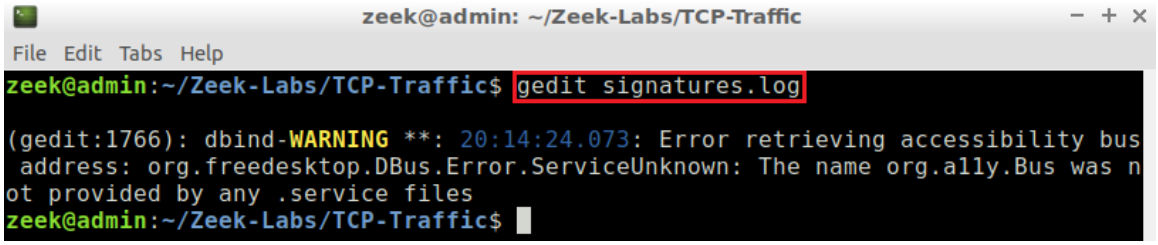
```
ls
```




A new log file that has not been previously introduced is now displayed: *signatures.log*. This log file will contain all signature matches and their corresponding events and notices.

**Step 4.** View the contents of the *signatures.log* file using the `gedit` text editor.

```
gedit signatures.log
```



```
signatures.log
~/Zeek-Labs/TCP-Traffic
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   signatures
#open   2020-03-14-20-13-17
#fields ts      uid     src_addr        src_port        dst_addr        dst_port        note    sig_id  event_msg       sub_msg
sig_count       host_count
#types  time    string  addr    port    addr    port    enum    string  string  string  count   count
1295981542.484409       CQis0DXT6RApT9FHh       192.168.3.131   57011   72.14.213.138   80      Signatures::Sensitive_Signature HTTP-GET-
sig     192.168.3.131: Found HTTP Request       GET /complete/search?client=chrome&hl=en-US&q=cr HTTP/1.1\x0d\x0aHost:
clients1.google.ca\x0d\x0aConnection: keep-alive\x0d\x0aUser-Agent: Mozilla/5.0 (Window...     -       -
1295981542.727459       C713r43nGTMxJoBpug      192.168.3.131   55950   72.14.213.102   80      Signatures::Sensitive_Signature HTTP-GET-
sig     192.168.3.131: Found HTTP Request       GET /complete/search?client=chrome&hl=en-US&q=msn HTTP/1.1\x0d\x0aHost:
clients1.google.ca\x0d\x0aConnection: keep-alive\x0d\x0aUser-Agent: Mozilla/5.0 (Windo...     -       -
1295981543.182793       C8HaSy3s9Hz4gRWETb      192.168.3.131   55953   65.55.206.209   80      Signatures::Sensitive_Signature HTTP-GET-
sig     192.168.3.131: Found HTTP Request       GET / HTTP/1.1\x0d\x0aHost: msn.ca\x0d\x0aConnection: keep-alive\x0d\x0aAccept: application/
xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/p...   -       -
1295981543.354015       Ci0CuG4zRGE0BYRRC5      192.168.3.131   55954   65.55.17.37     80      Signatures::Sensitive_Signature HTTP-GET-
sig     192.168.3.131: Found HTTP Request       GET / HTTP/1.1\x0d\x0aHost: ca.msn.com\x0d\x0aConnection: keep-alive\x0d\x0aAccept:
application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,ima...   -       -
1295981543.474636       CY51nn32umCReIiRJ6      192.168.3.131   55955   207.46.148.38   80      Signatures::Sensitive_Signature HTTP-GET-
sig     192.168.3.131: Found HTTP Request       GET /action/MMN_Homepage HTTP/1.1\x0d\x0aHost: view.atdmt.com\x0d\x0aConnection: keep-
alive\x0d\x0aReferer: http://ca.msn.com/\x0d\x0aAccept: */*\x0d\x0aUser-Agent: Mozil... -       -
1295981543.474636       -       192.168.3.131   -       -       -       Signatures::Multiple_Sig_Responders      HTTP-GET-sig    Found HTTP
Request 192.168.3.131 has triggered signature HTTP-GET-sig on 5 hosts   -       5
1295981543.528088       CGs8Lp42K5AcPyHNC9      192.168.3.131   55956   66.235.139.121  80      Signatures::Sensitive_Signature HTTP-GET-
sig     192.168.3.131: Found HTTP Request       GET /b/ss/msnportalhomepagecaen/1/H.7-pdv-2/s84495919100008?
[AQB]&ndh=1&t=25%2F0%2F2011%2010%3A52%3A23%202%20480&ns=msnportal&pageName=MSN%2...     -       -
1295981543.536159       CGP8riRzMRKnZYHH5       192.168.3.131   55957   65.55.5.232     80      Signatures::Sensitive_Signature HTTP-GET-
sig     192.168.3.131: Found HTTP Request       GET /ADSAdClient31.dll?GetSAd=&DPJS=0&PN=MSFT&PG=CAE9TX&AP=1389 HTTP/1.1\x0d\x0aHost:
rad.msn.com\x0d\x0aConnection: keep-alive\x0d\x0aReferer: http://ca.msn....     -       -
```

The file is explained as follows:

- The red box indicates the name of the signature that was matched.
- The orange box indicates the event or message that was included when defining the signature.
- The blue box indicates the packet payload that was matched against the input signatures.

**Step 5.** Click the ☒ mark to close the gedit window. Clear the contents of the TCP-Traffic directory.

```
./../Lab-Scripts/lab_clean.sh
```

```
zeek@admin: ~/Zeek-Labs/TCP-Traffic
File  Edit  Tabs  Help
zeek@admin:~/Zeek-Labs/TCP-Traffic$ ./../Lab-Scripts/lab_clean.sh
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

## 3 Executing Zeek signature matching for network traffic analysis

This section modifies the existing signature file to generate additional signature events and notices. We will be modifying the previous signatures from TCP-based HTTP messages to UDP-based SNMP and DNS messages.

### 3.1 Modifying the premade Zeek signature file

**Step 1.** View the contents of the *lab7_sec3-1.sig* file using `nl`.

```
nl ../Lab-Scripts/lab7_sec3-1.sig
```

```
zeek@admin: ~/Zeek-Labs/Lab-Scripts
File  Edit  Tabs  Help
zeek@admin:~/Zeek-Labs/Lab-Scripts$ nl ../Lab-Scripts/lab7_sec3-1.sig
     1  signature SNMP-REQUEST-sig{
     2          ip-proto == udp
     3          dst-port == 161
     4          event "Found SNMP Request"
     5  }

     6  signature SNMP-RESPONSE-sig{
     7          ip-proto == udp
     8          dst-port == 52400
     9          event "Found SNMP Response"
    10  }

    11  signature DNS-REQUEST-sig{
    12          ip-proto == udp
    13          dst-port == 53
    14          event "Found DNS Request"
    15  }
zeek@admin:~/Zeek-Labs/Lab-Scripts$
```

**Step 2.** Open the *lab7_sec3-1.sig* file with the `gedit` text editor.

```
gedit ../Lab-Scripts/lab7_sec3-1.sig
```

```
zeek@admin: ~/Zeek-Labs/TCP-Traffic
File  Edit  Tabs  Help
zeek@admin:~/Zeek-Labs/TCP-Traffic$ gedit ../Lab-Scripts/lab7_sec3-1.sig

(gedit:1904): dbind-WARNING **: 20:20:33.292: Error retrieving accessibility bus
 address: org.freedesktop.DBus.Error.ServiceUnknown: The name org.a11y.Bus was n
ot provided by any .service files

(gedit:1904): Gtk-WARNING **: 20:20:33.341: Attempting to read the recently used
 resources file at '/home/zeek/.local/share/recently-used.xbel', but the parser
failed: Failed to open file "/home/zeek/.local/share/recently-used.xbel": Permis
sion denied.
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

**Step 3.** Update the *lab7_sec3-1.sig* file to include the following signatures. Then, close out the gedit once finish editing.

```
signature SNMP-REQUEST-sig{
      ip-proto == udp
      dst-port == 161
      event "Found SNMP Request"
}
signature SNMP-RESPONSE-sig{
      ip-proto == udp
      dst-port == 52400
      event "Found SNMP Response"
}
signature DNS-REQUEST-sig{
      ip-proto == udp
      dst-port == 53
      event "Found DNS Request"
}
```

```
signature SNMP-REQUEST-sig{
        ip-proto == udp
        dst-port == 161
        event "Found SNMP Request"
}

signature SNMP-RESPONSE-sig{
        ip-proto == udp
        dst-port == 52400
        event "Found SNMP Response"
}

signature DNS-REQUEST-sig{
        ip-proto == udp
        dst-port == 53
        event "Found DNS Request"
}
```

## 3.2 Executing the updated Zeek signature file

**Step 1.** Process the *smallFlows.pcap* packet capture file using the signature file *lab7_sec3-1.sig*. It is possible to use the `tab` key to autocomplete the longer paths.

```
zeek -r ../Sample-PCAP/smallFlows.pcap -s ../Lab-Scripts/lab7_sec3-1.sig
```

```
zeek@admin:~/Zeek-Labs/TCP-Traffic$ zeek -r ../Sample-PCAP/smallFlows.pcap -s ..
/Lab-Scripts/lab7_sec3-1.sig
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

**Step 2.** List the generated log files in the current directory.

```
ls
```

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

The *signatures.log* file has been recreated and will contain the newly updated signature matches.

**Step 3.** View the contents of the *signatures.log* file using the `gedit` text editor. Then, close out the gedit once finish examining the new file content.

```
gedit signatures.log
```

```
zeek@admin: ~/Zeek-Labs/TCP-Traffic
File  Edit  Tabs  Help
zeek@admin:~/Zeek-Labs/TCP-Traffic$ gedit signatures.log

(gedit:1442): dbind-WARNING **: 14:44:48.194: Error retrieving accessibility bus
 address: org.freedesktop.DBus.Error.ServiceUnknown: The name org.a11y.Bus was n
ot provided by any .service files
zeek@admin:~/Zeek-Labs/TCP-Traffic$
```

```
Open ▾   signatures.log  ~/Zeek-Labs/TCP-Traffic   Save
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   signatures
#open   2020-03-15-14-42-18
#fields ts      uid     src_addr        src_port        dst_addr        dst_port        note    sig_id  event_msg       sub_msg
sig_count       host_count
#types  time    string  addr    port    addr    port    enum    string  string  string  count   count
1295981655.843173       Ch4Ifz23dedK6z0QMf      10.0.2.15       49796   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981655.926096       C6htqQ2iBqdCN3tAE5      10.0.2.15       50559   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981658.781806       CTzqJp1sof9eUhmMpa      10.0.2.15       54657   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981658.854004       C5vx411tVLQaGd7DI5      10.0.2.15       57524   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981659.567918       CyUWNQ14mIGNAM7j3j      10.0.2.15       54795   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981659.783932       Cqh2xI1SLGXl20wJt4      10.0.2.15       61870   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981660.144937       CEIl9BpY4uXc0D1ka       10.0.2.15       64982   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981663.533829       CBY6BO2rdUV0XEIbI6      10.0.2.15       57632   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981664.266166       C5ovQAF61YGPi6Hic       10.0.2.15       62310   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981664.492158       C9TKqM7eY4clREqyg       10.0.2.15       59794   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981665.894416       Cv38BD3vTNI7rARZue      10.0.2.15       58511   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981668.205083       Cn6chsp1hTmXLqTzc       10.0.2.15       58971   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981685.227252       CXkQnx1TuS8bqIRIma      10.0.2.15       59686   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981696.667788       CONHMjiCbbAgLibDa       10.0.2.15       61133   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981711.656223       CLWqCIAPbkM3BRCSg       10.0.2.15       59365   10.0.2.3        53      Signatures::Sensitive_Signature DNS-REQUEST-
sig     10.0.2.15: Found DNS Request    (empty) -       -
1295981744.511002       CZx0Ux3gaud2WgVwxg      192.168.3.131   52400   192.168.3.99    161     Signatures::Sensitive_Signature SNMP-
REQUEST-sig     192.168.3.131: Found SNMP Request       (empty) -       -
1295981744.570907       CZx0Ux3gaud2WgVwxg      192.168.3.99    161     192.168.3.131   52400   Signatures::Sensitive_Signature SNMP-
RESPONSE-sig    192.168.3.99: Found SNMP Response       (empty) -       -
```

The file is explained as follows:

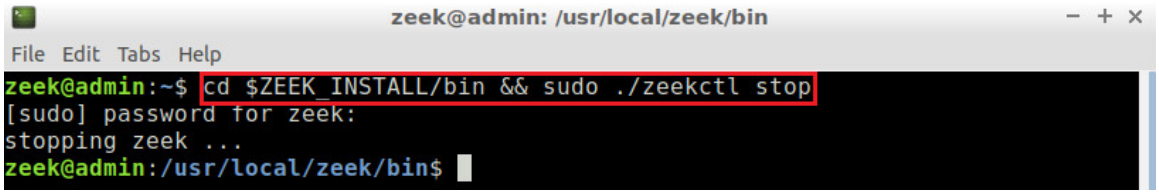
- The red box indicates the DNS-REQUEST-sig signature match as well as the triggered IP address and event message.
- The orange box indicates the SNMP-REQUEST-sig signature match as well as the triggered IP address and event message.
- The blue box indicates the SNMP-RESPONSE-sig signature match as well as the triggered IP address and event message.

## 3.3 Closing the current instance of Zeek

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

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

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




Concluding this lab, we have introduced the Zeek signature framework. Leveraging pattern matching, Zeek signatures can be used to quickly discover packets that follow predetermined formats, while employing a low-level framework for generating warnings and notifications.

## References


1. "Signature framework", Zeek user manual, [Online], Available: https://docs.zeek.org/en/stable/frameworks/signatures.html
2. "Logging framework", Zeek user manual, [Online], Available: https://docs.zeek.org/en/stable/frameworks/logging.html#streams
3. "Monitoring HTTP traffic", Zeek user manual, [Online], Available: https://docs.zeek.org/en/stable/examples/httpmonitor/
4. "Writing scripts", Zeek user manual, [Online], Available: https://docs.zeek.org/en/stable/examples/scripting/#the-event-queue-and-event-handlers.