ZEEK INTRUSION DETECTION SERIES

Lab 10: Application of the Zeek IDS for Real-Time Network Protection

Document Version: 03-15-2020

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

Overview .......................................................................................................................... 3
Objectives ....................................................................................................................... 3
Lab topology .................................................................................................................. 3
Lab settings .................................................................................................................... 3
Lab roadmap ................................................................................................................... 4

1 Introduction to real-time network traffic analysis using Zeek ........................................ 4
   1.1 Starting a new instance of Zeek ............................................................................ 4
   1.2 Launching Mininet ............................................................................................. 5
   1.3 Setting up the zeek1 virtual machine for live network capture .......................... 7
   1.4 Using the zeek2 virtual machine for network scanning activities ...................... 8
      1.4.1 Terminating live network capture ............................................................... 9
   1.5 Analyzing the generated Zeek log files ............................................................. 10

2 Introduction to the Zeek NetControl framework ......................................................... 12
   2.1 Viewing Zeek NetControl within a script file .................................................... 12
   2.2 Executing Zeek NetControl within a script file ................................................. 16

3 Identifying SSH attacks by leveraging the Zeek NetControl framework ................... 20
   3.1 Closing the current instance of Zeek ................................................................. 22

References ...................................................................................................................... 23
Overview

This lab introduces Zeek’s real-time packet analysis for intrusion prevention. By combining the various Zeek-specific events that were introduced and reviewed in previous labs, we are able to identify and mitigate malicious traffic in real-time.

Objectives

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

1. Run Zeek in live mode to process network traffic on the wire.
2. Understand the Zeek NetControl framework.
3. Leverage advanced Zeek scripts for anomaly event detection.

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.

Lab settings

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

<table>
<thead>
<tr>
<th>Device</th>
<th>Account</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>admin</td>
<td>password</td>
</tr>
</tbody>
</table>
### Table 2. Shell variables and their corresponding absolute paths.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Absolute Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ZEEK_INSTALL</td>
<td>/usr/local/zeek</td>
</tr>
<tr>
<td>$ZEEK_TESTING_TRACES</td>
<td>/home/zeek/zeek/testing/btest/Traces</td>
</tr>
<tr>
<td>$ZEEK_PROTOCOLS_SCRIPT</td>
<td>/home/zeek/zeek/scripts/policy/protocols</td>
</tr>
</tbody>
</table>

**Lab roadmap**

This lab is organized as follows:

1. Section 1: Introduction to real-time network traffic analysis using Zeek.
2. Section 2: Introduction to the Zeek NetControl framework.
3. Section 3: Identifying SSH attacks by leveraging the Zeek NetControl framework.

# 1 Introduction to real-time network traffic analysis using Zeek

The previous labs within this lab series have leveraged the `tcpdump` terminal utility for capturing network traffic and generating packet capture files. However, Zeek is capable of collecting and analyzing such network traffic in real-time, with the ability to apply signature-matching and event-based Zeek scripts for malicious event detection.

This section will introduce leveraging Zeek for real-time network traffic analysis, without needing to save the packets captured by the receiving interface.

In contrast to the previous terminal option used for offline packet analysis, this lab will be using the terminal option to indicate the receiving interface for real-time network traffic analysis.

## 1.1 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.

![Client button](image)

**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 \texttt{zeekctl} tool to start a new instance. To type capital letters, it is recommended to hold the \texttt{Shift} key while typing rather than using the \texttt{Caps} key. When prompted for a password, type \texttt{password} and hit \texttt{Enter}.

\texttt{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.

1.2 Launching Mininet

Step 1. From the \textit{Client} machine’s desktop, on the left side of the screen, click on the MiniEdit icon as shown below. When prompted for a password, type \texttt{password} and hit \texttt{Enter}. The MiniEdit editor will now launch.
Step 2. The MiniEdit editor will now launch and allow for the creation of new, virtualized lab topologies. Load the correct topology by clicking the **Open** button within the **File** tab on the top left of the MiniEdit editor.

![MiniEdit File Menu]

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

![Open Directory Window]

Step 4. Select the *Topology.mn* file by double clicking the *Topologies.mn* icon, or by clicking the **Open** button.
Step 5. To begin running the virtual machines, navigate to the **Run** button, found on the bottom left of the Miniedit editor, and select the **Run** button, as seen in the image below.

1.3 Setting up the zeek1 virtual machine for live network capture

Step 1. Launch the **zeek1** terminal by holding the right mouse button on the desired machine and clicking the **Terminal** button.
Step 2. Navigate to the TCP-Traffic directory.

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

Step 3. Start an instance of Zeek live packet capture on interface `zeek1-eth0` while applying the advanced Zeek script `ZeekDetectScans.zeek`. It is possible to use the `tab` key to autocomplete the longer paths.

```
zeek -C -i zeek1-eth0 ../Lab-Scripts/ZeekDetectScans.zeek
```

The `ZeekDetectScans.zeek` scripting file was introduced in Lab 8 of this lab series and will be used by the Zeek event-based engine to identify scan-based traffic. During live network traffic analysis, alternative scripts and signature files can be leveraged to identify specific anomalies and malicious attacks.

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

1.4 Using the `zeek2` virtual machine for network scanning activities

In this section we use the `nmap` software to generate TCP-based scan traffic in order to trigger Zeek’s logging notices.

Step 1. Minimize the `zeek1` Terminal and open the `zeek2` Terminal by following the previous steps. If necessary, right click within the Miniedit editor to activate your cursor.
Step 2. Launch a fragmented TCP scan against the `zeek1` machine.

```
nmap -sT 10.0.0.1
```

Now that we have generated scan-based traffic, we can verify that Zeek was able to identify such malicious events in real-time, while generating corresponding log files.

### 1.4.1 Terminating live network capture

**Step 1.** Minimize the `zeek2` Terminal and open the `zeek1` Terminal using the navigation bar at the bottom of the screen. If necessary, right click within the Miniedit editor to activate your cursor.
**Step 2.** Use the `Ctrl+c` key combination to stop live traffic capture. Statistics of the capture session will be displayed. 2002 packets were recorded by the interface, which were continually analyzed by the Zeek event-based engine.

![Image of Zeek terminal with captured packets](image)

Within the previous image, the red box denotes the live capture command while the orange box indicates the number of packets received on the `zeek1-eth0` interface. 2002 packets were generated by the `zeek2` virtual machine, and no packets were dropped during analysis.

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

![Image of MiniEdit editor](image)

### 1.5 Analyzing the generated Zeek log files

To verify the success of our real time application of Zeek’s event-based engine, we will return to the **Client** machine.

**Step 1.** On the left side of the **Client** desktop, click on the LXTerminal icon as shown below.
Step 2. Navigate to the TCP-Traffic directory to find the log files.

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

Step 3. View the file contents of the TCP-Traffic directory to ensure that Zeek generated log files based on the real-time network traffic analysis.

```shell
ls
```

A number of log files have been generated, specifically, the notice.log file which will contain the event’s triggered by the ZeekDetectScans.zeek script file.

Step 4. View the file contents of the notice.log file to verify scan-based traffic was correctly identified and recorded by the Zeek event-based engine.

```shell
head notice.log
```

Within the previous image, the red box denotes the terminal command while the orange box indicates the resulting notice generated by Zeek due to the ZeekDetectScans.zeek script's detection of a scan.
script file. The `zeek2` virtual machine, with an IP address of 10.0.0.2, was recorded to have scanned at least 15 unique ports on the `zeek1` virtual machine.

Concluding this section, we have reviewed the capabilities of Zeek for conducting packet analysis during live network traffic capture. The signature and script files reviewed in previous labs can be leveraged during such real-time analysis, allowing for Zeek to monitor and protect a network in real-time.

In the following section, we will review Zeek’s NetControl framework, which is used to create a `backend` communication channel with application firewalls and related monitoring systems.

2 Introduction to the Zeek NetControl framework

The Zeek NetControl framework is used to create a flexible, unified interface for active mitigation and response against anomalous traffic. The framework allows for connectivity between a large number of devices, removing the heterogeneity of such configurations through creating a task-oriented API. This API is developed using the Zeek scripting language, consisting of a number of high-level calls and lower-level rule syntax. This section will introduce and review basic Zeek NetControl calls and their implementation for network traffic analysis in real-time.

2.1 Viewing Zeek NetControl within a script file

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

![LXTerminal icon](image)

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

```bash
cd Zeek-Labs/Lab-Scripts/
```
Step 3. View the contents of the lab10_sec2-1.zeek file using \texttt{nl}.

\begin{verbatim}
nl lab10_sec2-1.zeek
\end{verbatim}

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

1. Initializes the NetControl API framework.
2. Creates a local variable to contain debug information.
3. Uses the NetControl API to activate debugging and display notifications and/or error messages.
4. Zeek event in which a connection between a source and destination is formed. This can be initialized by the TCP handshake or a series of UDP Request and Reply packets.
5. Checks if a NetControl rule already exists based on the source address requesting a connection.
6. Prints a debug error message if the rule exists.
7. Exits the function and begins checking for the next connection within the packet stream.
8. If a rule has not been created, add a rule to drop any connections made by the current source address that lasts over 20 seconds.
9. Prints a debug message that a new rule was created.

This script is relatively basic and straightforward yet shows the steps necessary to initialize the NetControl API. Without calling its initialization function, Zeek will be unable to communicate to various hardware devices through its backend.

Step 4. View the contents of the ZeekDetectSSHAttacks.zeek file using \texttt{nl}.

\begin{verbatim}
nl ZeekDetectSSHAttacks.zeek
\end{verbatim}
This script is very similar to the ZeekDetectScans.zeek default script reviewed in Lab 8 of this lab series. The following images will briefly review the file contents, while the Zeek documentation and previous lab provide a more in-depth analysis of this Zeek script. 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.


Command:

```
nl ZeekDetectSSHAttacks.zeek
```
Scroll down on the Terminal to view more of the script. Each number represents the respective line number:

25. Variable named `password_guesses_limit` that stores a numerical threshold for total number of failed SSH connections before marking a host as launching a brute-force attack.
28. Variable named `guessing_timeout` that stores a time-based threshold before resetting the `password_guesses_limit` variable back to 0.
29. Variable named `ignore_guessers` that stores a table of IP addresses identified to be launching SSH brute-force attacks. These addresses can be blocked or partially filtered.
Scroll down on the Terminal to view more of the script. Each number represents the respective line number:

79. Initializes the NetControl API framework.
80. Creates a local variable to contain debug information.
81. Uses the NetControl API to activate debugging and display notifications and/or error messages.
83. Zeek hook to the Notice logging stream so that we can append new information with default information.
84. Checks the Password_Guessing variable to determine if the current source address has been identified to be launching SSH brute-force attacks.
85. If the current source address was launching SSH brute-force attacks, create a new rule that will drop all network traffic from this source for the next 30 minutes.

Now that we have reviewed both scripts that will be used within the remainder of the lab, we can see the value of Zeek’s NetControl framework. By leveraging Zeek scripts we are able to identify anomalous network traffic events, detect malicious sources and finally leverage NetControl to mitigate their attacks. The remainder of this lab will include examples of executing the aforementioned Zeek scripts.

**2.2 Executing Zeek NetControl within a script file**

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

```bash
cd ../TCP-Traffic/
```
Step 2. Process the `smallFlows.pcap` packet capture file using the `lab10_sec2-1.zeek` script. To type capital letters, it is recommended to hold the **Shift** key while typing rather than using the **Caps** key. It is possible to use the **Tab** key to autocomplete the longer paths.

```
zeek -C -r ../Sample-PCAP/smallFlows.pcap ../Lab-Scripts/lab10_sec2-1.zeek > terminal.log
```

Because we have NetControl debugging enabled, we are going to save all error messages and notifications to the file `terminal.log`. By saving these notifications to a separate file, it is easier to view them in an organized fashion.

Ignore the permission denied error messages.

Step 3. View the file contents of the `terminal.log` file using **head**.

```
head terminal.log
```

```
netcontrol debug (Debug-All): init
netcontrol debug (Debug-All): add Rule:  
Error: Rule already exists!
Error: Rule already exists!
Error: Rule already exists!
Error: Rule already exists!
Error: Rule already exists!
Error: Rule already exists!
Error: Rule already exists!
Error: Rule already exists!
```

zeek@admin:~/.Zeek-Labs/TCP-Traffic$
Reviewing this image, the red box indicates the terminal command used to view the file. The orange box indicates the NetControl debug message that it has been initialized. The blue box indicates that a new rule has been created to drop all packets from the specified IP address for passing the connection-length threshold. The dark blue box indicates the Zeek event message that the rule was created successfully, while the yellow box indicates the Zeek event message that the rule already existed and a duplicate was not created.

**Step 4.** View the contents of the *TCP-Traffic* directory using `ls`.

```
ls
```

The two log files are interested in are the `netcontrol.log` and `netcontrol_drop.log` files. The `netcontrol.log` file will contain all information related to adding and removing rules, while the `netcontrol_drop.log` file will contain information regarding to when each rule was triggered and by which source address.

**Step 5.** View the file contents of the `netcontrol.log` file using `gedit`.

```
gedit netcontrol.log
```
Reviewing this image, the red box indicates that a Connection request was made by the source address 192.168.3.131 to the destination address 72.14.213.102. The orange box indicates that the connection was established while the blue box indicates that the connection was dropped and forced to time-out because of NetControl filtering packets from this source destination.

**Step 6.** Close the gedit by clicking the \(\times\) on the top right of the gedit window, then view the file contents of the `netcontrol_drop.log` file using `gedit`.

```bash
gedit netcontrol_drop.log
```

Reviewing this image, the red box indicates that a source address attempted to create a connection, breaking the NetControl rule we had previously implemented. Therefore, all packets were dropped from this source host during the time-out interval we declared within the `lab10_sec2-1.zeek` script.

**Step 7.** Clear the `TCP-Traffic` directory by using the `lab_clean.sh` shell script.

```bash
./../Lab-Scripts/lab_clean.sh
```
3 Identifying SSH attacks by leveraging the Zeek NetControl framework

Now that we have reviewed a basic implementation of the NetControl framework, creating a connection-based rule and identifying source addresses that broke the rule, we will conduct a more in-depth analysis on SSH brute-force password attacks.

**Step 1.** Process the sshguess.pcap packet capture file using ZeekDetectSSHAttacks.zeek. To type capital letters, it is recommended to hold the Shift key while typing rather than using the Caps key. It is possible to use the tab key to autocomplete the longer paths.

```
zeek -C -r ../Sample-PCAP/sshguess.pcap ./Lab-Scripts/ZeekDetectSSHAttacks.zeek
```

Similar to the previous section, we can see the NetControl debug messages including its initialization and creation of a new rule.

**Step 2.** View the contents of the TCP-Traffic directory using ls.

```
ls
```

We can see that the netcontrol.log, netcontrol_drop.log and notice.log files were created during packet capture analysis.

**Step 3.** View the file contents of the netcontrolLog file using gedit.
gedit netcontrol.log

Reviewing the previous image, the red box indicates that a new rule was created due to the address 192.168.56.1 surpassing the incorrect SSH password guessing threshold.

**Step 4.** Close the gedit by clicking the x on the top right of the gedit window, then view the file contents of the netcontrol_drop.log file using `gedit`.

gedit netcontrol_drop.log

Reviewing the previous image, the red box indicates which addresses were discovered to break the NetControl rules. In this example, only one address was discovered, 192.168.56.1.
Step 5. Close the gedit by clicking the X on the top right of the gedit window then, view the file contents of the notice.log file using gedit.

```
gedit notice.log
```

Recall that the notice.log file is generated by the ZeekDetectSSHAttacks.zeek script. The red box indicates which IP address was logged to have broken the SSH password guessing threshold.

### 3.1 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 NetControl framework and Zeek’s live processing of real-time network traffic. While the NetControl examples were performed
on offline packet capture files, by combining Zeek’s live analysis from Section 1 with the examples from Section 2 and 3, active measures can be taken for identifying malicious network traffic and blocking such sources.

References