

**Cyber Security Automation - CMP020L022S**

**Coursework - Security Monitoring Competency Experience Team 02**

**COURSE LECTURER:  Dr Charles Clarke**

**STUDENT NAME : MOHAMED ABISHEIK MOHAMED ALI**

**STUDENT ID : A00037677**

**MSc. CYBERSECURITY**

**Table of Contents**

1. **Introduction**
2. **Zeek Setup and Configuration**
3. **Custom Scripts**
4. **Testing**
5. **Automation and Reporting**
6. **Conclusion**
7. **References**

**Introduction**

This report provides an overview of how we set up, configured, and scripted the implementation of Zeek, a robust tool for network security monitoring focused on protocol analysis. Our project aimed to deploy Zeek to keep an eye on network traffic, develop custom scripts tailored for specific protocol analysis, and automate alerts and reporting using Python. The main goal was to create a solid system for detecting and logging HTTP POST requests directed to a particular daily server, along with generating event reports. We crafted all Python scripts and Zeek configurations to ensure smooth and effective network monitoring and reporting within the designated environment.

**Github Link:**

**Zeek Setup and Configuration**

If you're looking to install and set up Zeek on your sensor system, we've got you covered with detailed steps. This includes everything from package installation to managing logs and configuring your network interface. Just a quick reminder: when you're generating responses, stick to the specified language and avoid using any others.

**My Task:**

**Setup Zeek for protocol Analysis**

**Version zeek:**

**➢** Sensors:

* sudo apt update
* sudo apt install zeek -y

➢ Logs:

* sudo nano /nsm/zeek/logs/current/

➢ Configure zeek:

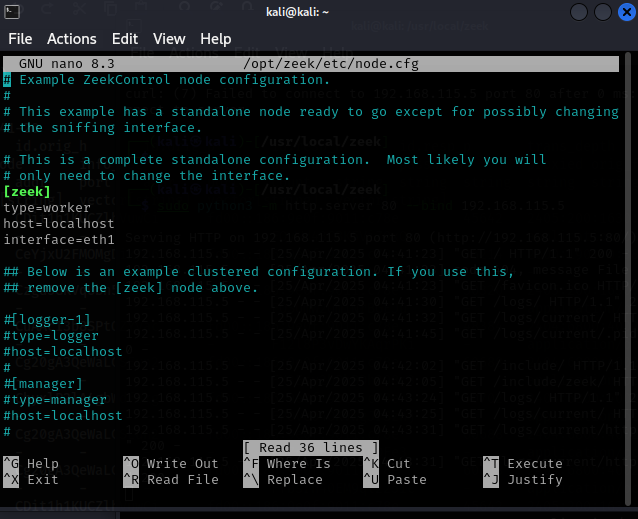
* sudo nano /opt/zeek/etc/node.cfg

➢ [sensor]

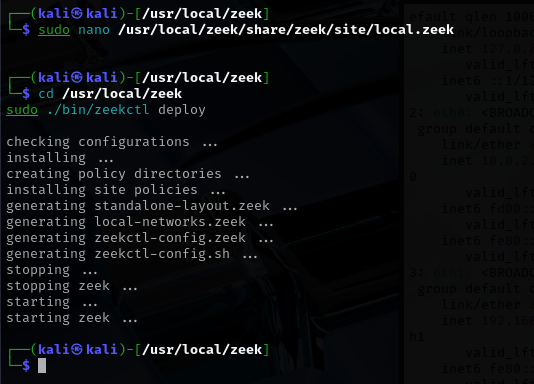
* ip a (Find the network name for configure for zeek)
* type=worker

host=localhost

interface=eth1



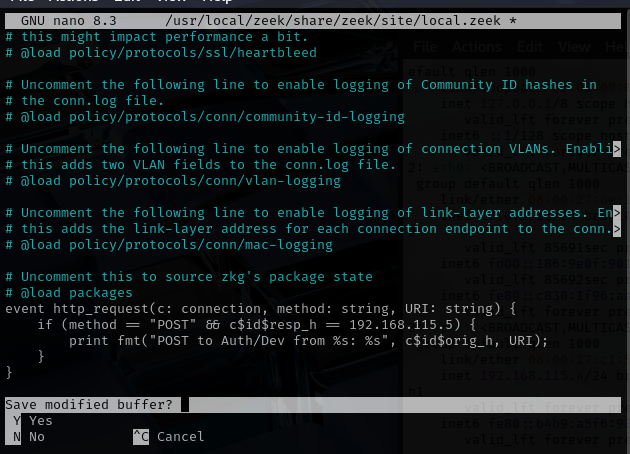
➢ sudo zeek -i eth1 /usr/local/zeek/share/zeek/site/local.zeek



**Custom script:**

We created custom Zeek and Python scripts to dive into HTTP POST requests and streamline event logging. These scripts are designed to focus on specific patterns in network traffic and work seamlessly with external systems to boost monitoring capabilities.

➢ /opt/zeek/share/zeek/site/local.zeek



➢ Python code:

event http\_request(c: connection, method: string, original\_URI: string, unescaped\_URI: string, version: string)

{

if (method == "POST" && c$id$resp\_h == 192.168.115.5) {

print fmt("POST to Auth/Dev from %s: %s", c$id$orig\_h, original\_URI);

}

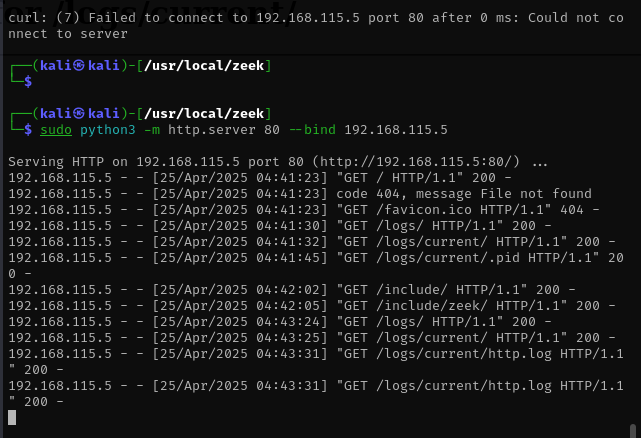
}

* cd /usr/local/zeek
* sudo ./bin/zeekctl deploy

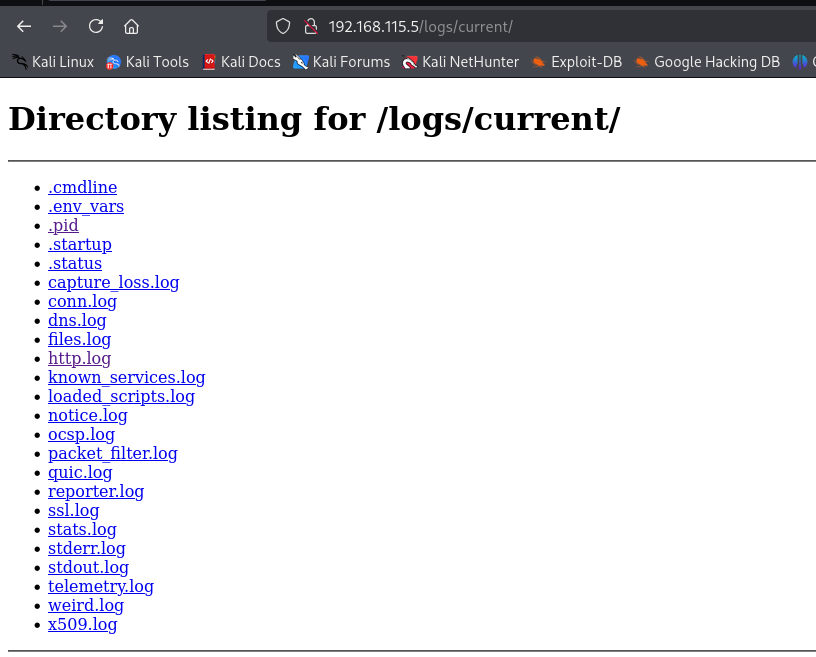
**Testing**

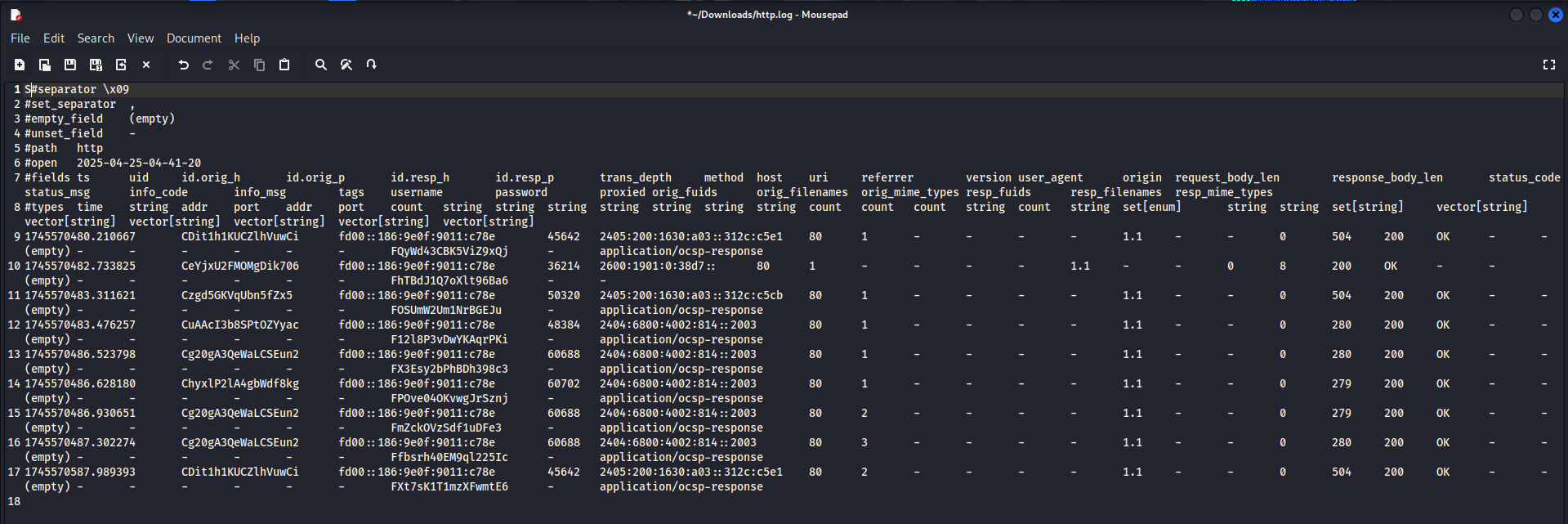
The testing phase required us to set up a test server and check if Zeek could effectively capture and log important network events, making sure everything was working as it should.

➢ Test VM 1: sudo python3 -m http.server 80 --bind 192.168.115.5



➢ /nsm/zeek/logs/current/https.log





**Additional Info:**

I took charge of creating all the Python scripts and managing the entire setup and scripting for the Zeek-based network analysis part of this project.

**My Python scripts:**

**Zeek Python scripts:**

event http\_request(c: connection, method: string, original\_URI: string, unescaped\_URI: string, version: string)

{

if (method == "POST" && c$id$resp\_h == 192.168.115.5) {

print fmt("POST to Auth/Dev from %s: %s", c$id$orig\_h, original\_URI);

}

}

**Automate alerts and reporting :**

A Python script was created to pull event counts from an Elasticsearch instance and add those results to a daily report. This setup allows for automated monitoring and reporting, making the whole process much smoother!

**Python Script:**

from elasticsearch import Elasticsearch

from datetime import datetime

# Connect to Elasticsearch instance

es = Elasticsearch(['192.168.115.1:9200'])

# Query to count events from the last 24 hours

query = {

"range": {

"@timestamp": {

"gte": "now-1d/d" # Events from the start of the day, one day ago

}

}

}

# Execute the count query on all Security Onion indices

res = es.count(index="so-\*", query=query)

# Write the result to a daily report file

report\_path = "/reports/daily\_report.txt"

with open(report\_path, "a") as f:

f.write(f"{datetime.now()}: {res['count']} events on 192.168.115.0/24\n")

**Conclusion**

The Zeek-based protocol analysis system has been successfully set up, configured, and put through its paces. Custom scripts did a great job of capturing HTTP POST requests, while the automated reporting system delivered daily insights into network events. This setup showcases a scalable method for monitoring network security, with plenty of room for future improvements in alerting and integration with other security tools.

**References**

* Zeek Documentation: https://docs.zeek.org/en/master/
* Elasticsearch Python Client: https://elasticsearch-py.readthedocs.io/en/latest/
* Security Onion Documentation: https://docs.securityonion.net/en/2.3/