Suricata IDS Rule Creation & Log Analysis

Project Type: Cybersecurity Lab
Tool Used: Suricata IDS
Focus Areas: Network Traffic Monitoring, Threat Detection, Log Analysis

---

Project Overview

This lab involved configuring and testing custom Intrusion Detection System (IDS) rules using **Suricata**, an open-source threat detection engine. The goal was to simulate real-world network monitoring and analyze traffic for potential threats using packet capture files and structured logs.

Objectives

- Develop and deploy custom Suricata rules
- Simulate network traffic using `.pcap` files
- Analyze Suricata logs (`fast.log`, `eve.json`) for triggered alerts
- Correlate events using flow identifiers and timestamps

Task 1: Examine a custom rule in Suricata

Rule Creation

- Wrote a custom rule to detect HTTP GET requests from `$HOME_NET` to `$EXTERNAL_NET`
- Used Suricata rule syntax including `alert`, `msg`, `flow`, `content`, `sid`, and `rev`
- Integrated rules into Suricata's configuration via `custom.rules`
- Use the `cat` command to display the rule in the `custom.rules` file: `cat custom.rules`

The command returns the rule as the output in the shell:

```
analyst@dc62de8bd78d:~$ cat custom.rules
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"GET on wire"; flow:est
ablished,to_server; content:"GET"; http_method; sid:12345; rev:3;)
analyst@dc62de8bd78d:~$
```

Task 2. Rule Execution & Log Analysis

- Ran Suricata against a sample `.pcap` file to simulate network activity
- Verified rule effectiveness by checking `fast.log` for alert entries

Before running Suricata, I verified the contents of the default log directory:

```
ls -l /var/log/suricata
```

```
analyst@dc62de8bd78d:~$ cat custom.rules
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"GET on wire"; flow:est
ablished,to_server; content:"GET"; http_method; sid:12345; rev:3;)
analyst@dc62de8bd78d:~$ ls -l /var/log/suricata
analyst@dc62de8bd78d:~$ sudo suricata -r sample.pcap -S custom.rules -k no
ne
22/9/2025 -- 12:34:13 - <Notice> - This is Suricata version 6.0.1 RELEASE
running in USER mode
22/9/2025 -- 12:34:13 - <Notice> - all 2 packet processing threads, 4 mana
gement threads initialized, engine started.
22/9/2025 -- 12:34:14 - <Notice> - Signal Received.  Stopping engine.
22/9/2025 -- 12:34:14 - <Notice> - Pcap-file module read 1 files, 200 pack
ets, 54238 bytes
analyst@dc62de8bd78d:~$ ls -l /var/log/suricata
total 16
-rw-r--r-- 1 root root 1419 Sep 22 12:34 eve.json
-rw-r--r-- 1 root root  292 Sep 22 12:34 fast.log
-rw-r--r-- 1 root root 3239 Sep 22 12:34 stats.log
-rw-r--r-- 1 root root 1512 Sep 22 12:34 suricata.log
analyst@dc62de8bd78d:~$ cat /var/log/suricata/fast.log
11/23/2022-12:38:34.624866  [**] [1:12345:3] GET on wire [**] [Classificat
ion: (null)] [Priority: 3] {TCP} 172.21.224.2:49652 -> 142.250.1.139:80
11/23/2022-12:38:58.958203  [**] [1:12345:3] GET on wire [**] [Classificat
ion: (null)] [Priority: 3] {TCP} 172.21.224.2:58494 -> 142.250.1.102:80
analyst@dc62de8bd78d:~$
```

Observation: The directory was empty, confirming no prior logs existed.

I executed Suricata using a sample packet capture file and a custom rule set:

```
sudo suricata -r sample.pcap -S custom.rules -k none
```

Explanation:

- `-r sample.pcap`: Loads the packet capture file to simulate network traffic.
- `-S custom.rules`: Applies my custom detection rules.
- `-k none`: Disables checksum validation, which is unnecessary for static `.pcap` files.

Note: `sudo` was required to access packet-level data during this lab, though it may not be needed in production environments.

Result: Suricata processed the packets and generated alerts based on rule matches.

After running Suricata, I listed the contents of the log directory again:

```
ls -l /var/log/suricata
```

Observation: Four new files were created, including:

- `fast.log`: Contains human-readable alert entries.
- `eve.json`: A structured JSON log for deeper analysis.

To inspect triggered alerts, I used the `cat` command:

```
cat /var/log/suricata/fast.log
```

Output: Alert entries were successfully logged, confirming that my rule was triggered by the simulated traffic.

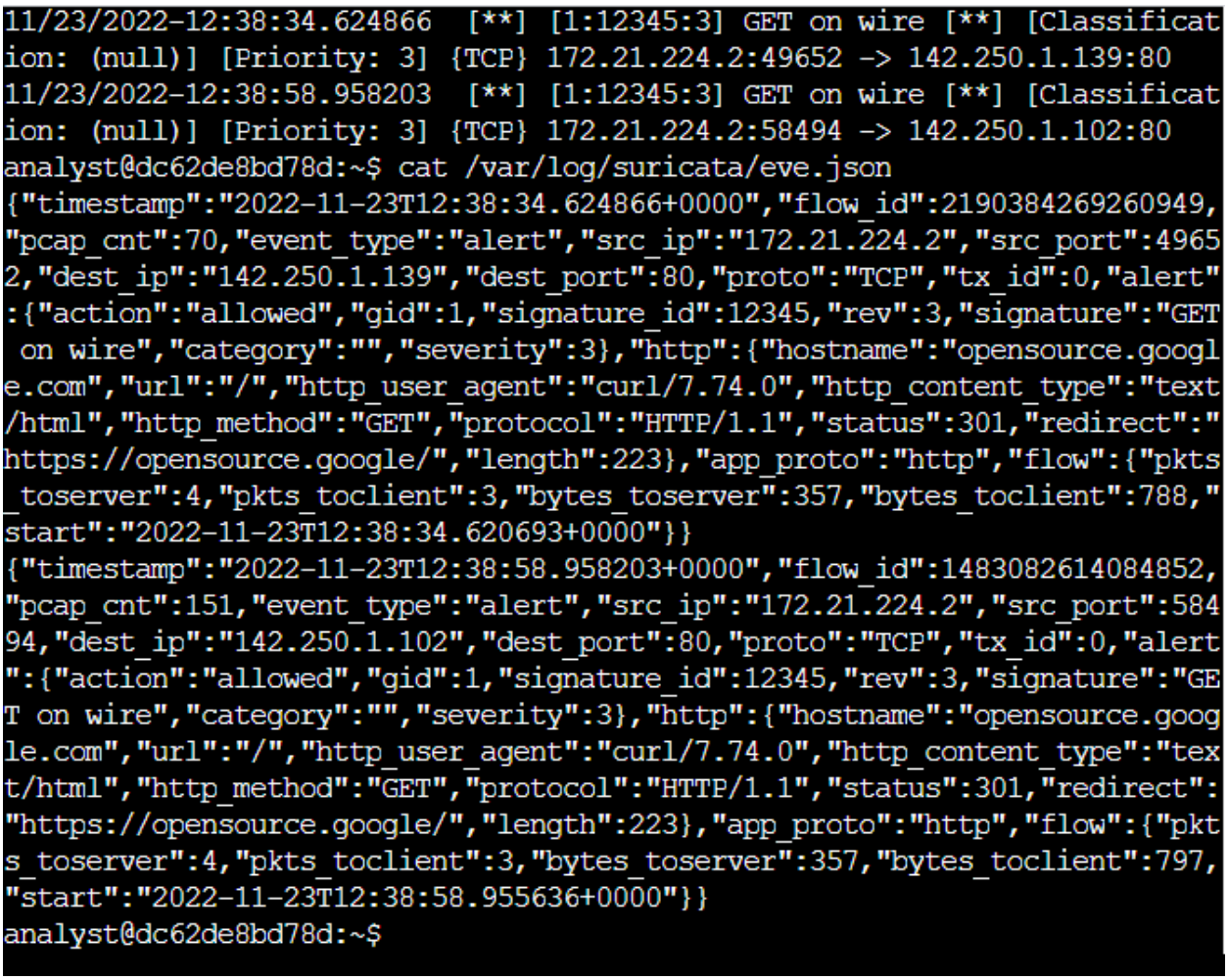
Task 3. Log Analysis

- Parsed `eve.json` using `jq` to extract key fields:
  - `timestamp`, `flow_id`, `alert.signature`, `proto`, `dest_ip`
- Used `flow_id` to correlate packets and reconstruct traffic flows
- Identified patterns and validated rule accuracy

Step 1: Viewing Raw Log Output

To begin, I used the `cat` command to display the contents of `eve.json`:

```
cat /var/log/suricata/eve.json
```



Observation: The output was extensive and difficult to interpret due to its raw JSON structure.

Step 2: Formatting with `jq`

To improve readability, I piped the output through the `jq` tool:

```
jq . /var/log/suricata/eve.json | less
```

Result: The formatted output was much easier to navigate. I used `less` to scroll through the entries and pressed `Q` to exit back to the terminal.

Note: `jq` is a powerful command-line tool for parsing and filtering JSON data essential for log analysis in cybersecurity workflows.

Step 3: Extracting Key Event Fields

To focus on specific threat indicators, I extracted select fields from each log entry:

```
jq -c "[.timestamp,.flow_id,.alert.signature,.proto,.dest_ip]"
/var/log/suricata/eve.json
```

Fields Extracted:

- `timestamp`: When the event occurred

- `flow_id`: Unique identifier for the network flow
- `alert.signature`: Description of the triggered rule
- `proto`: Protocol used (e.g., TCP)
- `dest_ip`: Destination IP address

Sample Output:

```
["2022-11-23T12:38:34.624866+0000",14500150016149,"GET on
wire","TCP","142.250.1.139"]
["2022-11-23T12:38:58.958203+0000",1647223379236084,"GET on
wire","TCP","142.250.1.102"]
```

Correlating Events by `flow_id`

To analyze all logs related to a specific network flow, I filtered entries using a known `flow_id`:

```
jq "select(.flow_id==14500150016149)" /var/log/suricata/eve.json
```

Insight: Suricata assigns a unique `flow_id` to each sequence of packets between a source and destination. This allows for precise event correlation and threat tracing across multiple log entries.

Skills Demonstrated

Skill	Description
IDS Rule Writing	Created precise detection logic using Suricata syntax
Log Analysis	Parsed JSON logs to extract and correlate threat data
CLI Proficiency	Used tools like `jq`, `cat`, and `less` for analysis
Network Security	Applied concepts of internal/external network boundaries

---

Key Takeaways

- Gained hands-on experience with Suricata's rule engine and logging system
- Learned to simulate and analyze network traffic for threat detection
- Strengthened my ability to interpret IDS alerts and correlate events