Detecting and Analysing Network Traffic

Introduction

In this project, I focused on learning how to detect and analyse network traffic using **Suricata**, a powerful open-source intrusion detection and prevention system (IDS/IPS). The project involved setting up Suricata on a Kali Linux virtual machine, creating custom detection rules, analyzing ICMP traffic, and validating the alerts generated. By completing this exercise, I strengthened my understanding of network monitoring, rule-based traffic detection, and real-world intrusion prevention techniques.

About Suricata

Suricata is an **open-source, high-performance network threat detection engine** developed by the Open Information Security Foundation (OISF). It functions as:

- **Intrusion Detection System (IDS):** Monitors network traffic and raises alerts when suspicious activity is detected.
- **Intrusion Prevention System (IPS):** Actively blocks or drops malicious traffic to protect the network.
- **Network Security Monitoring (NSM) tool:** Provides deep packet inspection and detailed traffic analysis.

In this project, I used Suricata in **IPS mode** to detect ICMP ping requests and generate alerts.

Project Steps

1. **Prepare Your Environment**
2. **Install Suricata**
3. **Update Rule Sets**
4. **Create a Custom Rule**
5. **Update Suricata Configuration**
6. **Restart Suricata**
7. **Verify Rule Loading**
8. **Generate ICMP Traffic**
9. **Check Suricata Logs**
10. **Real-Time Monitoring**

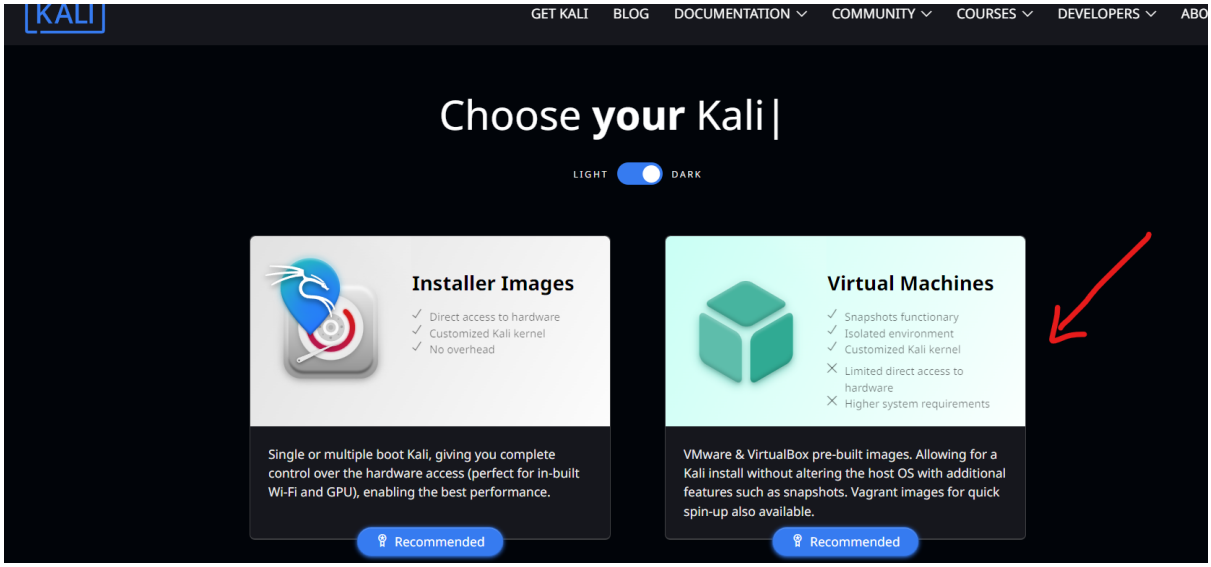
Step 1: Prepare Your Environment

For this project, I used **Kali Linux** as my primary operating system because of its robust security tools and suitability for penetration testing and network monitoring tasks. Instead of VirtualBox, I set up Kali Linux on **VMware Workstation**, which provided me with a stable and flexible virtual environment.

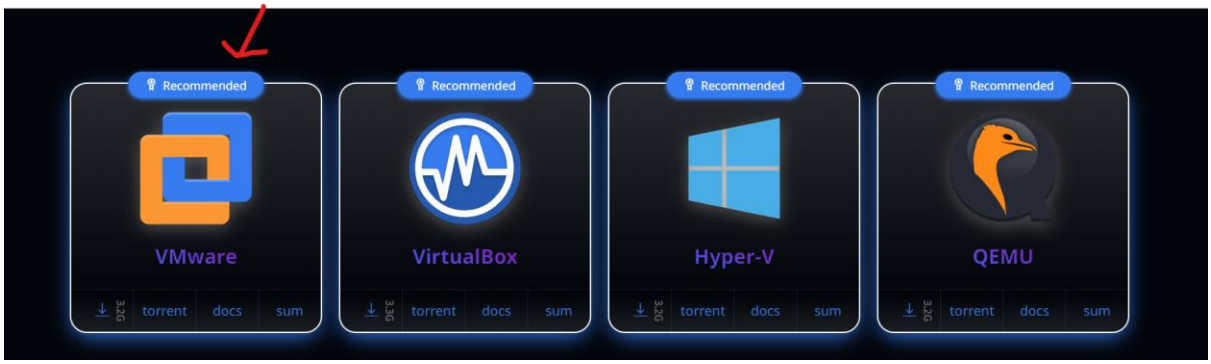
Within VMware, I created a dedicated virtual machine and allocated sufficient system resources (CPU, RAM, and disk space) to ensure smooth performance during the installation

and execution of Suricata. This setup allowed me to work in an isolated environment, ensuring both security and control while simulating real-world network monitoring scenarios.

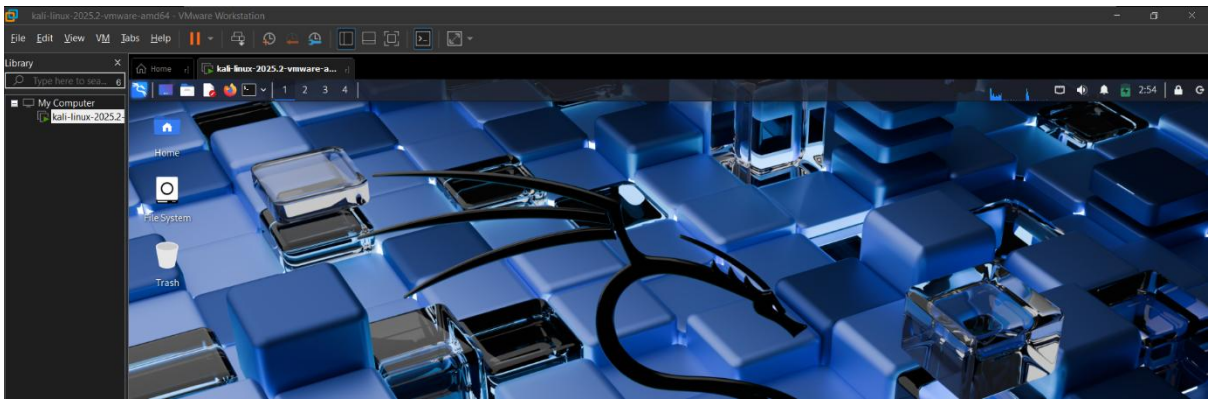
- Here, I downloaded Kali from official website.



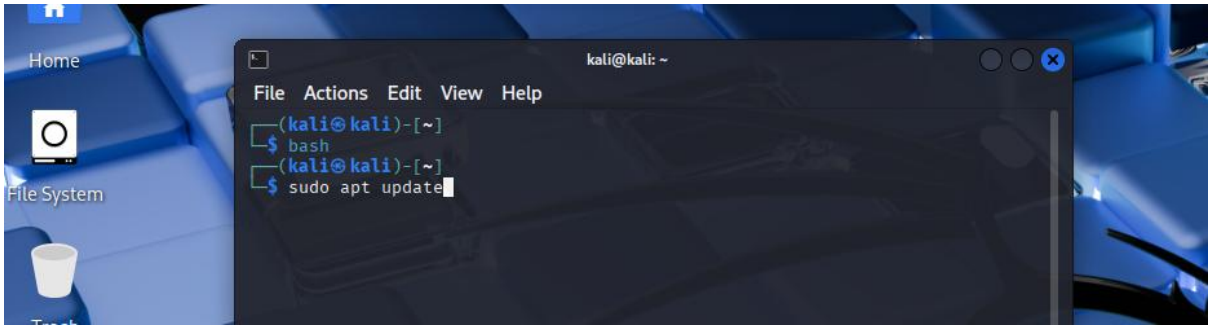
- Then, I have to choose for which VM tool I have to download I choose VMware.



- After completing the installation, I started the virtual machine, and logged in using the default credentials (username: *kali*, password: *kali*).



- Before moving forward, I had to update the OS.

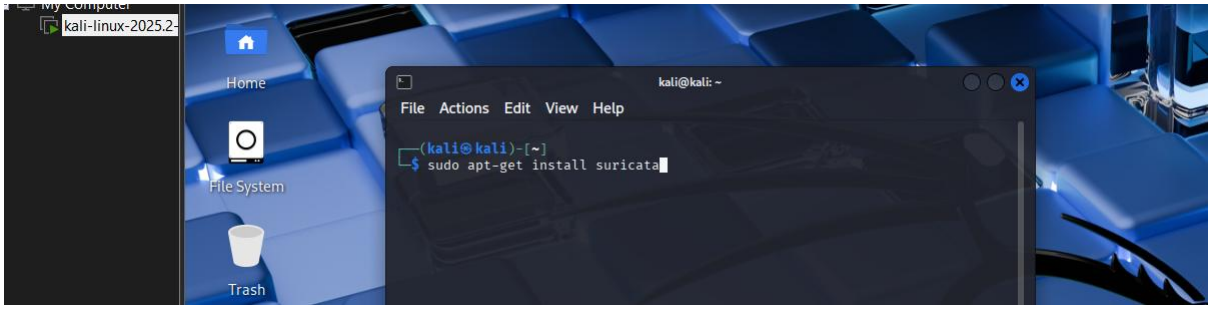


Step 2: Install Suricata

Once my Kali Linux environment was ready, I proceeded to install **Suricata**. I first updated the package list to ensure I had the latest repositories, and then installed Suricata using the following commands:

```
sudo apt update
sudo apt-get install suricata
```

After the installation, I verified it by checking the Suricata version to confirm that it was successfully installed and ready for configuration.



Step 3: Update Rule Sets

After installing Suricata, I updated the rule sets to ensure the latest detection signatures were available. This was done by running the following command:

```
sudo suricata-update
```

This step allowed Suricata to stay current with the most recent threat intelligence, enabling it to detect and respond to known network threats effectively.

Step 4: Create a Custom Rule

To detect specific network activity, I created a **custom Suricata rule**. I opened the `local.rules` file using the following command:

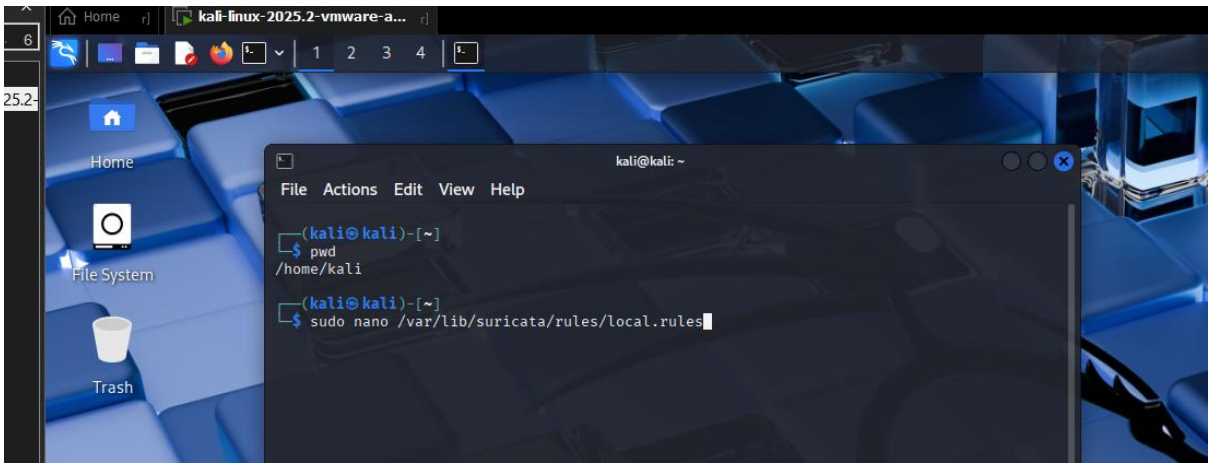
```
sudo nano /var/lib/suricata/rules/local.rules
```

Inside this file, I added a rule to detect **ICMP ping requests**:

```
alert icmp any any -> any any (msg:"ICMP Ping Detected"; itype:8;
sid:1000001; rev:1;)
```

- **alert** → action (generate an alert when matched)
- **icmp** → protocol (ICMP packets, e.g., ping)
- **any any -> any any** → from any IP/port to any IP/port
- **msg:** → the message shown in the alert log
- **itype:8** → ICMP type 8 (echo request, i.e., ping request)
- **sid:1000001** → unique rule ID (custom rules usually start from 1,000,000)
- **rev:1** → rule revision (increment if you update it later)

This rule allowed Suricata to generate alerts whenever an ICMP ping was detected, helping me monitor network traffic for potential reconnaissance activity.



Why custom rules matter

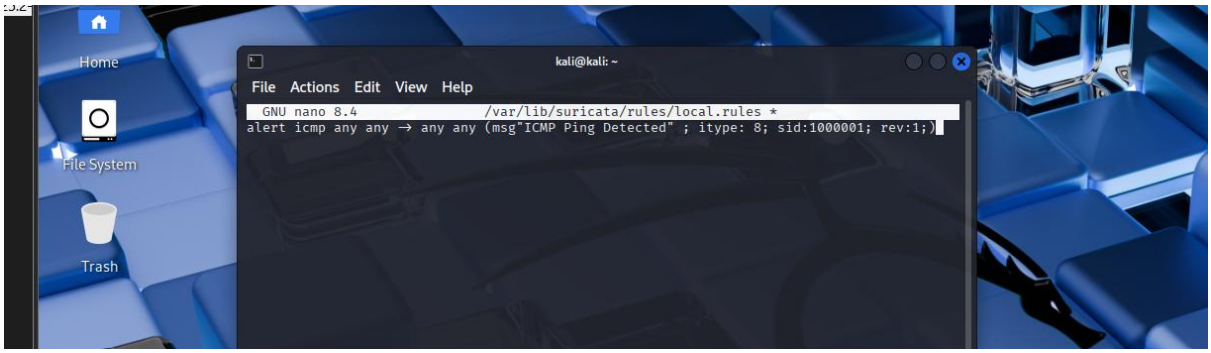
Suricata comes with community rule sets (Emerging Threats, etc.), but:

- They're **generic** and cover a wide range of known threats.
- They may not match the **specific traffic, systems, or policies** in *your* environment.

Custom rules let you:

- Detect **specific behaviours** relevant to your network.
- Practice **rule writing** — essential for security analysts.

- Create **lab exercises** to confirm Suricata is working correctly (like the ICMP ping rule).



Step 5: Update Suricata Configuration

- I opened the Suricata configuration file using the command:

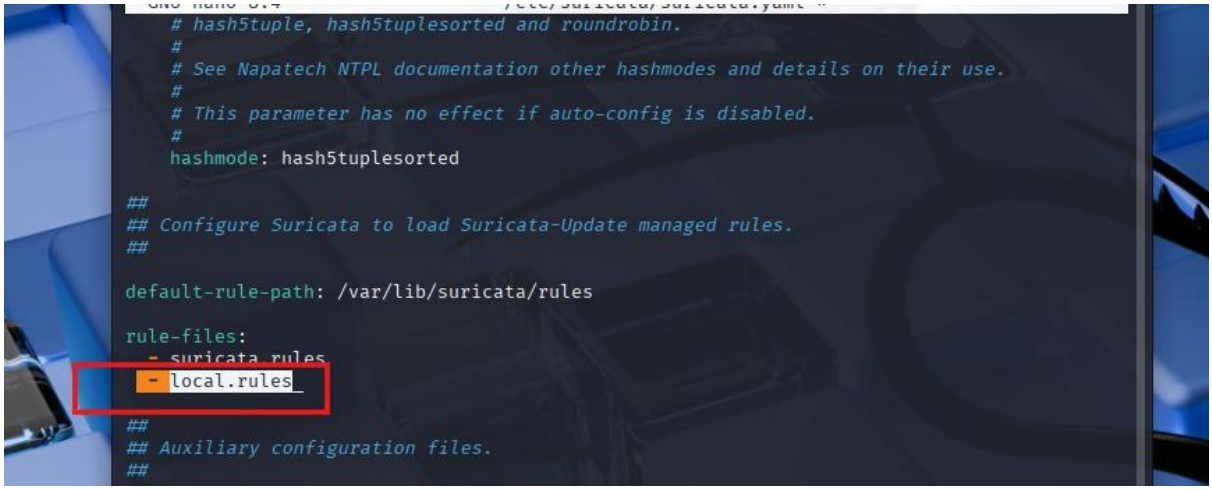
```
sudo nano /etc/suricata/suricata.yaml
```

- I set the **default-rule-path** to the directory where the rules are stored.
- I included **local.rules** in the **rule-files** section so that my custom rules would be loaded.
- I saved the changes and exited the editor.
- This ensured that Suricata would apply both default and custom rules whenever I started the service.

- I opened the Suricata configuration file using:

```
sudo nano /etc/suricata/suricata.yaml
```

- To quickly locate the setting, I pressed **Ctrl + W** and searched for `default-rule-path`.
- I set the path to the directory where the rules are stored.
- Under the `rule-files` section, I added **local.rules** to tell Suricata to include the custom rules I created.
- I saved the changes and exited the editor.
- This ensured that Suricata would load both default and custom rules whenever the service was started.

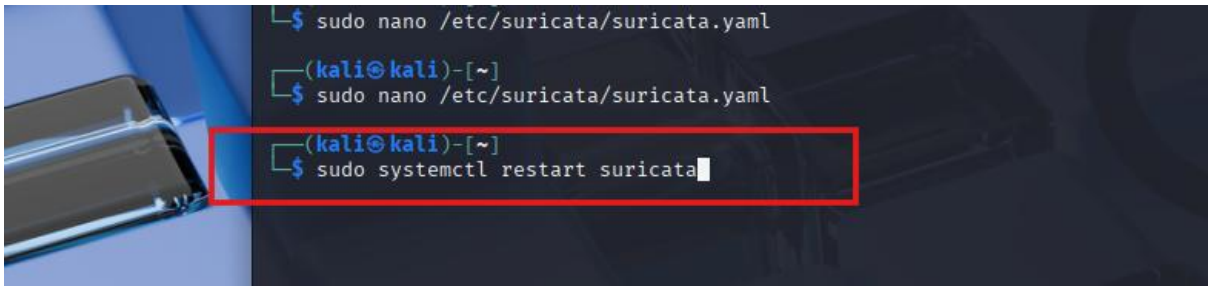


Step 6: Restart Suricata

- After updating the configuration, I restarted Suricata to apply the changes.
- I executed the following command:

```
sudo systemctl restart suricata
```

- Restarting the service ensured that Suricata loaded the updated configuration along with the custom rules I had added.

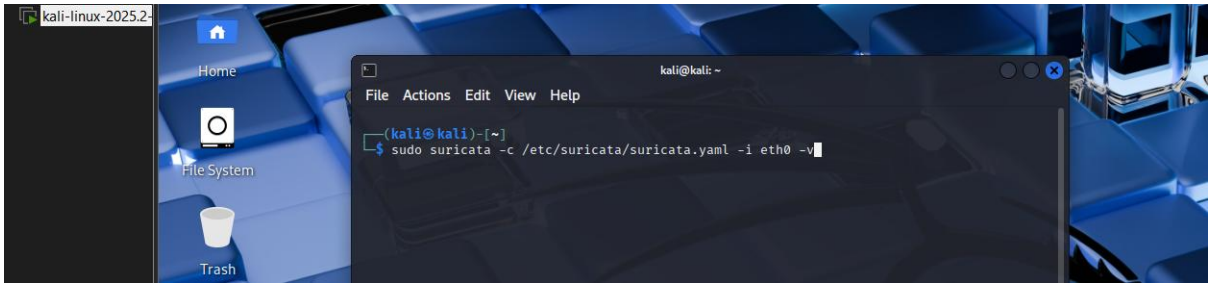


Step 7: Verify Rule Loading

- To ensure that my custom rules were properly loaded, I ran Suricata in verbose mode with the updated configuration.
- I executed the following command:

```
sudo suricata -c /etc/suricata/suricata.yaml -i eth0 -v
```

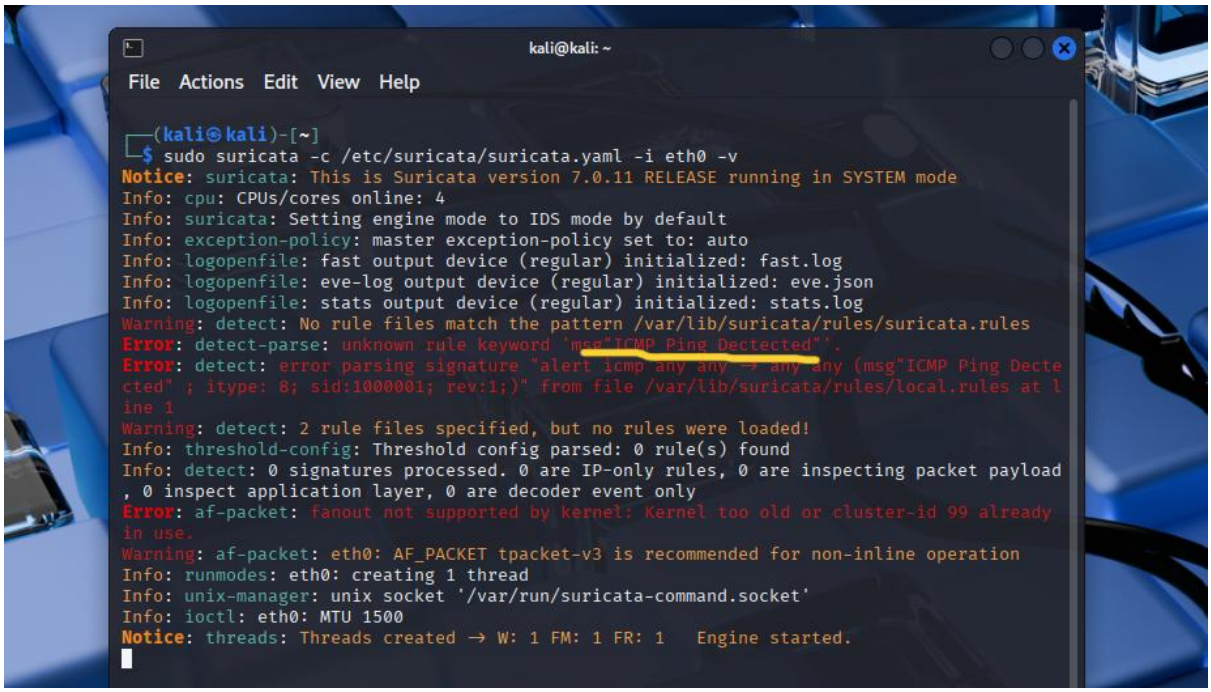
- This allowed me to verify that both the default and custom rules, including my ICMP detection rule, were active and ready to monitor network traffic.



- To ensure that my custom rules were properly loaded, I ran Suricata in verbose mode with the updated configuration:

```
sudo suricata -c /etc/suricata/suricata.yaml -i eth0 -v
```

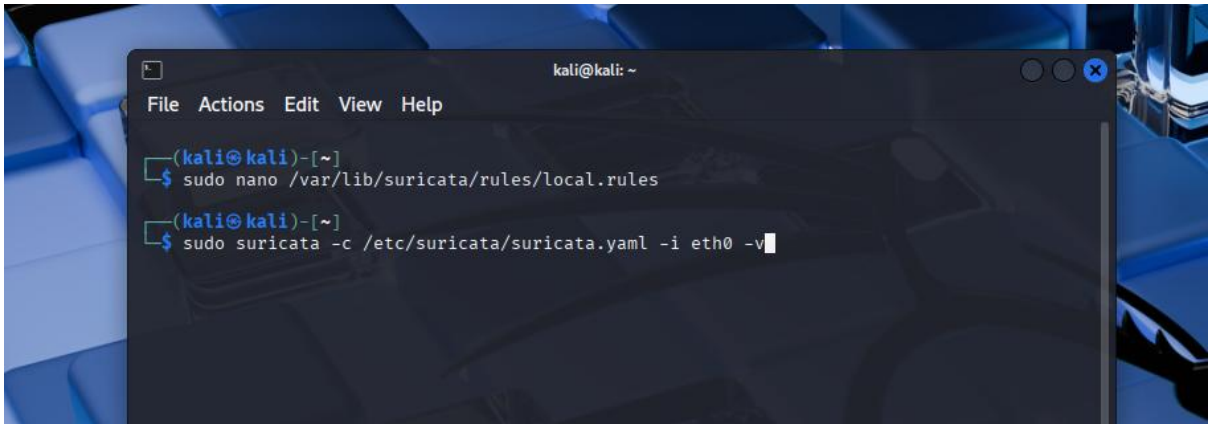
- During this process, I encountered a few errors:
  - A spelling mistake in my rule message ("Deteced" instead of "Detected") and some indentation issues, which I corrected.
  - An **af-packet fanout** error occurred initially because I was using NAT networking. I resolved this by switching the VM network to **bridged mode** and adjusting the `cluster-id` in the YAML configuration.
- After these corrections, Suricata successfully loaded all default and custom rules, including my ICMP detection rule, and was ready to monitor network traffic.



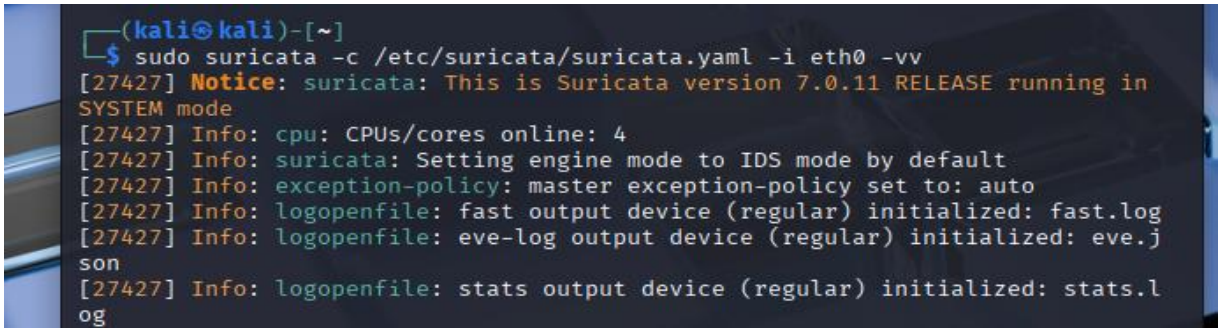
- After fixing the errors, I **restarted Suricata** to apply the changes I made:

```
sudo systemctl restart suricata
```

- This ensured that all corrections, including rule fixes and configuration updates, were properly loaded and active.



- After restarting Suricata, I ran it again and confirmed that it loaded **without any errors or warnings**, indicating that all rules and configuration changes were successfully applied.



Step 8: Generate ICMP Traffic

- To test whether my custom ICMP detection rule was working, I needed to generate ICMP traffic on the network.
- I opened a **new terminal** on my Kali Linux VM to avoid interrupting Suricata.
- I executed the following command to send ICMP ping requests:

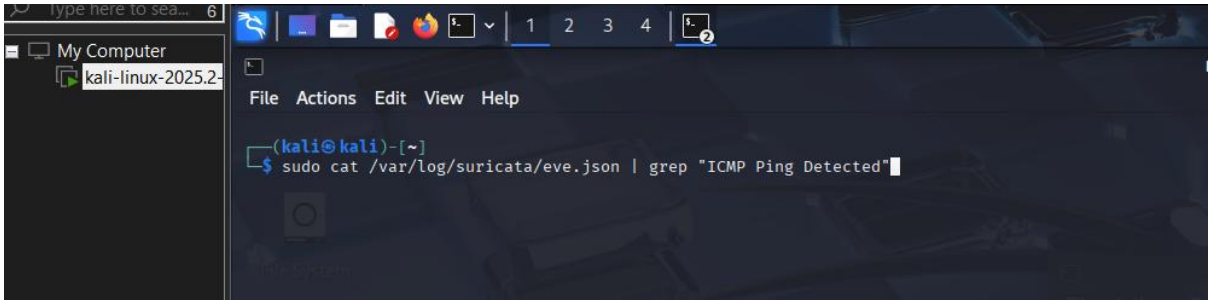
```
ping -c 4 8.8.8.8
```

- This generated ICMP traffic that could be detected by Suricata, allowing me to verify that the custom rule was functioning correctly.

Step 9: Check Suricata Logs

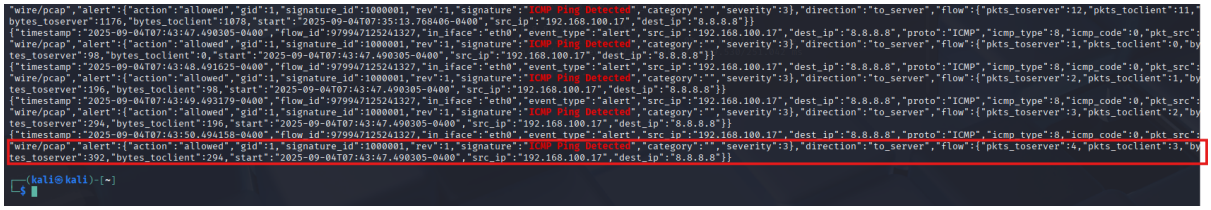
- After generating ICMP traffic, I verified that Suricata successfully detected the pings by checking its logs.
- I used the following command on new terminal to filter the alerts for my custom ICMP rule:

```
sudo cat /var/log/suricata/eve.json | grep "ICMP Ping Detected"
```



.

- This allowed me to confirm that each ICMP ping was detected and logged by Suricata, validating that my custom rule was functioning as intended.



Step 10: Real-Time Monitoring

- To observe Suricata detections in **real-time**, I used **three terminals** on my Kali Linux VM:
    1. In the **first terminal**, I started Suricata so it could actively monitor network traffic.
    2. In the **second terminal**, I viewed real-time alerts using the following command:

```
sudo tail -f /var/log/suricata/eve.json | jq . | grep "ICMP Ping Detected"
```

- **Explanation of the command:**
    - `sudo tail -f /var/log/suricata/eve.json` → Continuously reads the end of the Suricata log file (`eve.json`) as new events are added.
    - `jq .` → Formats the JSON output to make it easier to read.
    - `grep "ICMP Ping Detected"` → Filters the output to show only alerts triggered by my custom ICMP ping rule.
- In the **third terminal**, I generated ICMP traffic using:

```
ping -c 4 8.8.8.8
```

- This setup allowed me to **watch Suricata detect ICMP pings in real-time**, validating that the system was functioning as intended and providing immediate feedback on network activity.

- To make the real-time alerts more **refined and readable**, I used the following command in another terminal:

```
sudo tail -f /var/log/suricata/eve.json \
  | jq -r 'select(.alert.signature=="ICMP Ping Detected") | "\"The time of
the ping is = \(.timestamp) = \(.src_ip) → \(.dest_ip) :
\(.alert.signature)"'
```

- **Explanation of the command:**
  - `sudo tail -f /var/log/suricata/eve.json` → Continuously monitors the Suricata log file.
  - `jq -r 'select(.alert.signature=="ICMP Ping Detected") ...'` → Filters and formats only the ICMP ping alerts.
  - The output displays the **timestamp, source IP, destination IP, and alert signature** in a clear, human-readable format.
- This allowed me to **watch ICMP ping alerts in real-time with a concise summary**, making it easier to analyze network activity immediately.