Building a Lightweight SIEM and IDPS: Featuring Suricata, Loki, Promtail and LogCLI

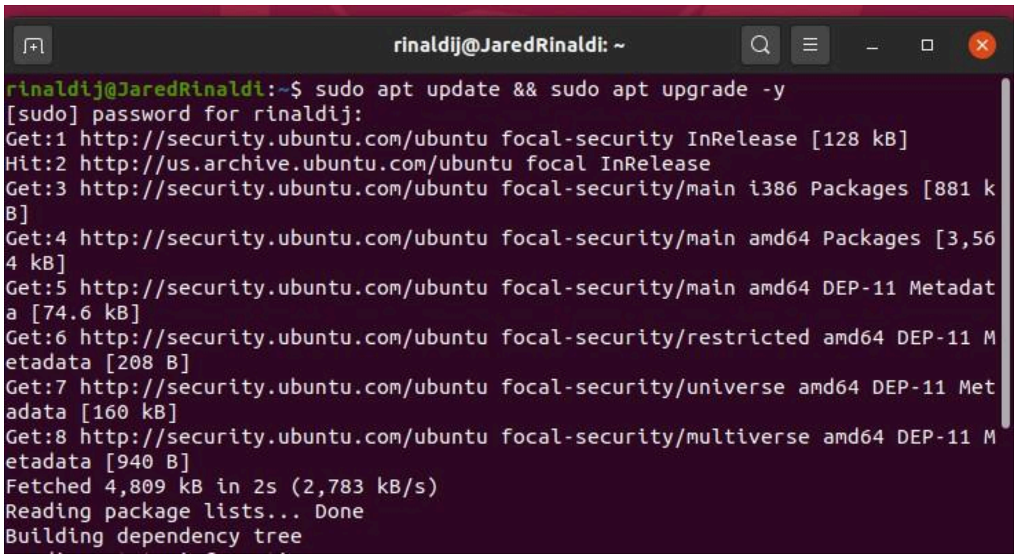
Researcher: Jared Rinaldi
November 11, 2025
Github.com/PyronicGreen

SIEM: Security Information and Event Management. A type of centralized platform that aggregates security information from logs, alerts, artifacts and events. All SIEM solutions perform some level of data aggregation, consolidation and sorting functions to identify threats and adhere to data compliance requirements.

IDPS: Amalgam of IDS (Intrusion Detection System) and IPS (Intrusion Prevention System). A network security solution that monitors, detects and prevents potential malicious activity or policy violations.

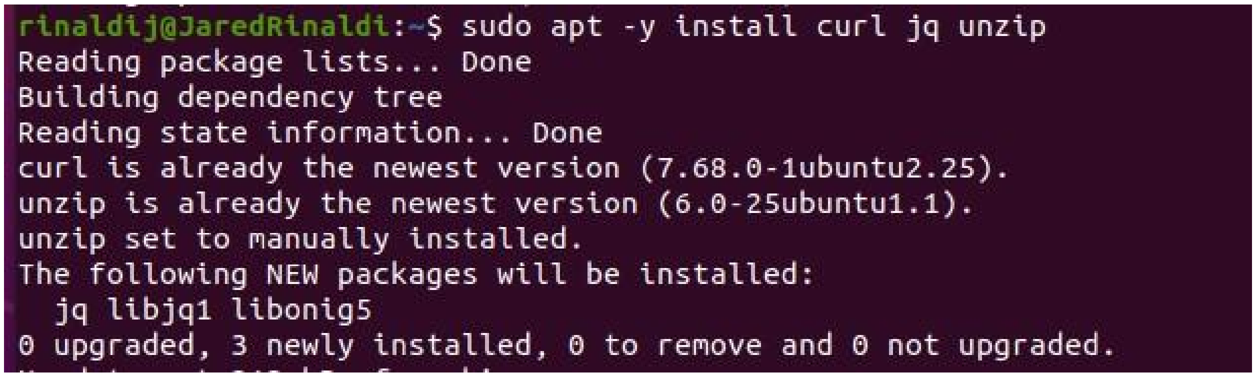
Part One - Prepare the System

1. For this step, the goal is to make sure the system is updated and then install the essential tools for this lab: **curl**, **jq**, and **unzip**.
    a. **curl** - a command-line tool for transferring data via HTTP/S, FTP, etc.
    b. **jq** - a lightweight JSON processor used to pretty-print log files.
    c. **unzip** - utility to extract .zip files.



```
rinaldij@JaredRinaldi:~$ sudo apt update && sudo apt upgrade -y
[sudo] password for rinaldij:
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [881 k
B]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [3,56
4 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadat
a [74.6 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 DEP-11 M
etadata [208 B]
Get:7 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Met
adata [160 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 DEP-11 M
etadata [940 B]
Fetched 4,809 kB in 2s (2,783 kB/s)
Reading package lists... Done
Building dependency tree
```

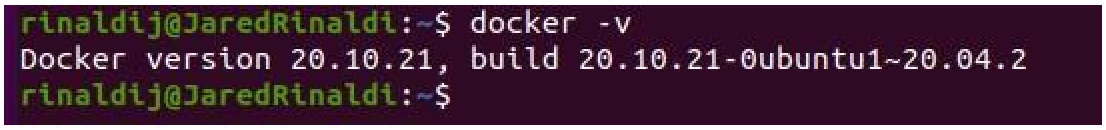
Packages are updated.



```
rinaldij@JaredRinaldi:~$ sudo apt -y install curl jq unzip
Reading package lists... Done
Building dependency tree
Reading state information... Done
curl is already the newest version (7.68.0-1ubuntu2.25).
unzip is already the newest version (6.0-25ubuntu1.1).
unzip set to manually installed.
The following NEW packages will be installed:
  jq libjq1 libonig5
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
```

*It seems curl and unzip are already installed and the newest versions, so this command installed 3 new packages related to jq: **jq**, **libjq1**, and **libonig5**.*

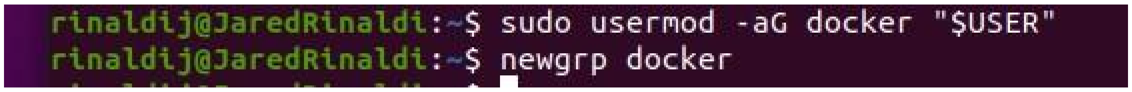
2. Install Docker. **Docker** is a lightweight virtualization platform that lets a user run applications in isolated containers. **Containers** are lightweight and contain everything needed to run the application.



```
rinaldij@JaredRinaldi:~$ docker -v
Docker version 20.10.21, build 20.10.21-0ubuntu1~20.04.2
rinaldij@JaredRinaldi:~$
```

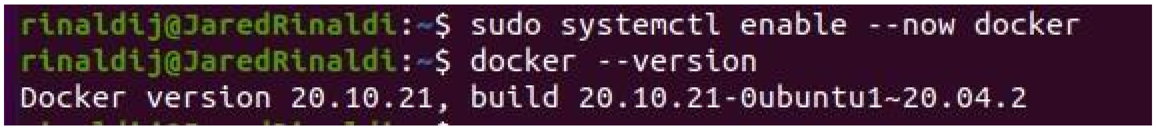
Docker is already installed.

3. The next command adds the user to the docker group which allows the user to run docker without sudo. The second command starts a fresh shell with the new group, docker.



```
rinaldij@JaredRinaldi:~$ sudo usermod -aG docker "$USER"
rinaldij@JaredRinaldi:~$ newgrp docker
```

4. Enable and start the docker service.



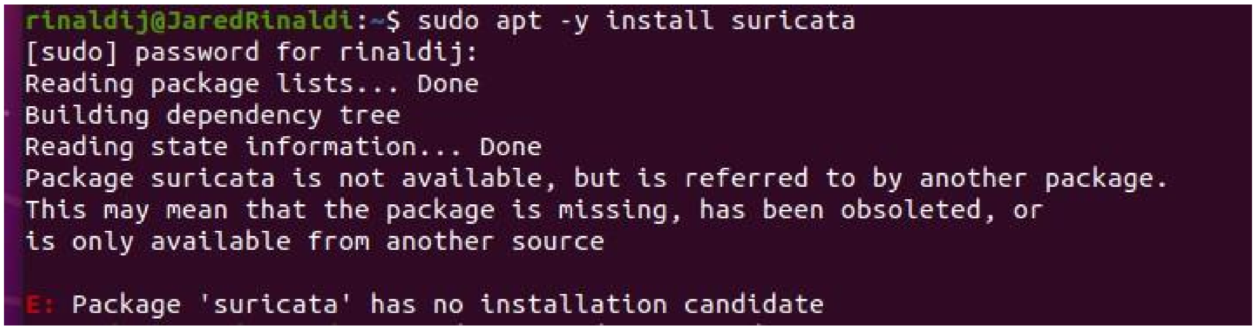
```
rinaldij@JaredRinaldi:~$ sudo systemctl enable --now docker
rinaldij@JaredRinaldi:~$ docker --version
Docker version 20.10.21, build 20.10.21-0ubuntu1~20.04.2
```

Part Two - Suricata

Suricata: A high-performance network IDS, IPS and network security monitoring engine.

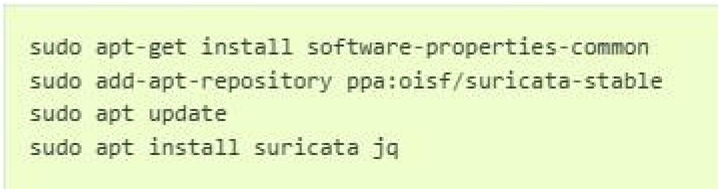
After reading the following [Medium article](), I see that there is a contentious debate about whether **Suricata** or **Snort** is the preferred network intrusion detection system (NIDS). While both tools are open-source and use rules to detect malicious activity, Suricata has a multi-threaded architecture, while Snort is single-threaded. This means that Suricata can process multiple tasks simultaneously and do so in a fast and efficient way. Snort stands out by having an extensive rule set, which allows for customization to meet specific purposes. Snort also has broad compatibility with a variety of platforms, making it versatile. It seems both tools excel at packet capture and analysis, which explains why both are so popular. They both also offer full protocol analysis.

When I first ran the pre-flight setup for Suricata, I received an error message stating that Suricata had no installation candidate.



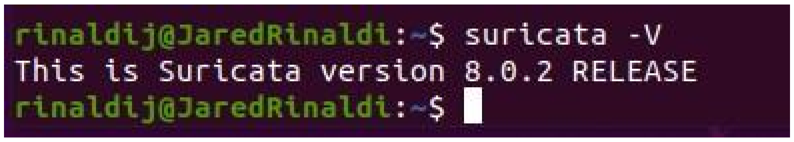
```
rinaldij@JaredRinaldi:~$ sudo apt -y install suricata
[sudo] password for rinaldij:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package suricata is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'suricata' has no installation candidate
```

I then referred to the Suricata documentation, and followed the instructions there for setup:



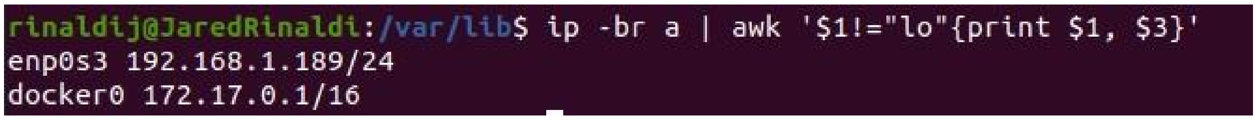
```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt update
sudo apt install suricata jq
```

After running these commands, Suricata was successfully installed.



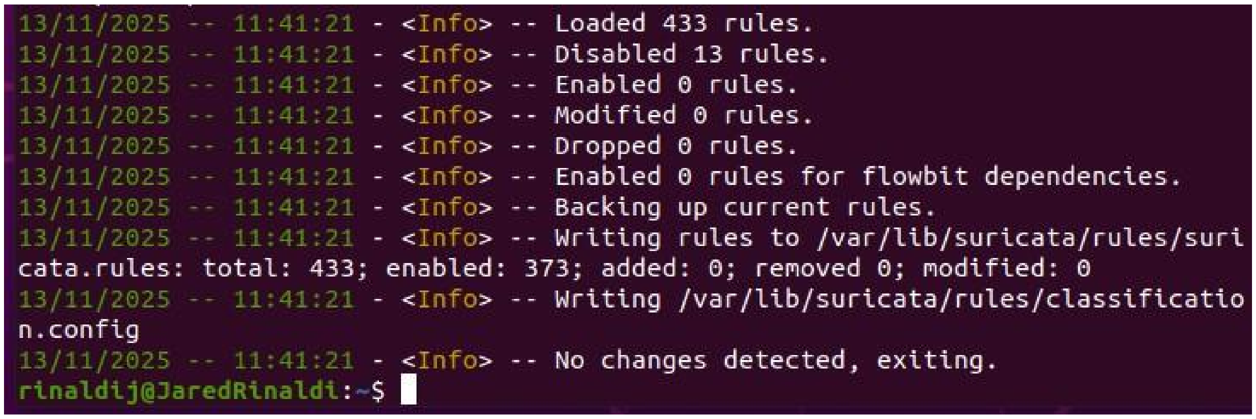
```
rinaldij@JaredRinaldi:~$ suricata -V
This is Suricata version 8.0.2 RELEASE
rinaldij@JaredRinaldi:~$
```

I ran the command to check my interface name:



```
rinaldij@JaredRinaldi:/var/lib$ ip -br a | awk '$1!="lo"{print $1, $3}'
enp0s3 192.168.1.189/24
docker0 172.17.0.1/16
```

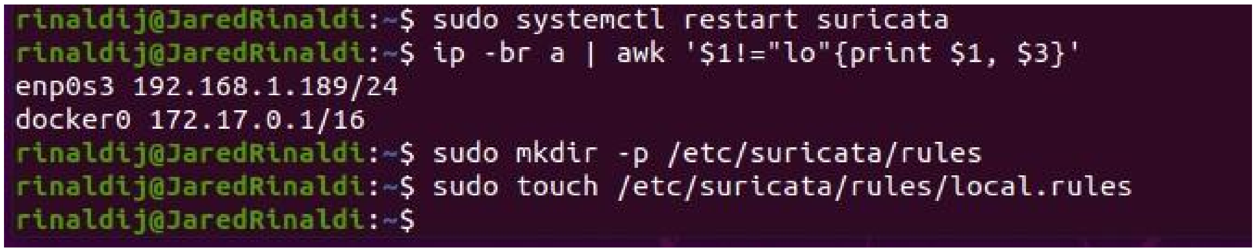
I was having some issues getting suricata to run, so I wanted to make sure I installed everything correctly. I therefore followed the “SIEM lab fixes with Suricata” document to uninstall Suricata and then re-install and download the rules.



```
13/11/2025 -- 11:41:21 - <Info> -- Loaded 433 rules.
13/11/2025 -- 11:41:21 - <Info> -- Disabled 13 rules.
13/11/2025 -- 11:41:21 - <Info> -- Enabled 0 rules.
13/11/2025 -- 11:41:21 - <Info> -- Modified 0 rules.
13/11/2025 -- 11:41:21 - <Info> -- Dropped 0 rules.
13/11/2025 -- 11:41:21 - <Info> -- Enabled 0 rules for flowbit dependencies.
13/11/2025 -- 11:41:21 - <Info> -- Backing up current rules.
13/11/2025 -- 11:41:21 - <Info> -- Writing rules to /var/lib/suricata/rules/suri
cata.rules: total: 433; enabled: 373; added: 0; removed 0; modified: 0
13/11/2025 -- 11:41:21 - <Info> -- Writing /var/lib/suricata/rules/classificatio
n.config
13/11/2025 -- 11:41:21 - <Info> -- No changes detected, exiting.
rinaldij@JaredRinaldi:~$
```

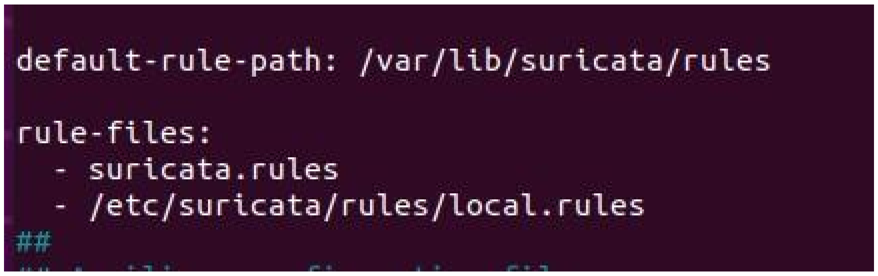
*After running **sudo suricata-update**, 433 rules were confirmed to be loaded.*

I next created a directory and file for custom rules:



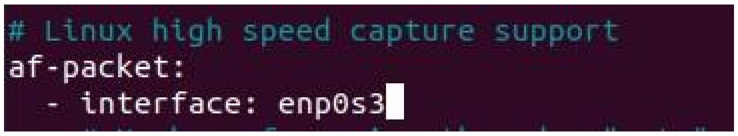
```
rinaldij@JaredRinaldi:~$ sudo systemctl restart suricata
rinaldij@JaredRinaldi:~$ ip -br a | awk '$1!="lo"{print $1, $3}'
enp0s3 192.168.1.189/24
docker0 172.17.0.1/16
rinaldij@JaredRinaldi:~$ sudo mkdir -p /etc/suricata/rules
rinaldij@JaredRinaldi:~$ sudo touch /etc/suricata/rules/local.rules
rinaldij@JaredRinaldi:~$
```

My next step was to update the suricata.yaml file to the correct rule path. After searching for the **default-rule-path** using **control + w** in nano, I saw that the rule path was already set to **/var/lib/suricata/rules**. I added **/etc/suricata/rules/local.rules** to rule-files.



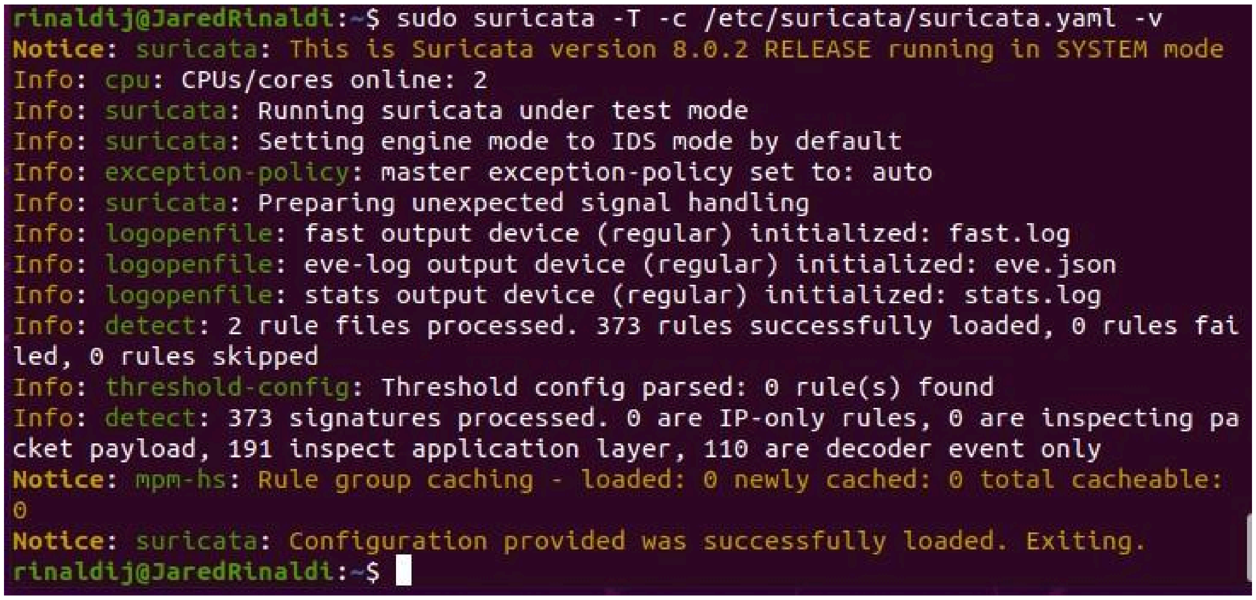
```
default-rule-path: /var/lib/suricata/rules

rule-files:
  - suricata.rules
  - /etc/suricata/rules/local.rules
##
```

I then searched for **af-packet** using **control+w** and replaced the listed interface with my interface: enp0s3.



```
# Linux high speed capture support
af-packet:
  - interface: enp0s3
```

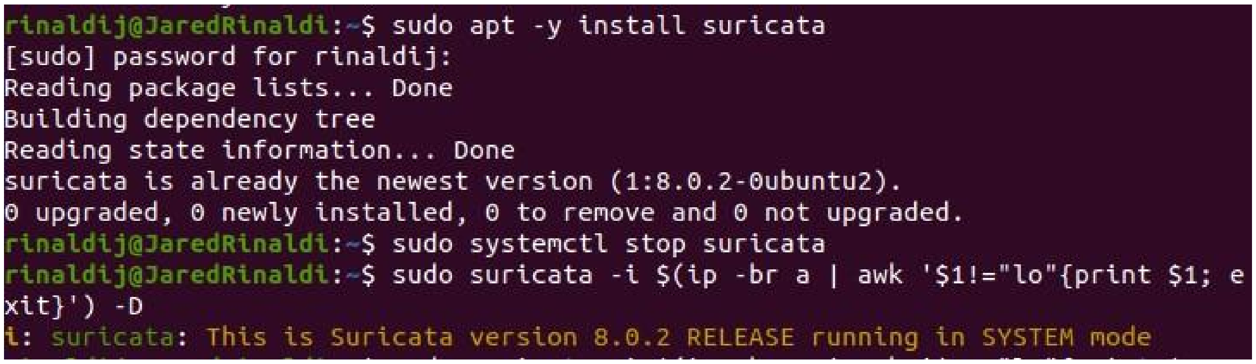
After this, I checked to validate my Suricata connection. All seems to be in working order:



```
rinaldij@JaredRinaldi:~$ sudo suricata -T -c /etc/suricata/suricata.yaml -v
Notice: suricata: This is Suricata version 8.0.2 RELEASE running in SYSTEM mode
Info: cpu: CPUs/cores online: 2
Info: suricata: Running suricata under test mode
Info: suricata: Setting engine mode to IDS mode by default
Info: exception-policy: master exception-policy set to: auto
Info: suricata: Preparing unexpected signal handling
Info: logopenfile: fast output device (regular) initialized: fast.log
Info: logopenfile: eve-log output device (regular) initialized: eve.json
Info: logopenfile: stats output device (regular) initialized: stats.log
Info: detect: 2 rule files processed. 373 rules successfully loaded, 0 rules fai
led, 0 rules skipped
Info: threshold-config: Threshold config parsed: 0 rule(s) found
Info: detect: 373 signatures processed. 0 are IP-only rules, 0 are inspecting pa
cket payload, 191 inspect application layer, 110 are decoder event only
Notice: mpm-hs: Rule group caching - loaded: 0 newly cached: 0 total cacheable:
0
Notice: suricata: Configuration provided was successfully loaded. Exiting.
rinaldij@JaredRinaldi:~$ 
```

- **Suricata -T flag:** Test configuration.
- **Suricata -c flag:** This flag, followed by the path name, will provide the path to the configuration file.
- **Suricata -v flag:** Increase the verbosity of the Suricata application logging by increasing the log level from the default. This option can be passed multiple times to further increase the verbosity. A single -v will provide INFO.

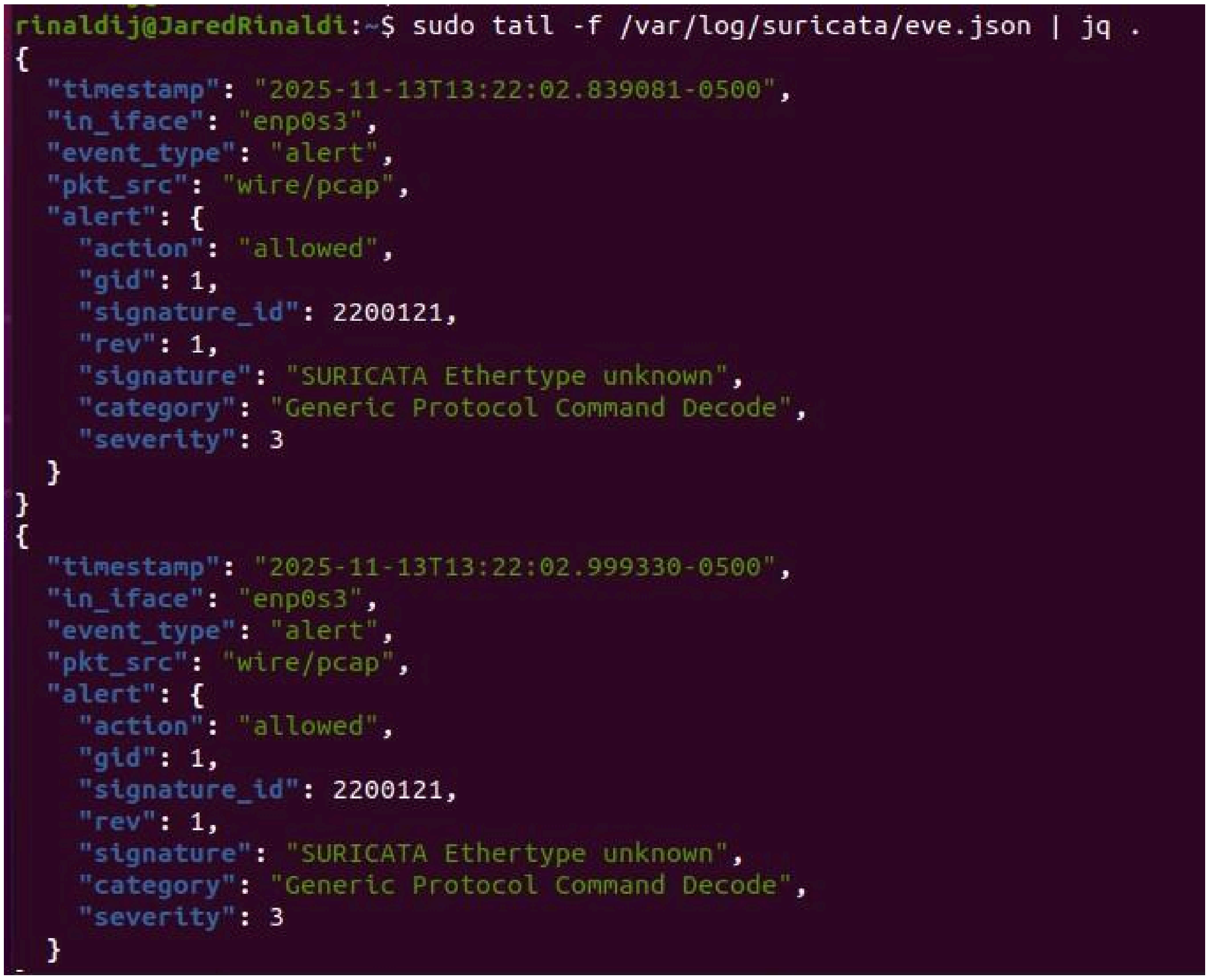
The following commands were run to install and start Suricata.



```
rinaldij@JaredRinaldi:~$ sudo apt -y install suricata
[sudo] password for rinaldij:
Reading package lists... Done
Building dependency tree
Reading state information... Done
suricata is already the newest version (1:8.0.2-0ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
rinaldij@JaredRinaldi:~$ sudo systemctl stop suricata
rinaldij@JaredRinaldi:~$ sudo suricata -i $(ip -br a | awk '$1!="lo"{print $1; e
xit}') -D
i: suricata: This is Suricata version 8.0.2 RELEASE running in SYSTEM mode
```

The first line installs Suricata, the second stops the default service so that Suricata can be controlled manually, and the third starts Suricata in the background of my VM's primary network interface, allowing it to monitor network traffic and generate alerts.

I then confirmed that Suricata was writing logs:



```
rinaldij@JaredRinaldi:~$ sudo tail -f /var/log/suricata/eve.json | jq .
{
  "timestamp": "2025-11-13T13:22:02.839081-0500",
  "in_iface": "enp0s3",
  "event_type": "alert",
  "pkt_src": "wire/pcap",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2200121,
    "rev": 1,
    "signature": "SURICATA Ethertype unknown",
    "category": "Generic Protocol Command Decode",
    "severity": 3
  }
}
{
  "timestamp": "2025-11-13T13:22:02.999330-0500",
  "in_iface": "enp0s3",
  "event_type": "alert",
  "pkt_src": "wire/pcap",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2200121,
    "rev": 1,
    "signature": "SURICATA Ethertype unknown",
    "category": "Generic Protocol Command Decode",
    "severity": 3
  }
```

Question - What type of events are shown in the eve.json file?

I noticed several different types of events listed in the eve.json file:

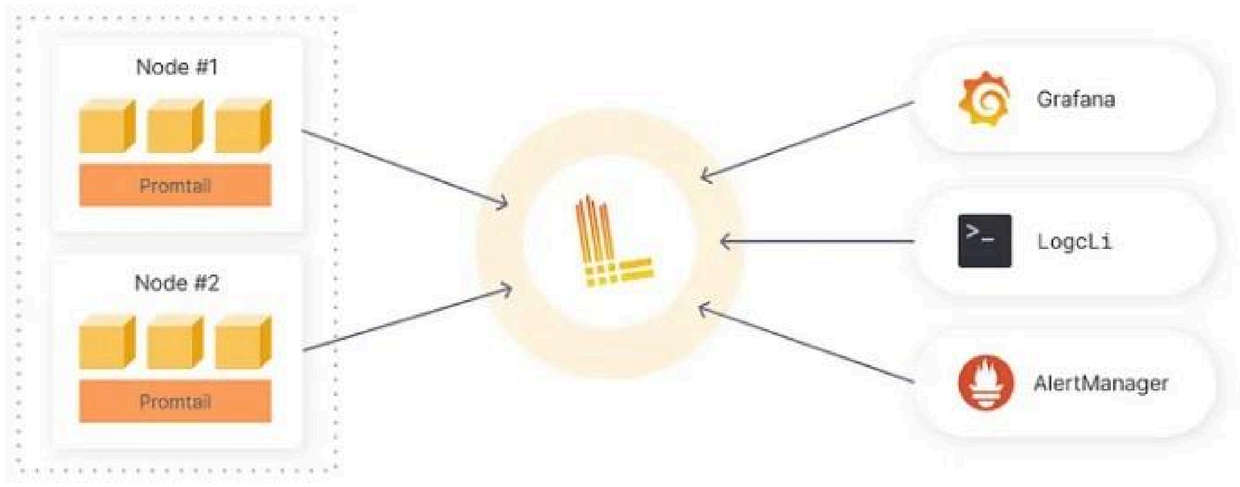
1. **Alerts:** these events are triggered by rules. It can also contain information about the source and target of the attack in the *alert.source* and *alert.target* field.
2. **Stats**: these events are generated display certain states/features of the system.
3. **dns**, which has details pertaining to the source and destination IP addresses as well as information about the data within the packet.

PART THREE - LOKI

Loki: Loki acts as the central log database in your SIEM setup. It receives logs, like Suricata's eve.json, from agents such as Promtail, and stores them efficiently. Loki lets you search and analyze logs using simple queries. Unlike traditional log systems that index all text, which uses lots of resources, Loki indexes only metadata labels, such as *job*, *host*, or *filename*.

After reading this Medium article about Loki, I learned a good deal about the tool. Loki is a log management tool created by Grafana Labs. Loki is made up of three components:

- **Promtail -** an agent that operates in a “push” configuration rather than “pull.” Promtail proactively acquires logs and sends them to Loki, rather than passively waiting for Loki to query it for logs. Multiple Promtail agents can run on a single machine.
- **Loki** - actually responsible for storing the logs. It will only index the labels and metadata of each message, gathering and presenting log data.
- **Grafana** - a visualization system which can query Loki to perform filtering and select the desired results from the logs.



https://grafana.com/oss/loki/

There are a variety of log-types that Loki can collect and analyze. These include:

- **Event Logs**: a high-level record that logs information about network traffic and interactions.
- **Server Logs**: a text file that will contain the records of actions related to a specific server over a defined period of time.