Wazuh - Suricata IDS

Integration of Suricata IDS with Wazuh

Created By: Amir Raza

Follow Me: www.linkedin.com/in/amirsoc

Comprehensive Suricata & Wazuh Integration Guide

Introduction to Suricata and Wazuh

What is Suricata?

Suricata is an open-source network security tool designed for real-time monitoring and protection of network traffic. It serves multiple roles, including:

Network Intrusion Detection System (NIDS)
Intrusion Prevention System (IPS)
Network Security Monitoring (NSM)

Suricata inspects all incoming and outgoing network packets. It looks for signs of attacks, malware, or unusual behavior by using rules and signatures. When it detects any suspicious activity, it generates alerts so that system administrators can take appropriate action. Suricata is known for its high performance, accuracy, and ability to process complex traffic efficiently.

What is Wazuh?

Wazuh is an open-source Security Information and Event Management (SIEM) solution. It is used to collect, analyze, and visualize data from various endpoints such as servers, desktops, applications, and firewalls. The main features of Wazuh include:

Log collection and analysis
Real-time alerting
Security event monitoring

Compliance reporting

Wazuh centralizes security information in a single dashboard, making it easier for organizations to monitor their IT infrastructure, detect threats, and ensure security compliance.

Purpose of Integration

Integrating Suricata with Wazuh allows organizations to combine Suricata's powerful network-level threat detection with Wazuh's centralized log management and analysis capabilities. Suricata sends its alerts and logs to Wazuh, where they are processed and displayed in an organized and user-friendly interface.

Benefits of Integration

The integration of Suricata with Wazuh offers several key advantages:

Centralized Monitoring
All network threat alerts from Suricata can be viewed in one central dashboard within Wazuh, simplifying management and monitoring.
Enhanced Threat Analysis
Wazuh adds context and detail to Suricata alerts, making it easier to understand the severity and nature of threats.
Faster Incident Response
Security teams can detect and respond to threats more quickly and effectively using combined data.
Improved Security Posture
The integration creates a more comprehensive security solution by covering both network-level and endpoint-level threats.

Lab Setup Overview

In this lab setup, we are creating a basic environment to demonstrate the integration of **Suricata** with **Wazuh**. The setup consists of two machines with the following roles:

System Setup

Kali Linux
Acts as the **Wazuh Server** and hosts the **Wazuh Dashboard**. This is where alerts and logs will be collected, analyzed, and visualized.
Ubuntu Linux
Functions as the **Wazuh Agent** and runs the **Suricata IDS**. This machine monitors network traffic and generates security alerts using Suricata.

Lab Goals

The main objectives of this lab are as follows:

Install Suricata on Ubuntu
Set up Suricata IDS on the Ubuntu machine to inspect network traffic.
Configure Suricata to Monitor Traffic
Ensure Suricata is properly configured to analyze real-time network traffic for any suspicious behavior.
Enable JSON Alert Logging
Modify Suricata settings to output alerts in the `eve.json` format, which is structured and easily parsed by Wazuh.
Configure Wazuh Agent
Set up the Wazuh agent on Ubuntu to forward Suricata-generated alerts to the Wazuh server on Kali Linux.
Create Custom Wazuh Rules
Write and implement custom rules in Wazuh to classify, tag, and prioritize alerts generated by Suricata based on severity or type.

Step-by-Step Installation of Suricata (on Ubuntu)

First, Install Suricata IDS by using following command on terminal:

```
sudo apt install suricata -y
```

Suricata is a network security tool that inspects network traffic to detect malicious activity, based on rules/signatures.

```
amir@Ubuntu:~$ sudo apt install suricata -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  isa-support libevent-2.1-7t64 libevent-pthreads-2.1-7t64 libhiredis1.1.0
  libhtp2 libhyperscan5 libluajit-5.1-2 libluajit-5.1-common liblzma-dev
  libnet1 libnetfilter-queue1 sse3-support
```

Verify Suricata installed correctly and show version/build info by following command: suricata --build-info

```
amir@Ubuntu:~$ suricata --build-info
This is Suricata version 7.0.10 RELEASE
Features: NFQ PCAP_SET_BUFF AF_PACKET HAVE_PACKET_FANOUT LIBCAP_NG LIBNET1.1 HAV
E_HTP_URI_NORMALIZE_HOOK PCRE_JIT HAVE_NSS HTTP2_DECOMPRESSION HAVE_LUA HAVE_JA3
 HAVE_JA4 HAVE_LUAJIT HAVE_LIBJANSSON TLS TLS_C11 MAGIC RUST POPCNT64
SIMD support: SSE_2
Atomic intrinsics: 1 2 4 8 byte(s)
64-bits, Little-endian architecture
GCC version 13.3.0, C version 201112
compiled with _FORTIFY_SOURCE=2
L1 cache line size (CLS)=64
thread local storage method: _Thread_local
compiled with LibHTP v0.5.50, linked against LibHTP v0.5.50
```

After installing and configuring Suricata, the next important step is to download a **ruleset** that tells Suricata what to look for in the network traffic.

```
amir@Ubuntu:/tmp$ ^C
amir@Ubuntu:/tmp$ sudo curl -LO https://rules.emergingthreats.net/open/suricata-7.0.10/emerging.rules.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 4863k  100 4863k    0     0  1428k      0  0:00:03  0:00:03 --:--:-- 1428k
amir@Ubuntu:/tmp$
```

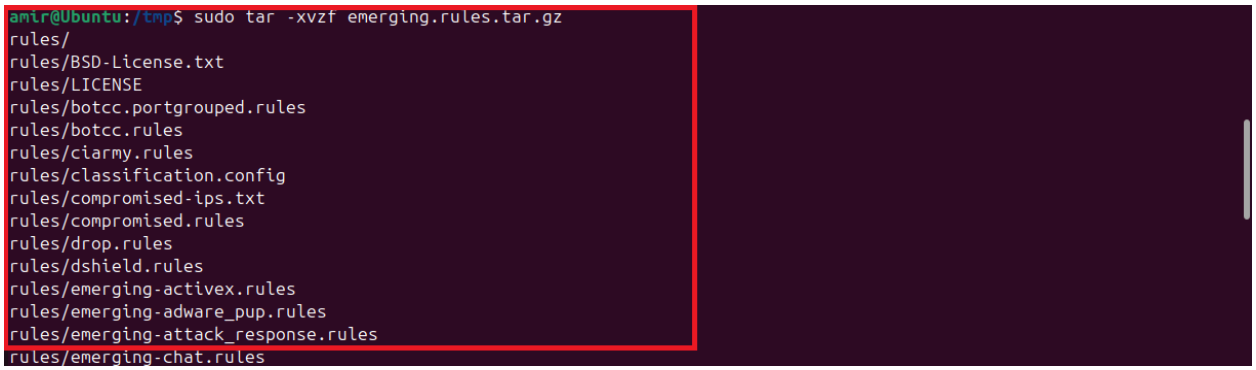
Here is the downloaded file of suricata rules "[emerging.rules.tar.gz]"

```
amir@Ubuntu:/tmp$ ls
emerging.rules.tar.gz
snap-private-tmp
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-apache2.service-omtAnZ
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-colord.service-kCakYV
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-ModemManager.service-QRbfgh
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-polkit.service-Z360TL
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-power-profiles-daemon.service-kOmngR
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-switcheroo-control.service-eoalb6
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-systemd-logind.service-tfxmnl
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-systemd-oomd.service-bbhCyv
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-systemd-resolved.service-p9pXm0
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-systemd-timesyncd.service-b21ML9
systemd-private-53cc1f5ee41242b892dbb077c7c1203f-upower.service-HzAsnq
amir@Ubuntu:/tmp$
```

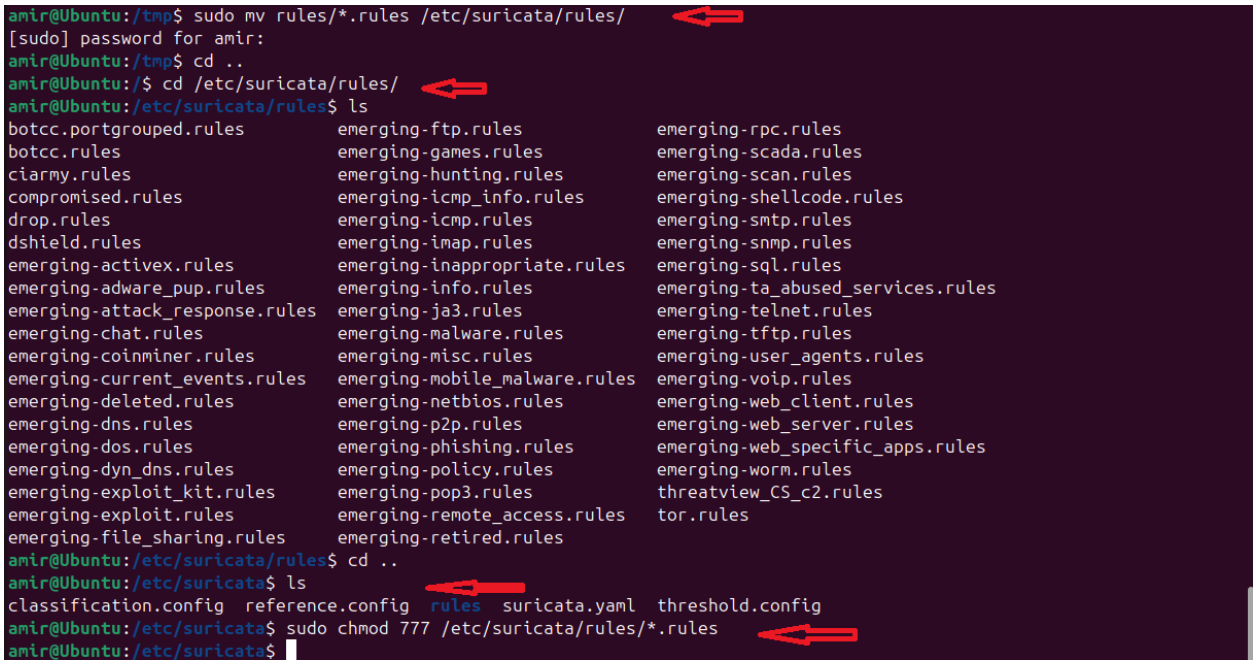
After downloading the Suricata rules file (`emerging.rules.tar.gz`), the next step is to extract it so that the individual rule files can be used by Suricata.

Use the following command to extract the contents of the archive:

sudo tar -xvzf emerging.rules.tar.gz



Now we have to move rules into suricata rules directory and give the permission to all rules. Follow as shown in figure.



Now we must edit the Suricata configuration file to define:

The correct network interface
Ensure the rules and logging are properly configured

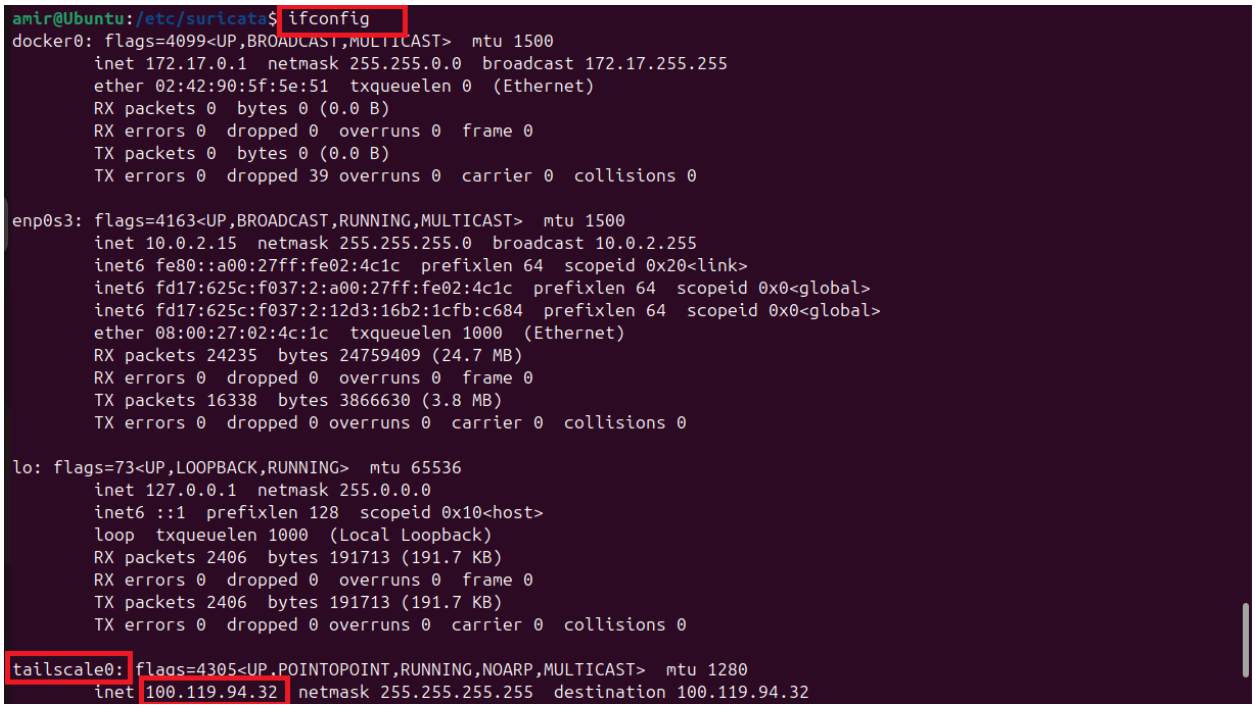
In environments where a virtual machine (VM) uses NAT networking mode and cannot be switched to bridged mode, Suricata cannot directly monitor Wi-Fi or Ethernet network traffic. Since Tailscale VPN is installed and active on the VM, we configure Suricata to monitor VPN traffic instead. This ensures meaningful network traffic inspection despite the VM's networking limitations.

So i use the interface of tailscale vpn and you can use it according to your choice.

Find Your Network Interface Name .Suricata must monitor the correct network interface to capture traffic.
Wrong interface means no or wrong traffic captured.

So use this to checks network interface:

ifconfig



Now we have to configure suricata according to our network setting

Open the main Suricata configuration file

sudo nano /etc/suricata/suricata.yaml

```
 GNU nano 7.2                                    suricata.yaml
---

# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://docs.suricata.io/en/latest/configuration/suricata-yaml.html

# This configuration file generated by Suricata 7.0.10.
suricata-version: "7.0"

##
## Step 1: Inform Suricata about your network
##

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[100.64.0.0/10]"
   # HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    #EXTERNAL_NET: "!$HOME_NET"
    EXTERNAL_NET: "any"
```

Find the section that looks like this (or add if missing):

af-packet:

- interface: enp0s3 # Replace 'enp0s3' with your actual interface name

  threads: auto

  cluster-id: 99

  cluster-type: cluster_flow

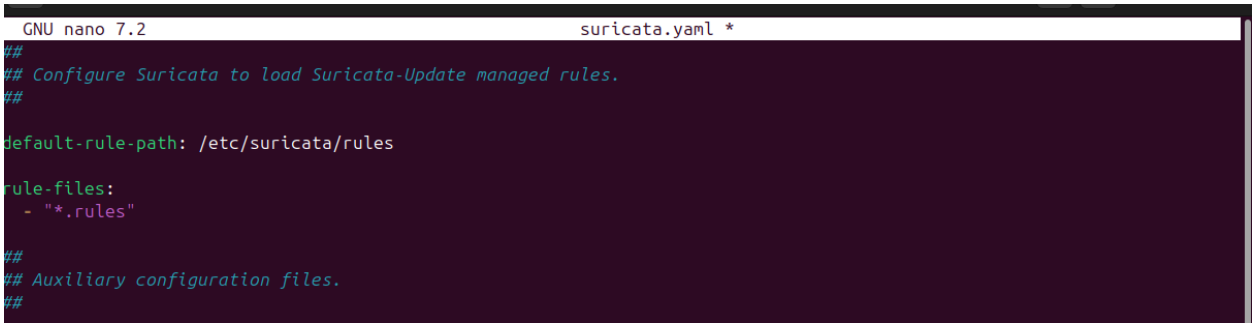
```
af-packet:
  - interface: tailscale0
    #interface: enp0s3
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
    cluster-type: cluster_flow
    # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per hash.
    # This is only supported for Linux kernel > 3.1
    # possible value are:
```

Purpose:

The af-packet section tells Suricata which network interface to sniff and how. interface specifies the exact NIC.

threads: auto uses all CPU cores for processing traffic efficiently.
cluster-id and cluster-type optimize packet capture for performance.

Now scroll down and edit "rule-files:" section. Follow same as shown in figure.



And now save all configuration by press ctrl+o ,Enter, and then ctrl+x.

Start Suricata Service and Enable at Boot

sudo systemctl enable suricata

sudo systemctl start suricata

Verify Suricata is Running

sudo systemctl status suricata

Purpose:

Check that Suricata service is active (running) without errors.
If failed or inactive, troubleshoot before proceeding.

```
amir@Ubuntu:/etc/suricata$ sudo systemctl enable suricata
suricata.service is not a native service, redirecting to systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable suricata
amir@Ubuntu:/etc/suricata$ sudo systemctl status suricata
● suricata.service - LSB: Next Generation IDS/IPS
     Loaded: loaded (/etc/init.d/suricata; generated)
     Active: active (running) since Thu 2025-06-26 10:17:22 UTC; 1h 30min ago
       Docs: man:systemd-sysv-generator(8)
      Tasks: 22 (limit: 4545)
     Memory: 535.3M (peak: 556.6M)
        CPU: 1min 54.073s
     CGroup: /system.slice/suricata.service
             └─1878 /usr/bin/suricata -c /etc/suricata/suricata.yaml --pidfile /var/run/suricata.pid --af-packet -D -vvv

Jun 26 10:17:21 Ubuntu systemd[1]: Starting suricata.service - LSB: Next Generation IDS/IPS...
Jun 26 10:17:22 Ubuntu suricata[1715]: Starting suricata in IDS (af-packet) mode... done.
Jun 26 10:17:22 Ubuntu systemd[1]: Started suricata.service - LSB: Next Generation IDS/IPS.
amir@Ubuntu:/etc/suricata$ sudo systemctl restart suricata
amir@Ubuntu:/etc/suricata$
```

Now we have to configure suricata logs into Wazuh-agent "ossec.conf" file and add there location of file and its location is here:

```
amir@Ubuntu:/var/log/suricata$ ls
certs  core  eve.json  fast.log  files  stats.log  suricata.log  suricata-start.log
amir@Ubuntu:/var/log/suricata$
```

Configure Wazuh Agent to Monitor Suricata Logs

Open Wazuh agent config file:

sudo nano /var/ossec/etc/ossec.conf

Add this block inside `<ossec_config>` but outside other `<localfile>` blocks:

Purpose:

Tell Wazuh agent to watch the Suricata JSON alert log file for new entries.
The agent reads this file, sends events to Wazuh server for analysis and alerting.

```
 GNU nano 7.2                                  /var/ossec/etc/ossec.conf
    <log_format>syslog</log_format>
    <location>/var/ossec/logs/active-responses.log</location>
  </localfile>
  <localfile>
  <log_format>syslog</log_format>
  <location>/var/log/auth.log</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/dpkg.log</location>
  </localfile>
  <localfile>
  <log_format>json</log_format>
  <location>/var/log/suricata/eve.json</location>
  </localfile>

</ossec_config>
```

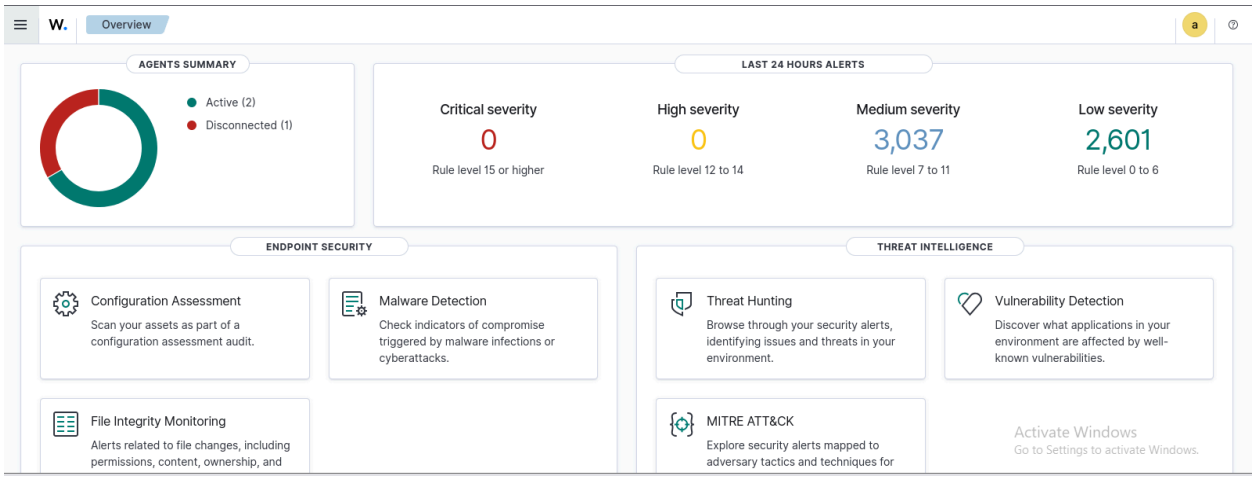
Restart Wazuh Agent and Suricata to Apply Changes and checks its status:

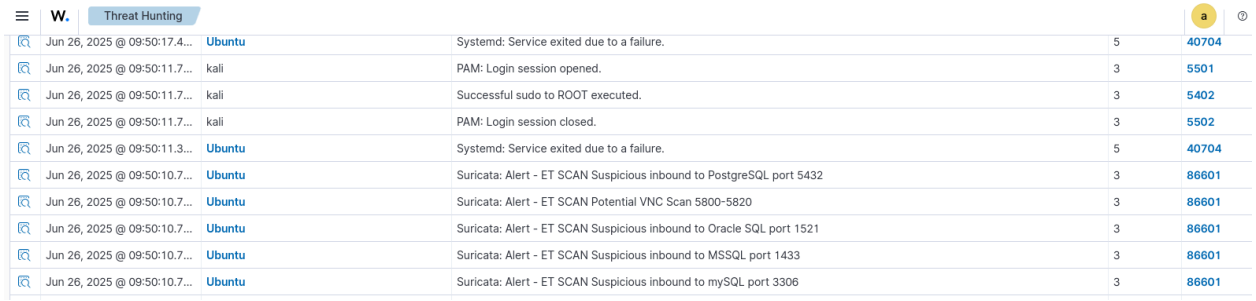
```
amir@Ubuntu:/$ sudo systemctl restart suricata
amir@Ubuntu:/$ sudo systemctl restart wazuh-agent
amir@Ubuntu:/$ sudo systemctl status wazuh-agent
● wazuh-agent.service - Wazuh agent
     Loaded: loaded (/usr/lib/systemd/system/wazuh-agent.service; enabled; preset: enabled)
     Active: active (running) since Thu 2025-06-26 13:10:04 UTC; 1min 13s ago
    Process: 35242 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
      Tasks: 35 (limit: 4545)
     Memory: 533.7M (peak: 554.7M)
        CPU: 13.351s
     CGroup: /system.slice/wazuh-agent.service
             ├─35265 /var/ossec/bin/wazuh-execd
             ├─35276 /var/ossec/bin/wazuh-agentd
             ├─35292 /var/ossec/bin/wazuh-syscheckd
             ├─35302 /var/ossec/bin/wazuh-logcollector
             └─35316 /var/ossec/bin/wazuh-modulesd

Jun 26 13:09:58 Ubuntu env[35252]: 2025/06/26 13:09:58 wazuh-syscheckd: WARNING: (1230): Invalid element in the configu>
Jun 26 13:09:59 Ubuntu env[35242]: Starting Wazuh v4.11.2...
Jun 26 13:10:00 Ubuntu env[35242]: Started wazuh-execd...
Jun 26 13:10:01 Ubuntu env[35242]: Started wazuh-agentd...
Jun 26 13:10:01 Ubuntu env[35290]: 2025/06/26 13:10:01 wazuh-syscheckd: WARNING: (1230): Invalid element in the configu>
Jun 26 13:10:01 Ubuntu env[35242]: Started wazuh-syscheckd...
Jun 26 13:10:02 Ubuntu env[35242]: Started wazuh-logcollector...
Jun 26 13:10:02 Ubuntu env[35242]: Started wazuh-modulesd...
Jun 26 13:10:04 Ubuntu env[35242]: Completed.
Jun 26 13:10:04 Ubuntu systemd[1]: Started wazuh-agent.service - Wazuh agent.
```

To verify that Suricata is correctly monitoring network traffic and generating alerts, we will perform a controlled **Nmap scan** from an attacker machine to the Suricata-monitored machine (the target). This is a commonly used technique to test intrusion detection functionality.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sT 100.119.94.32
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-26 09:41 EDT
Nmap scan report for ubuntu.taild44d1c.ts.net (100.119.94.32)
Host is up (0.0026s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.38 seconds
```

Now go to wazuh dashboard and see the Events.



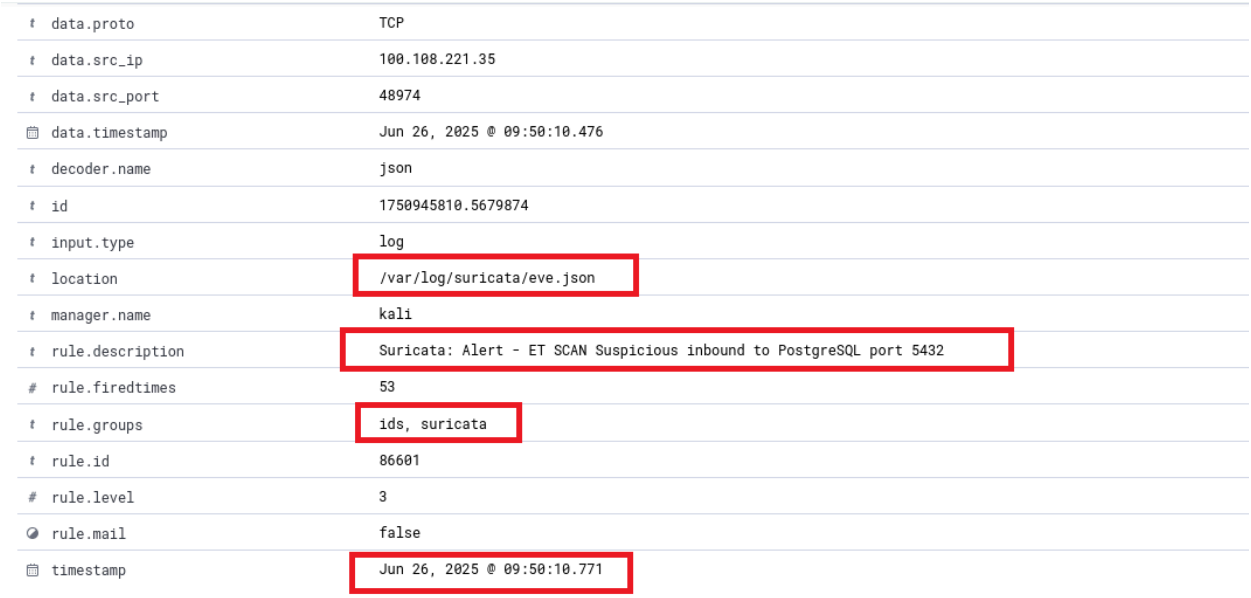


Jun 26, 2025 @ 09:50:17.4...	Ubuntu	Systemd: Service exited due to a failure.	5	40704
Jun 26, 2025 @ 09:50:11.7...	kali	PAM: Login session opened.	3	5501
Jun 26, 2025 @ 09:50:11.7...	kali	Successful sudo to ROOT executed.	3	5402
Jun 26, 2025 @ 09:50:11.7...	kali	PAM: Login session closed.	3	5502
Jun 26, 2025 @ 09:50:11.3...	Ubuntu	Systemd: Service exited due to a failure.	5	40704
Jun 26, 2025 @ 09:50:10.7...	Ubuntu	Suricata: Alert - ET SCAN Suspicious inbound to PostgreSQL port 5432	3	86601
Jun 26, 2025 @ 09:50:10.7...	Ubuntu	Suricata: Alert - ET SCAN Potential VNC Scan 5800-5820	3	86601
Jun 26, 2025 @ 09:50:10.7...	Ubuntu	Suricata: Alert - ET SCAN Suspicious inbound to Oracle SQL port 1521	3	86601
Jun 26, 2025 @ 09:50:10.7...	Ubuntu	Suricata: Alert - ET SCAN Suspicious inbound to MSSQL port 1433	3	86601
Jun 26, 2025 @ 09:50:10.7...	Ubuntu	Suricata: Alert - ET SCAN Suspicious inbound to mySQL port 3306	3	86601

Table JSON

@timestamp	Jun 26, 2025 @ 09:50:10.771
_index	wazuh-alerts-4.x-2025.06.26
agent.id	003
agent.ip	10.0.2.15
agent.name	Ubuntu
data.alert.action	allowed
data.alert.category	Potentially Bad Traffic
data.alert.gid	1
data.alert.metadata.confidence	Medium
data.alert.metadata.created_at	2010_07_30
data.alert.metadata.signature_severity	Informational
data.alert.metadata.updated_at	2019_07_26
data.alert.rev	3
data.alert.severity	2
data.alert.signature	ET SCAN Suspicious inbound to PostgreSQL port 5432

data.dest_ip	100.119.94.32
data.dest_port	5432
data.direction	to_server
data.event_type	alert
data.flow.bytes_toclient	0
data.flow.bytes_toserver	60
data.flow.dest_ip	100.119.94.32
data.flow.dest_port	5432
data.flow.pkts_toclient	0
data.flow.pkts_toserver	1
data.flow.src_ip	100.108.221.35
data.flow.src_port	48974
data.flow.start	2025-06-26T13:50:10.476754+0000
data.flow_id	640270320386932.000000
data.in_iface	tailscale0
data.pkt_src	wire/pcap
data.proto	TCP



data.proto	TCP
data.src_ip	100.108.221.35
data.src_port	48974
data.timestamp	Jun 26, 2025 @ 09:50:10.476
decoder.name	json
id	1750945810.5679874
input.type	log
location	/var/log/suricata/eve.json
manager.name	kali
rule.description	Suricata: Alert - ET SCAN Suspicious inbound to PostgreSQL port 5432
rule.firedtimes	53
rule.groups	ids, suricata
rule.id	86601
rule.level	3
rule.mail	false
timestamp	Jun 26, 2025 @ 09:50:10.771

Summary:

This documentation details the integration of Suricata IDS with Wazuh SIEM in a virtualized environment using a Tailscale VPN interface for network traffic monitoring due to NAT networking limitations.

Suricata was installed and configured on an Ubuntu VM to monitor the `tailscale0` interface with customized network settings (`HOME_NET` set to the VPN IP range). Community rule sets were downloaded, extracted, and properly integrated, including a custom test rule to verify detection.

Wazuh agent was configured to monitor Suricata's JSON alert log (`eve.json`), enabling centralized collection and visualization of network security events on the Kali Linux Wazuh server.

Network scanning tests using `nmap` and ICMP confirmed Suricata's ability to detect and log relevant traffic, and firewall rules were implemented to restrict open ports to HTTP only, enhancing the security posture.

This setup provides an effective solution for network intrusion detection and centralized security monitoring in environments where direct network bridging is unavailable.