SENTINEL PI

A Raspberry Pi–Based Intrusion Detection System with Physical Alerts and Cloud SIEM Integration

---

Project by: Khushi

Project type:- Cyber physical system project

Project Overview

Sentinel Pi is a lightweight, low-cost Network Intrusion Detection System (NIDS) implemented on a Raspberry Pi. The system monitors live network traffic using the Suricata IDS engine to detect suspicious activities such as ICMP probing, port scanning, and OS fingerprinting. On detecting high-severity threats, Sentinel Pi triggers physical alerts via GPIO-connected hardware and forwards structured JSON logs to a cloud-based SIEM platform (Wazuh) for centralized monitoring and analysis, simulating real-world SOC workflows.

Technologies Used

Raspberry Pi OS · Suricata IDS · Python · GPIO · Linux Networking · Wazuh SIEM

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Background

With the rapid growth of computer networks, IoT devices, and internet-connected systems, networks have become increasingly vulnerable to cyber-attacks. Even small-scale environments such as home networks, laboratories, and educational setups are frequently targeted through techniques like port scanning, ping sweeps, and unauthorized access attempts. These attacks often go unnoticed due to the absence of continuous monitoring mechanisms.

Traditional enterprise-grade security solutions and Security Information and Event Management (SIEM) platforms are expensive and resource-intensive, making them unsuitable for small or budget-constrained environments. As a result, there is a growing need for lightweight, low-cost intrusion detection solutions that can provide real-time visibility into network activity while remaining practical and accessible.

1.2 Problem Statement

Most small networks and standalone systems operate without any form of real-time intrusion detection. Common issues include the absence of packet inspection, lack of immediate alerting mechanisms, and no centralized logging or monitoring. When suspicious activity occurs, users often become aware only after damage has already been done.

Additionally, software-only alerts such as logs or console messages may go unnoticed, especially in offline or unattended environments. There is a need for a system that not only detects malicious network activity but also provides an immediate physical alert and centralized monitoring capability.

1.3 Motivation

The motivation behind developing *Sentinel Pi* was to gain hands-on experience in network security, intrusion detection systems, and real-world security monitoring workflows. This project aims to bridge the gap between theoretical cybersecurity concepts and practical implementation by deploying an IDS on resource-constrained hardware.

By combining network traffic analysis, automation through scripting, hardware-based alerts, and cloud-based SIEM integration, Sentinel Pi demonstrates how multiple security domains can be integrated into a single, functional system. The project also focuses on understanding how professional Security Operations Centers (SOCs) monitor and analyze security events, even in minimal hardware environments.

1.4 Project Objectives

The primary objectives of the Sentinel Pi project are:

- To deploy a network-based Intrusion Detection System on a Raspberry Pi.
- To monitor live network traffic and detect common attack patterns in real time.
- To generate immediate physical alerts using GPIO-connected hardware upon detection of high-severity threats.
- To log detected security events in structured JSON format.
- To forward intrusion alerts to a cloud-based SIEM platform for centralized monitoring and analysis.
- To simulate real-world SOC-style security monitoring on low-cost hardware.

1.5 Scope of the Project

The scope of Sentinel Pi includes real-time network traffic monitoring, intrusion detection using signature-based rules, physical alert generation, and cloud-based log visualization. The system is designed for small-scale networks, learning environments, and demonstration purposes.

The project does not focus on active intrusion prevention, automatic blocking of attackers, machine learning–based anomaly detection, or large enterprise-scale deployments. These aspects are considered outside the current scope and are identified as potential future enhancements.

2. LITERATURE SURVEY

2.1 Intrusion Detection Systems (IDS)

An Intrusion Detection System (IDS) is a security mechanism designed to monitor network or system activities for malicious behavior or policy violations. IDS solutions analyze traffic patterns, system logs, and events to identify potential security threats. IDS can broadly be classified into Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS).

Network-based IDS solutions inspect packets flowing through a network segment and compare them against predefined rules or behavioral patterns. Signature-based IDS detect known attack patterns, while anomaly-based IDS identify deviations from normal traffic behavior. Hybrid approaches combine both techniques to improve detection accuracy. IDS plays a critical role in identifying reconnaissance activities such as port scanning, ping sweeps, and unauthorized access attempts.

2.2 Suricata Intrusion Detection System

Suricata is an open-source, high-performance Network Intrusion Detection and Prevention System (NIDS/NIPS) widely used in modern cybersecurity environments. It supports deep packet inspection, protocol identification, and multi-threaded packet processing, making it suitable for real-time network monitoring.

One of the key features of Suricata is its ability to generate structured event logs in JSON format (EVE JSON), which simplifies integration with Security Information and Event Management (SIEM) platforms. Suricata supports rule-based detection using Emerging Threats (ET) rulesets, allowing detection of common network attacks such as ICMP probing, TCP/UDP scans, and operating system fingerprinting. Due to its flexibility and open-source nature, Suricata is commonly adopted in research, enterprise, and educational security projects.

2.3 Raspberry Pi in Network Security Applications

The Raspberry Pi is a low-cost, compact single-board computer widely used for educational and experimental projects. Despite its limited hardware resources compared to enterprise servers, it is capable of running lightweight security tools and monitoring applications.

In network security research and learning environments, Raspberry Pi devices have been used for intrusion detection systems, honeypots, firewalls, and network monitoring solutions. Its small form factor, low power consumption, and support for Linux-based operating systems make it suitable for portable and cost-effective security deployments. However, hardware constraints such as limited CPU and memory require careful configuration and optimization when running IDS tools.

2.4 Security Information and Event Management (SIEM)

Security Information and Event Management (SIEM) platforms collect, correlate, and analyze security events from multiple sources to provide centralized visibility into an organization's security posture. SIEM solutions help security teams detect threats, investigate incidents, and maintain compliance through dashboards, alerts, and reports.

Wazuh is an open-source SIEM and security monitoring platform that supports log analysis, intrusion detection integration, and real-time alerting. By forwarding IDS-generated logs to a SIEM platform, security events can be visualized and analyzed in a manner similar to professional Security Operations Centers (SOCs). This centralized approach enhances situational awareness and simplifies security monitoring across distributed systems.

2.5 Physical Alert Mechanisms in Security Systems

Traditional IDS implementations primarily rely on software-based alerts such as logs, emails, or dashboards. However, in offline or unattended environments, these alerts may go unnoticed. Integrating physical alert mechanisms such as buzzers or LEDs enables immediate local notification when a security event is detected.

In IoT and embedded security applications, physical alerts provide an additional layer of awareness by converting digital security events into real-world signals. Such integrations improve responsiveness and make IDS solutions more practical for small-scale or isolated deployments.

3. WORK DONE

3.1 System Architecture

The Sentinel Pi system is designed as a lightweight and modular Intrusion Detection System that integrates network traffic monitoring, alert processing, physical notification, and centralized logging.

The overall system architecture details the flow of network traffic through Suricata, log parsing, and the parallel physical and cloud-based alerting mechanisms.
The process begins with Network Traffic being inspected by the Suricata IDS running on the Raspberry Pi. Upon detection, an EVE JSON alert is generated, which is consumed by a Python Alert Parser Script. This script manages the Log Forwarding to the Wazuh SIEM and also controls the GPIO Trigger Logic to activate the buzzer or LEDs.

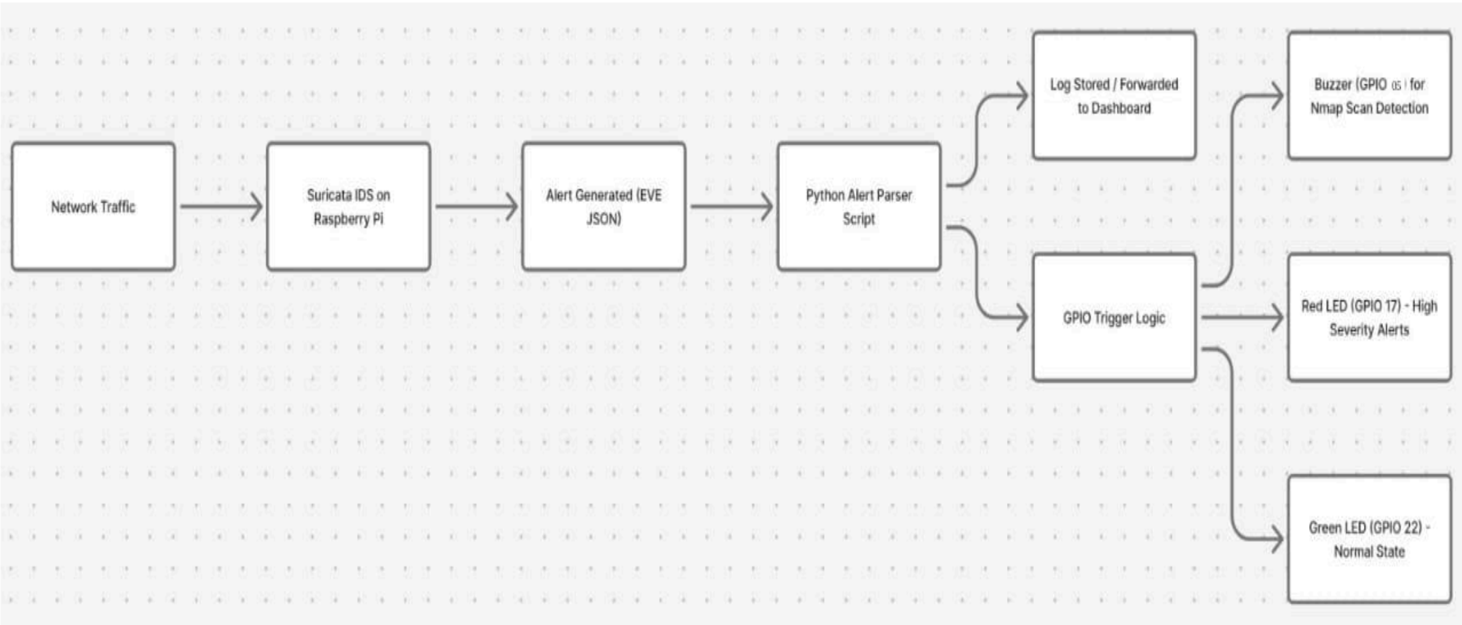


Figure 3.1 Raspberry Pi IDS System Flowchart

3.2 Hardware Setup

The following hardware components were used in the implementation of Sentinel Pi:

- Raspberry Pi 5
- Ethernet connectivity for network monitoring
- Active internet connection (for Wazuh Cloud forwarding)
- Buzzer and Resistors connected to the Raspberry Pi's GPIO pins
- SSD card
- Python environment installed

The physical wiring diagram below illustrates the connection between the Raspberry Pi's GPIO pins and the components on the breadboard.

- Buzzer: Connected to Pin 5 / GPIO 5 for Nmap scan alerts.
- Red LED: Connected to Pin 11 / GPIO 17 for high-severity alerts.
- Green LED: Connected to GPIO 22 for normal system state indication.
- GND pins provide the common ground for the circuit.

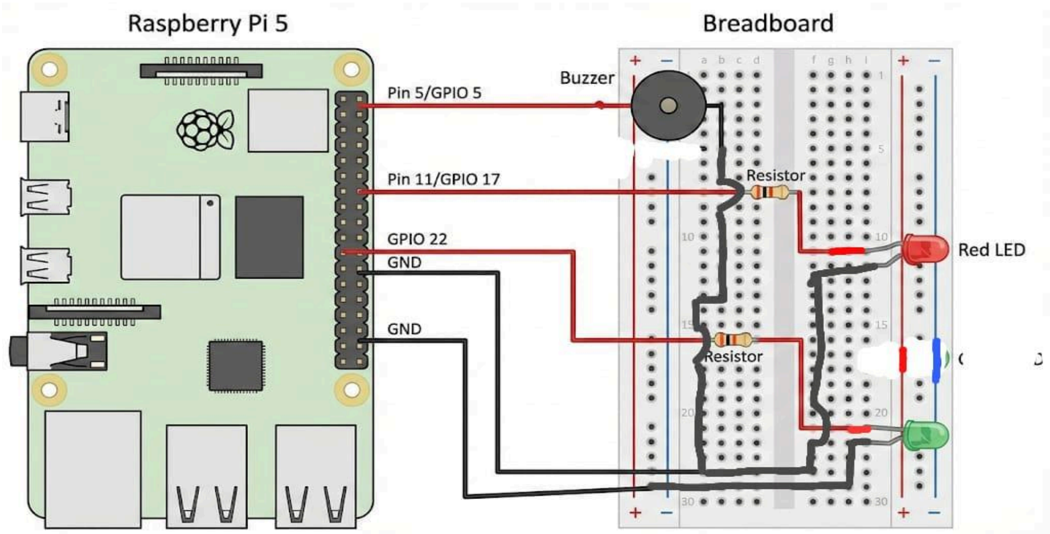


Figure 3.2.1 Hardware Circuit Diagram

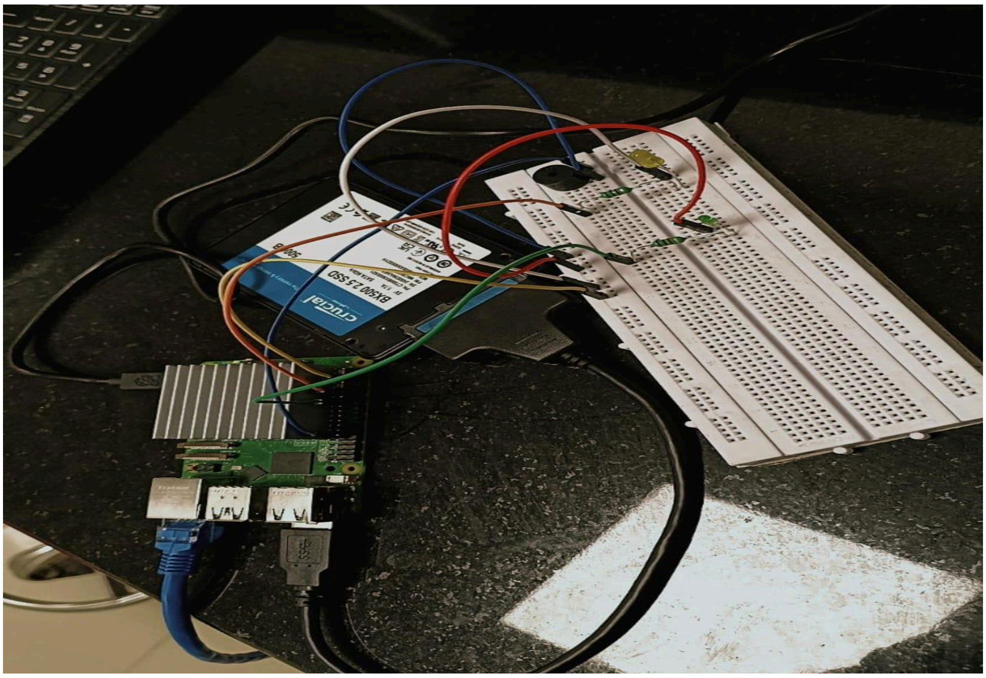


Figure 3.2.2 Real Hardware setup

The Raspberry Pi functions as the core IDS engine, while the buzzer and LEDs provide immediate physical feedback when intrusion events are detected.

3.3 Raspberry Pi OS Installation

The Raspberry Pi was prepared by installing Raspberry Pi OS on SSD card to provide a stable Linux-based operating environment for deploying the intrusion detection system. Raspberry Pi OS was selected due to its compatibility with Raspberry Pi hardware and support for security and networking tools.

The operating system image was written to the SSD card using an official imaging utility. After flashing, the Raspberry Pi was booted and initial system configuration was completed. This included expanding the filesystem, setting system locale and time zone, and updating system packages to ensure the latest security patches were applied.

Updating the system ensured a clean and secure baseline before installing intrusion detection and monitoring tools.

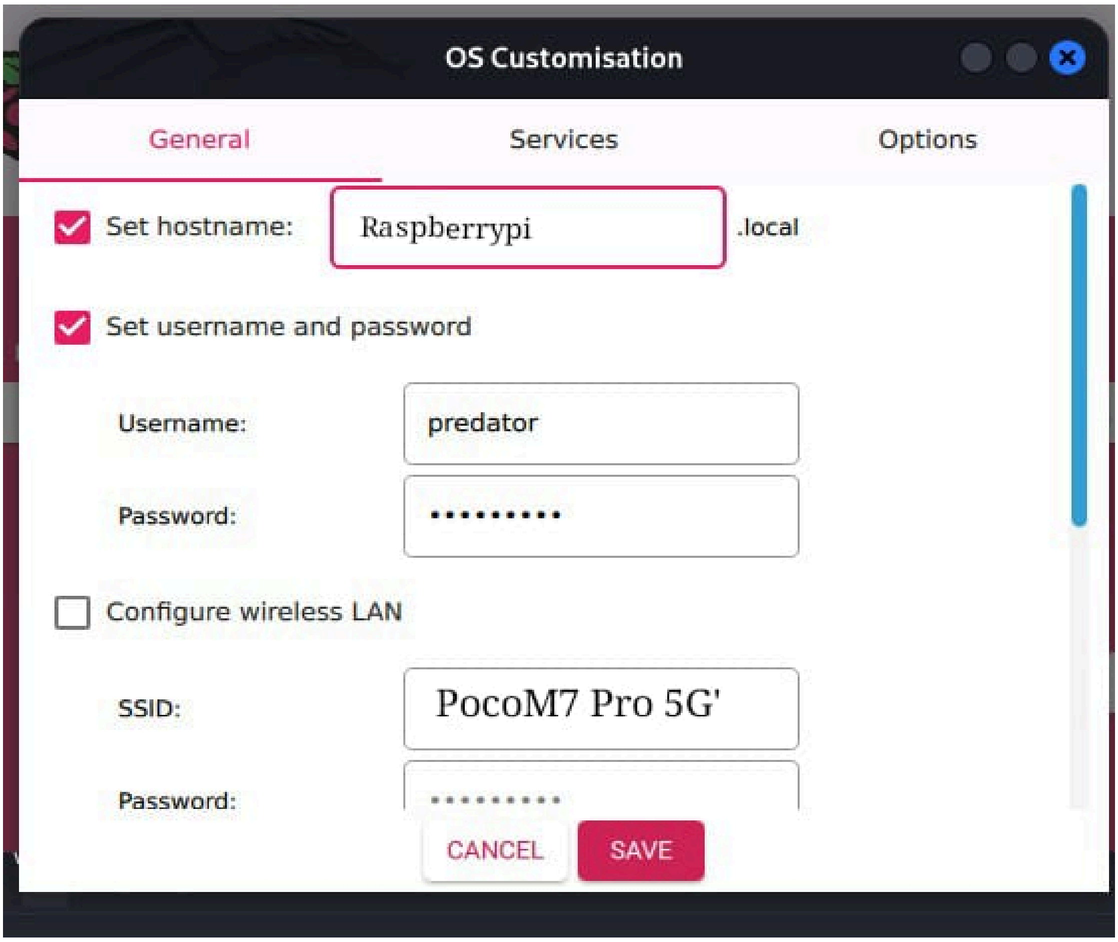


Figure 3.3.1 Raspberry Pi OS Customization

```
┌──(predator㉿kali)-[~]
└─$ ssh predator@raspberrypi.local
The authenticity of host 'raspberrypi.local (192.168.47.11)' can't be established.
ED25519 key fingerprint is SHA256:agfwwsEQEoKyXbvLwN1zVfcOQa7rPjP7VMprIdJqMgY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'raspberrypi.local' (ED25519) to the list of known hosts.
predator@raspberrypi.local's password:
Linux raspberrypi 6.12.47+rpt-rpi-2712 #1 SMP PREEMPT Debian 1:6.12.47-1+rpt1 (2025-09-16) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Nov 21 09:58:40 2025 from 2409:40d5:59:b48a:4121:4663:47c2:df49
```

Figure 3.3.2 ssh successful into Raspberry Pi

```
predator@raspberrypi:~ $ sudo apt update && sudo apt upgrade -y
Hit:1 http://archive.raspberrypi.com/debian trixie InRelease
Hit:2 http://deb.debian.org/debian trixie InRelease
Hit:3 http://deb.debian.org/debian trixie-updates InRelease
Hit:4 http://deb.debian.org/debian-security trixie-security InRelease
201 packages can be upgraded. Run 'apt list --upgradable' to see them.
Upgrading:
  7zip                            libpoppler-glib8t64       rasputin
  agnostics                       libpoppler147             rc-gui
  base-files                      libpostproc58             rp-bookshelf
  bluez-firmware                  libqt6core6t64            rp-prefapps
  chromium                        libqt6dbus6               rpcc
  chromium-common                 libqt6gui6                rpd-applications
  chromium-l10n                   libqt6network6            rpd-common
  chromium-sandbox                libqt6opengl6             rpd-developer
  console-setup                   libqt6openglwidgets6      rpd-graphics
  console-setup-linux             libqt6widgets6            rpd-plym-splash
  cups                            librpicam-app1            rpd-preferences
  cups-client                     libsmbclient0             rpd-theme
  cups-common                     libssh-4                  rpd-utilities
  cups-core-drivers               libssl3t64                rpd-wayland-core
  cups-daemon                     libswresample5            rpd-wayland-extras
  cups-ipp-utils                  libswscale8               rpd-x-core
  cups-ppdc                       libsystemd-shared         rpd-x-extras
  cups-server-common              libsystemd0               rpi-chromium-mods
  curl                            libtalloc2                rpi-connect
  dhcpcd-base                     libtbb12                  rpi-eeprom
  distro-info-data                libtbbbind-2-5            rpi-firefox-mods
  dns-root-data                   libtbbmalloc2             rpi-loop-utils
  ffmpeg                          libtdb1                   rpi-swap
  firefox                         libtevent0t64             rpi-userguide
  firmware-atheros                libtiff6                  rpicam-apps
  firmware-brcm80211              libudev1                  rpicam-apps-core
  firmware-libertas               libvlc-bin                rpicam-apps-encoder
  firmware-mediatek               libvlc5                   rpicam-apps-lite
  firmware-realtek                libvlccore9               rpicam-apps-opencv-postprocess
  ghostscript                     libwbclient0              rpicam-apps-preview
  gpiod                           libwtmpdb0                rpinters
  gui-pkinst                      libxml2                   samba-libs
  gui-updater                     lpplug-bluetooth          systemd
  imagemagick-7-common            lpplug-clock              systemd-sysv
  keyboard-configuration          lpplug-ejecter            systemd-timesyncd
  libavcodec61                    lpplug-menu               udev
  libavdevice61                   lpplug-netman             userconf-pi
  libavfilter10                   lpplug-power              vlc
  libavformat61                   lpplug-updater            vlc-bin
  libavutil59                     lpplug-volumepulse        vlc-data
  libcamera-ipa                   lxpanel-pi                vlc-l10n
  libcamera-tools                 mesa-libgallium           vlc-plugin-access-extra
  libcamera-v4l2                  mesa-vulkan-drivers       vlc-plugin-base
  libcamera0.5                    openssl                   vlc-plugin-notify
  libcups2t64                     openssl-provider-legacy   vlc-plugin-qt
  libcupsimage2t64                pipanel                   vlc-plugin-samba
```

Figure 3.3.3 Raspberry Pi updated successfully

3.4 Network Configuration and Static IP Assignment

For reliable intrusion detection and testing, consistent network addressing was required. A static IP configuration was applied to ensure stable communication between the Raspberry Pi and the external machine used for attack simulation.

The Ethernet interface of the external laptop was assigned a static IP address within the same subnet as the Raspberry Pi. This ensured predictable connectivity during scanning and testing activities and eliminated issues caused by dynamic IP address changes.

Using a static IP configuration allowed accurate identification of source and destination IP addresses in Suricata alert logs and simplified correlation of intrusion events during analysis.

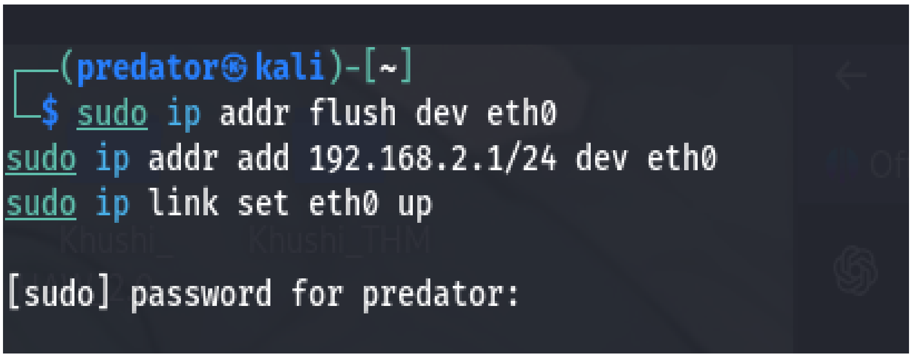


Figure 3.4 Assigned static ip to laptop

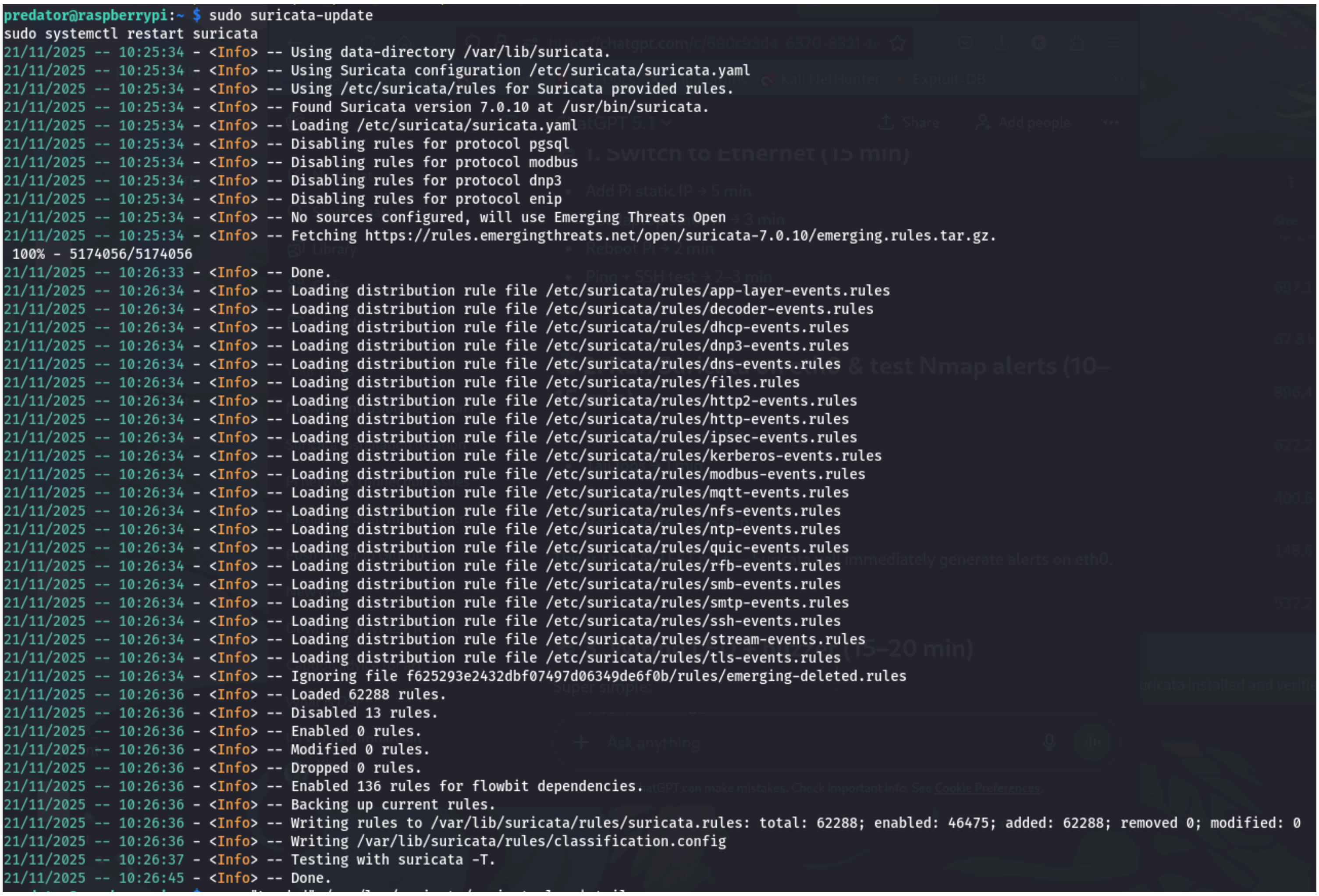
3.5 Installing Suricata

The latest stable version of Suricata was installed on the Raspberry Pi OS using the following commands:

BASH SCRIPT :
sudo apt install suricata
sudo systemctl enable suricata
sudo systemctl start suricata

```
predator@raspberrypi:~ $ sudo apt install suricata -y
Installing:
  suricata

Installing dependencies:
  armv8crc-support  libnetfilter-log1    librte-hash25     librte-meter25     librte-sched25
  isa-support       libnetfilter-queue1  librte-ip-frag25  librte-net-bond25  librte-telemetry25
  libhiredis1.1.0   librte-bus-pci25     librte-kvargs25   librte-net25       libxdp1
  libhtp2           librte-bus-vdev25    librte-log25      librte-pci25       suricata-update
  libmaxminddb0     librte-eal25         librte-mbuf25     librte-rcu25
  libnet1           librte-ethdev25      librte-mempool25  librte-ring25

Suggested packages:
  mmdb-bin  libtcmalloc-minimal4

Recommended packages:
  snort-rules-default

Summary:
  Upgrading: 0, Installing: 29, Removing: 0, Not Upgrading: 0
  Download size: 3,679 kB
  Space needed: 13.7 MB / 464 GB available

Get:1 http://deb.debian.org/debian trixie/main arm64 isa-support arm64 27 [7,312 B]
Get:2 http://deb.debian.org/debian trixie/main arm64 armv8crc-support arm64 27 [3,620 B]
Get:3 http://deb.debian.org/debian trixie/main arm64 libhiredis1.1.0 arm64 1.2.0-6+b3 [48.5 kB]
Get:4 http://deb.debian.org/debian trixie/main arm64 libhtp2 arm64 1:0.5.50-1+deb13u1 [67.6 kB]
Get:5 http://deb.debian.org/debian trixie/main arm64 libmaxminddb0 arm64 1.12.2-1 [25.9 kB]
Get:6 http://deb.debian.org/debian trixie/main arm64 libnet1 arm64 1.3+dfsg-2 [51.4 kB]
Get:7 http://deb.debian.org/debian trixie/main arm64 libnetfilter-log1 arm64 1.0.2-4+b1 [12.7 kB]
Get:8 http://deb.debian.org/debian trixie/main arm64 libnetfilter-queue1 arm64 1.0.5-4+b1 [13.9 kB]
Get:9 http://deb.debian.org/debian trixie/main arm64 librte-log25 arm64 24.11.3-1~deb13u1 [23.9 kB]
Get:10 http://deb.debian.org/debian trixie/main arm64 librte-kvargs25 arm64 24.11.3-1~deb13u1 [17.9 kB]
Get:11 http://deb.debian.org/debian trixie/main arm64 librte-telemetry25 arm64 24.11.3-1~deb13u1 [26.8 kB]
Get:12 http://deb.debian.org/debian trixie/main arm64 librte-eal25 arm64 24.11.3-1~deb13u1 [146 kB]
Get:13 http://deb.debian.org/debian trixie/main arm64 librte-ring25 arm64 24.11.3-1~deb13u1 [20.1 kB]
Get:14 http://deb.debian.org/debian trixie/main arm64 librte-mempool25 arm64 24.11.3-1~deb13u1 [31.4 kB]
Get:15 http://deb.debian.org/debian trixie/main arm64 librte-mbuf25 arm64 24.11.3-1~deb13u1 [28.8 kB]
Get:16 http://deb.debian.org/debian trixie/main arm64 librte-meter25 arm64 24.11.3-1~deb13u1 [17.3 kB]
Get:17 http://deb.debian.org/debian trixie/main arm64 librte-net25 arm64 24.11.3-1~deb13u1 [21.8 kB]
Get:18 http://deb.debian.org/debian trixie/main arm64 librte-ethdev25 arm64 24.11.3-1~deb13u1 [114 kB]
Get:19 http://deb.debian.org/debian trixie/main arm64 librte-pci25 arm64 24.11.3-1~deb13u1 [17.7 kB]
Get:20 http://deb.debian.org/debian trixie/main arm64 librte-bus-pci25 arm64 24.11.3-1~deb13u1 [34.4 kB]
Get:21 http://deb.debian.org/debian trixie/main arm64 librte-bus-vdev25 arm64 24.11.3-1~deb13u1 [21.8 kB]
Get:22 http://deb.debian.org/debian trixie/main arm64 librte-rcu25 arm64 24.11.3-1~deb13u1 [22.0 kB]
Get:23 http://deb.debian.org/debian trixie/main arm64 librte-hash25 arm64 24.11.3-1~deb13u1 [45.5 kB]
Get:24 http://deb.debian.org/debian trixie/main arm64 librte-ip-frag25 arm64 24.11.3-1~deb13u1 [33.4 kB]
Get:25 http://deb.debian.org/debian trixie/main arm64 librte-sched25 arm64 24.11.3-1~deb13u1 [32.5 kB]
Get:26 http://deb.debian.org/debian trixie/main arm64 librte-net-bond25 arm64 24.11.3-1~deb13u1 [59.3 kB]
Get:27 http://deb.debian.org/debian trixie/main arm64 libxdp1 arm64 1.5.4-1 [55.4 kB]
Get:28 http://deb.debian.org/debian trixie/main arm64 suricata arm64 1:7.0.10-1+deb13u1 [2,614 kB]
Get:29 http://deb.debian.org/debian trixie/main arm64 suricata-update arm64 1.3.4-1 [64.4 kB]
Fetched 3,679 kB in 26s (144 kB/s)
```

Figure 3.5.1 Suricata Installation



```
predator@raspberrypi:~ $ sudo suricata-update
sudo systemctl restart suricata
21/11/2025 -- 10:25:34 - <Info> -- Using data-directory /var/lib/suricata.
21/11/2025 -- 10:25:34 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
21/11/2025 -- 10:25:34 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
21/11/2025 -- 10:25:34 - <Info> -- Found Suricata version 7.0.10 at /usr/bin/suricata.
21/11/2025 -- 10:25:34 - <Info> -- Loading /etc/suricata/suricata.yaml
21/11/2025 -- 10:25:34 - <Info> -- Disabling rules for protocol pgsql
21/11/2025 -- 10:25:34 - <Info> -- Disabling rules for protocol modbus
21/11/2025 -- 10:25:34 - <Info> -- Disabling rules for protocol dnp3
21/11/2025 -- 10:25:34 - <Info> -- Disabling rules for protocol enip
21/11/2025 -- 10:25:34 - <Info> -- No sources configured, will use Emerging Threats Open
21/11/2025 -- 10:25:34 - <Info> -- Fetching https://rules.emergingthreats.net/open/suricata-7.0.10/emerging.rules.tar.gz.
 100% - 5174056/5174056
21/11/2025 -- 10:26:33 - <Info> -- Done.
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/app-layer-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/decoder-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/dhcp-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/dnp3-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/dns-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/files.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/http2-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/http-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/ipsec-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/kerberos-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/modbus-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/mqtt-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/nfs-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/ntp-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/quic-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/rfb-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/smb-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/smtp-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/ssh-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/stream-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Loading distribution rule file /etc/suricata/rules/tls-events.rules
21/11/2025 -- 10:26:34 - <Info> -- Ignoring file f625293e2432dbf07497d06349de6f0b/rules/emerging-deleted.rules
21/11/2025 -- 10:26:36 - <Info> -- Loaded 62288 rules.
21/11/2025 -- 10:26:36 - <Info> -- Disabled 13 rules.
21/11/2025 -- 10:26:36 - <Info> -- Enabled 0 rules.
21/11/2025 -- 10:26:36 - <Info> -- Modified 0 rules.
21/11/2025 -- 10:26:36 - <Info> -- Dropped 0 rules.
21/11/2025 -- 10:26:36 - <Info> -- Enabled 136 rules for flowbit dependencies.
21/11/2025 -- 10:26:36 - <Info> -- Backing up current rules.
21/11/2025 -- 10:26:36 - <Info> -- Writing rules to /var/lib/suricata/rules/suricata.rules: total: 62288; enabled: 46475; added: 62288; removed 0; modified: 0
21/11/2025 -- 10:26:36 - <Info> -- Writing /var/lib/suricata/rules/classification.config
21/11/2025 -- 10:26:37 - <Info> -- Testing with suricata -T.
21/11/2025 -- 10:26:45 - <Info> -- Done.
```

Figure 3.5.2 Suricata Rules added

Suricata was configured in its main configuration file (`suricata.yaml`) to use the af-packet capture method on the primary network interface, eth0, allowing it to passively monitor the network traffic.

In addition to the default Emerging Threats (ET) rules, a custom Suricata rule was created and enabled for basic testing and validation:

Code snippet

alert icmp any any -> any any (msg:"ICMP Test Detected"; sid:1000001; rev:1;)

```
predator@raspberrypi:~ $ nl -ba /etc/suricata/rules/local.rules || true
     1	alert icmp any any -> any any (msg:"ICMP Test Detected"; sid:1000001; rev:1;)
```

Figure 3.5.3 Rule customization

This simple rule ensures that any basic ICMP ping sweep from an external source would generate an alert, confirming the IDS functionality. Rules for port scan detection were primarily handled using default ET rules.

3.5 Alert Processing and Physical Notification

A custom Python script was developed using the RPi.GPIO library to process Suricata alert logs in real time. The script continuously parses the EVE JSON file and filters events based on alert severity and type.

When a high-severity intrusion event is detected, the script activates GPIO-connected hardware components. The buzzer generates an audible alert, while LEDs provide visual indication of the system state. This design converts digital intrusion events into immediate physical alerts, making the system effective even in unattended or offline environments.

Python Alert Monitoring Script :

```
#!/usr/bin/env python3

import json

import time

import RPi.GPIO as GPIO

from gpiozero import LED

# ---------------- GPIO SETUP ----------------

GPIO.setmode(GPIO.BCM)

# LEDs

GREEN = LED(22)      # Normal state indicator

RED = LED(17)        # Attack indicator

# Passive buzzer (using PWM for loud sound)

BUZZ_PIN = 5

GPIO.setup(BUZZ_PIN, GPIO.OUT)

pwm = GPIO.PWM(BUZZ_PIN, 1000)   # Base frequency

pwm.start(0)

# Suricata alert log file

EVE_LOG = "/var/log/suricata/eve.json"

# ---------------- BUZZER FUNCTION ----------------

def loud_beep(freq=1500, duration=0.25):
```

```
    pwm.ChangeFrequency(freq)
    pwm.ChangeDutyCycle(50)
    time.sleep(duration)
    pwm.ChangeDutyCycle(0)
    time.sleep(0.1)
# ---------------- NORMAL STATE ----------------
def normal_state():
    RED.off()
    pwm.ChangeDutyCycle(0)
    GREEN.on()
# ---------------- ALERT STATE ----------------
def alert_state():
    GREEN.off()
    for _ in range(5):
        RED.on()
        loud_beep(1500, 0.20)
        RED.off()
        time.sleep(0.2)
    GREEN.on()
# ---------------- MONITOR SURICATA ----------------
def monitor():
    print("Monitoring Suricata alerts... (LED + buzzer active)")
    GREEN.on()   # Normal state on boot
    with open(EVE_LOG, "r") as f:
        f.seek(0, 2)  # Move to end of file
```

```
    while True:
        line = f.readline()
        if not line:
            time.sleep(0.05)
            continue
        try:
            event = json.loads(line)
        except:
            continue
        if event.get("event_type") == "alert":
            sig = event["alert"]["signature"]
            print(f"ALERT: {sig}")
            alert_state()
# ---------------- MAIN ----------------
try:
    monitor()
except KeyboardInterrupt:
    print("Exiting...")
finally:
    pwm.stop()
    GPIO.cleanup()
```

Working Explanation

The script initializes the Raspberry Pi GPIO pins using BCM numbering and configures LEDs and a passive buzzer as output devices. A green LED indicates the normal operating state of the system, while a red LED and buzzer are used to signal intrusion events.

Suricata intrusion alerts are generated in the EVE JSON log file located at `/var/log/suricata/eve.json`. The script continuously monitors this file in real time by reading newly appended log entries. Each log entry is parsed as a JSON object, and events with the type `"alert"` are identified as intrusion events.

Upon detection of an alert, the script extracts the alert signature and activates the alert state. In this state, the green LED is turned off, the red LED blinks, and the buzzer emits a loud siren-like sound using PWM modulation. This converts digital intrusion alerts into immediate physical notifications.

The script also includes exception handling to ensure stable operation and performs proper GPIO cleanup when terminated. This design allows Sentinel Pi to function effectively even in offline or unattended environments.

3.6 Cloud-Based SIEM Integration

To achieve centralized monitoring and analysis, Sentinel Pi integrates with a cloud-based SIEM platform using the Wazuh Agent.

Wazuh Agent Installation: The Wazuh Agent was installed on the Raspberry Pi and registered with the Wazuh Cloud manager.

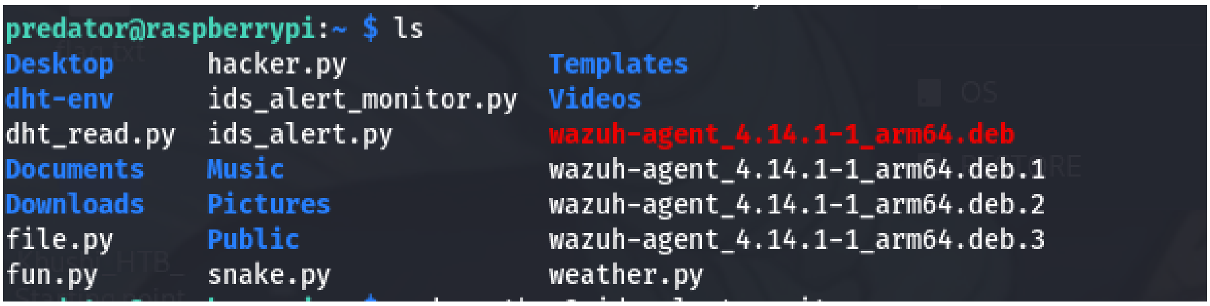


Figure 3.6 Wazuh agent installed

Log Forwarding: The agent configuration file (`ossec.conf`) was modified to instruct the agent to monitor and forward the Suricata EVE JSON file:

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

This integration enables real-time visualization of intrusion alerts through dashboards and event logs, similar to those used in professional Security Operations Centers (SOCs). Cloud-based logging also ensures that security data is preserved for later analysis and auditing.

4. TESTING AND RESULTS

4.1 Test Environment

The testing of the Sentinel Pi system was conducted using a multi-session SSH-based setup. The Raspberry Pi was accessed remotely from a laptop using Secure Shell (SSH). Multiple SSH sessions were used simultaneously to run different components of the system.

In one SSH session, the Suricata Intrusion Detection System was started to monitor live network traffic on the Raspberry Pi. In a second SSH session, the custom Python-based alert monitoring script was executed to parse Suricata alerts and control the GPIO-connected hardware components. Network attacks were initiated directly from the laptop terminal targeting the Raspberry Pi.

This setup allowed real-time observation of intrusion detection, alert generation, physical notifications, and centralized logging.

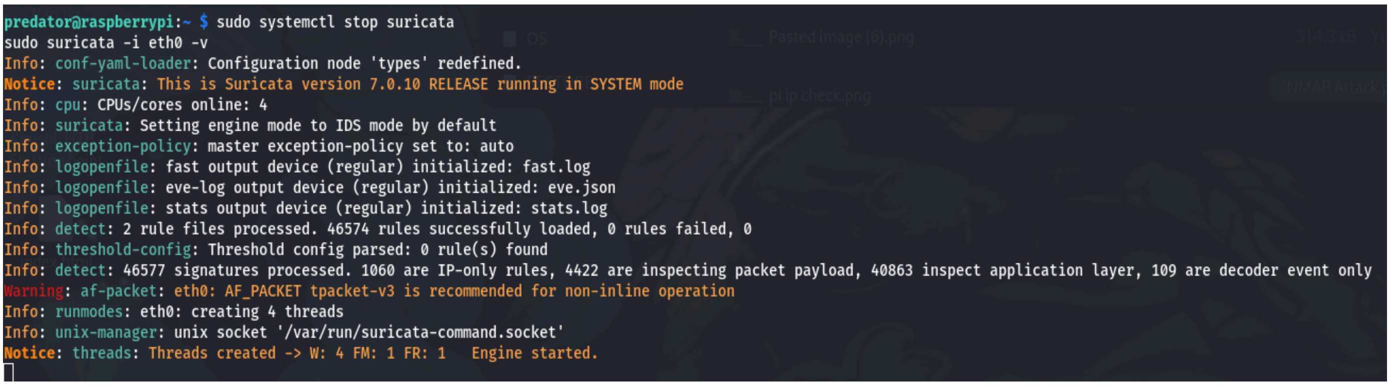


Figure 4.1 Suricata Engine started

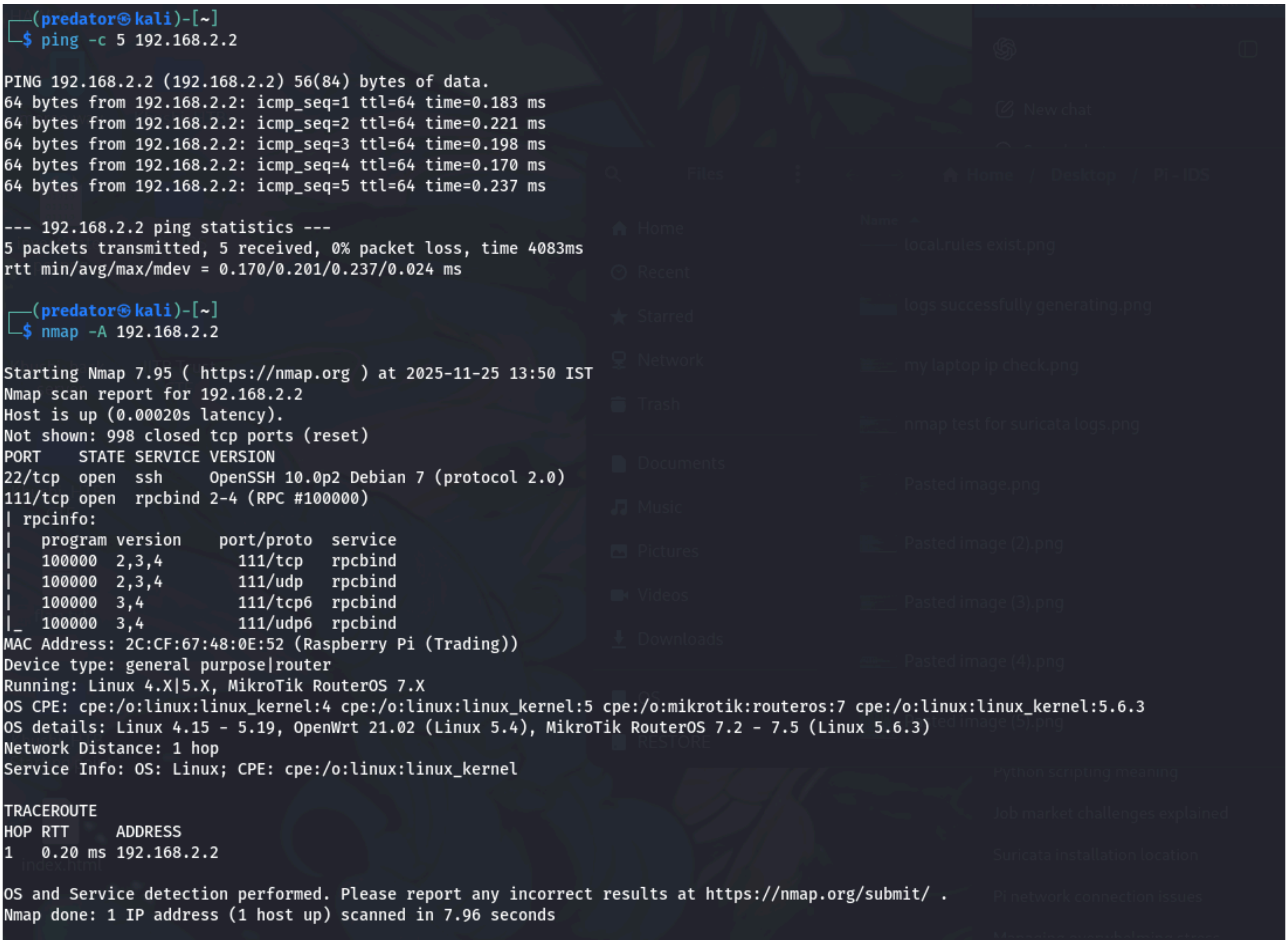
4.2 Attack Scenarios Executed

To validate the functionality of Sentinel Pi, common network reconnaissance activities were executed from the laptop terminal against the Raspberry Pi. These attack techniques simulate typical actions performed during the reconnaissance phase of a cyber attack.

The following attack scenarios were tested:

- ICMP ping requests to verify basic network probing
- ICMP ping sweeps for host discovery
- TCP SYN port scanning using Nmap
- UDP port scanning using Nmap

These attacks were executed while Suricata and the alert monitoring script were running simultaneously on the Raspberry Pi.



```
┌──(predator㉿kali)-[~]
└─$ ping -c 5 192.168.2.2

PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=64 time=0.183 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=64 time=0.221 ms
64 bytes from 192.168.2.2: icmp_seq=3 ttl=64 time=0.198 ms
64 bytes from 192.168.2.2: icmp_seq=4 ttl=64 time=0.170 ms
64 bytes from 192.168.2.2: icmp_seq=5 ttl=64 time=0.237 ms

--- 192.168.2.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4083ms
rtt min/avg/max/mdev = 0.170/0.201/0.237/0.024 ms

┌──(predator㉿kali)-[~]
└─$ nmap -A 192.168.2.2

Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-25 13:50 IST
Nmap scan report for 192.168.2.2
Host is up (0.00020s latency).
Not shown: 998 closed tcp ports (reset)
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 10.0p2 Debian 7 (protocol 2.0)
111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|_  100000  3,4          111/udp6  rpcbind
MAC Address: 2C:CF:67:48:0E:52 (Raspberry Pi (Trading))
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.20 ms 192.168.2.2

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.96 seconds
```

Figure 4.2 Attack commands executed from laptop terminal

4.3 Intrusion Detection and Alert Generation

While the attack scenarios were executed, Suricata actively inspected incoming network traffic in real time. Upon detecting suspicious activities, Suricata generated alerts based on enabled rule sets and recorded them in the EVE JSON log file.

Each alert entry contained detailed information including the event type, source IP address, destination IP address, protocol, and alert signature. Real-time monitoring of the `/var/log/suricata/eve.json` file confirmed that alerts were generated immediately after attack execution.

Figure 4.3 Suricata detecting intrusion events in real time

4.4 Physical Alert Verification

The Python-based alert monitoring script was executed in a separate SSH session on the Raspberry Pi. The script continuously monitored the EVE JSON log file and filtered events classified as intrusion alerts.

When a high-severity alert was detected, the script triggered the GPIO-connected hardware components. During these events, the green LED indicating normal operation was turned off, the red LED blinked repeatedly, and the buzzer emitted a loud alarm sound. This confirmed successful conversion of digital intrusion alerts into immediate physical notifications.

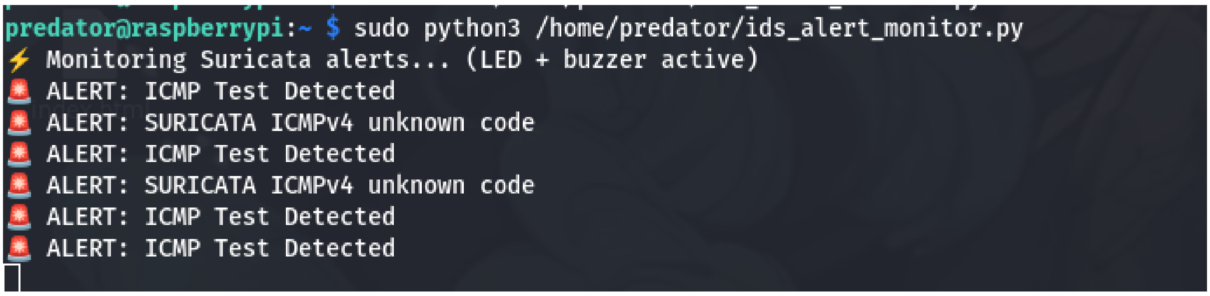


Figure 4.4 Physical alert activation

4.5 SIEM Dashboard Analysis

Service Restart: The agent was restarted: **`sudo systemctl restart wazuh-agent`**.

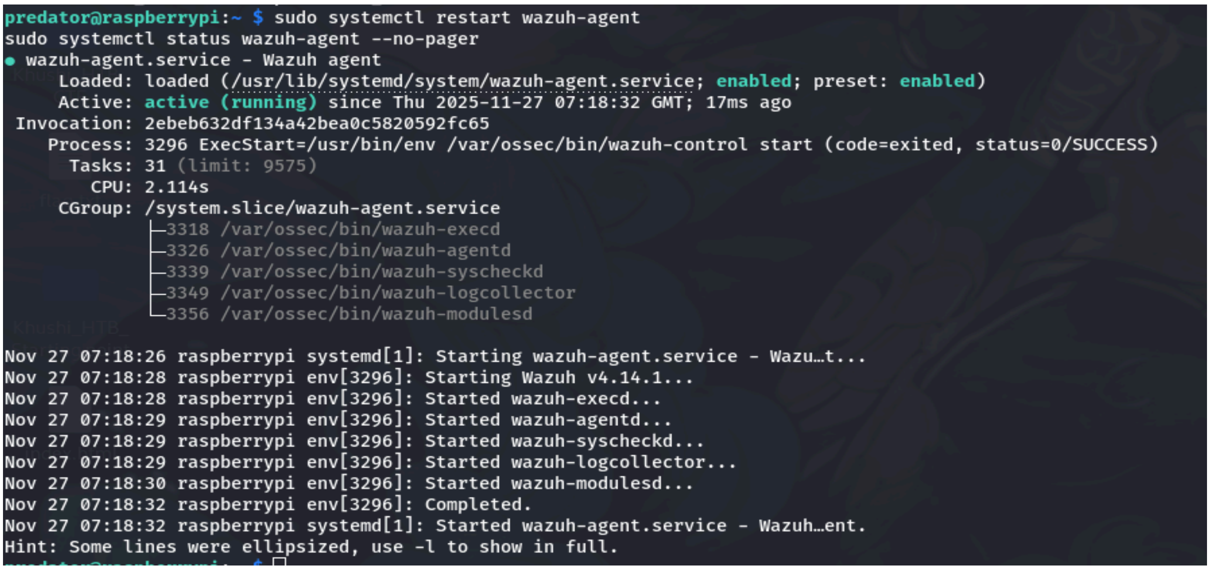


Figure 4.5.1 Wazuh agent started

Suricata alerts were forwarded to the Wazuh Cloud SIEM platform using the Wazuh Agent running on the Raspberry Pi. Detected intrusion events appeared on the SIEM dashboard in real time.

The dashboard displayed alert signatures, timestamps, and severity levels, enabling centralized visualization and analysis of detected network attacks. This demonstrated successful integration of Sentinel Pi with a professional-grade SIEM platform and simulated real-world Security Operations Center (SOC) monitoring.

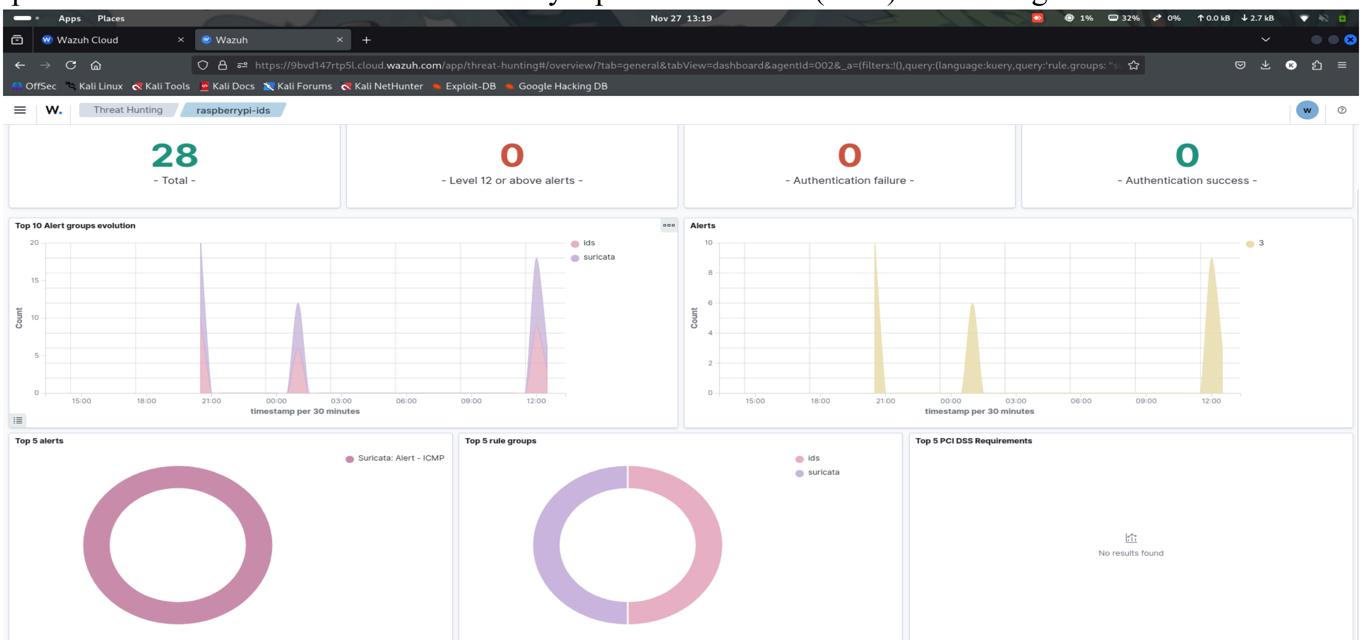


Figure 4.5.2 Wazuh SIEM dashboard displaying detected alerts

4.6 Observations and Results

The testing process confirmed that Sentinel Pi reliably detected network reconnaissance activities such as ICMP probing and port scanning. Alerts were generated in real time, physical notifications were triggered consistently, and intrusion events were successfully forwarded to the SIEM platform.

Running Suricata and the alert monitoring script in parallel SSH sessions allowed effective real-time monitoring and validation. The system demonstrated stable performance on the Raspberry Pi and successfully integrated intrusion detection, physical alerting, and centralized logging.

Attack Type	Detection	Buzzer	Wazuh Log
ICMP Ping	Yes	No (filtered)	Yes
Nmap SYN Scan	Yes	Yes	Yes
Port Scan	Yes	Yes	Yes
OS Fingerprinting	Yes	Yes	Yes

5. CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

In this project, *Sentinel Pi* was successfully designed and implemented as a low-cost Network Intrusion Detection System using a Raspberry Pi. The system effectively monitored live network traffic, detected common network reconnaissance activities, and generated alerts in real time.

Suricata IDS was deployed to inspect network packets and identify suspicious activities such as ICMP probing and port scanning. Detected alerts were logged in structured JSON format and processed by a custom Python-based alert monitoring script. The script enabled immediate physical notifications through GPIO-connected LEDs and a buzzer, converting digital intrusion alerts into real-world signals.

Additionally, the integration of Sentinel Pi with a cloud-based SIEM platform enabled centralized monitoring and visualization of intrusion events. This setup closely simulated real-world Security Operations Center (SOC) workflows while operating on resource-constrained hardware.

The successful detection of attacks, reliable physical alert generation, and real-time SIEM integration demonstrate that Sentinel Pi is a practical and effective intrusion detection solution for small-scale networks, learning environments, and demonstration purposes.

5.2 Limitations of the System

Sentinel Pi operates as a passive intrusion detection system and does not actively block malicious traffic. Additionally, the intrusion detection mechanism primarily relies on signature-based rules and therefore may not detect zero-day or previously unknown attacks unless rule sets are updated regularly.

5.3 Future Scope and Applications

While Sentinel Pi meets its current objectives, several enhancements and deployment scenarios can be considered to extend its functionality and practical applicability.

Sentinel Pi can be deployed in public and semi-public environments such as railway stations, bus terminals, educational institutions, and small offices where dedicated cybersecurity teams or Security Operations Centers may not be available. In such environments, personnel may not have the technical expertise to monitor SIEM dashboards or analyze security logs. The inclusion of physical alert mechanisms such as LEDs and buzzers enables immediate awareness of suspicious network activity, allowing non-technical staff to recognize potential threats and escalate incidents to technical teams when required.

Future enhancements may include:

- Implementation of intrusion prevention features such as automatic blocking of malicious IP addresses using firewall rules.
- Integration of machine learning–based anomaly detection to identify unknown or zero-day attack patterns.
- Enhancement of alert prioritization and severity classification for improved response handling.
- Deployment of a web-based dashboard for local visualization and system control.
- Integration with public announcement systems or mobile notifications for large public setups.
- Optimization of system performance to handle higher traffic volumes.

These enhancements would further strengthen Sentinel Pi as a scalable and deployable security monitoring solution for both technical and non-technical environments.

LITERATURE REFERENCES

[1] Open Information Security Foundation (OISF), *Suricata: Open Source IDS/IPS/NSM Engine*, 2024. [Online]. Available: https://suricata.io

[2] Proofpoint, *Emerging Threats Open Ruleset Documentation*. [Online]. Available: https://rules.emergingthreats.net

[3] Wazuh Inc., *Wazuh Documentation – SIEM Platform*, 2024. [Online]. Available: https://documentation.wazuh.com

[4] Raspberry Pi Foundation, *Raspberry Pi GPIO Pinout and Hardware Documentation*, 2023. [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[5] R. Bejtlich, *The Practice of Network Security Monitoring*. San Francisco, CA, USA: No Starch Press, 2013.