**Real-Time OS Security Event Logger**

**1. Project Overview**

The **Real-Time OS Security Event Logger (RTOS Logger)** is a Python-based monitoring tool designed to log OS-level security events. It provides insights into system vulnerabilities by recording **failed login attempts, active processes, and network connections** in real-time. The logger is designed for **continuous security monitoring** and can be deployed as a **background service** on Kali Linux.

**2. Module-Wise Breakdown**

| **Module Name** | **Functionality** |
| --- | --- |
| **Login Monitor** | Parses /var/log/auth.log to detect failed login attempts. |
| **Process Monitor** | Tracks running processes with their **PID, name, and user**. |
| **Network Monitor** | Logs **active network connections** (PID, status, IP). |
| **Logging Module** | Writes all security events to /var/log/rtos\_security.log. |
| **Daemon Service** | Runs the script as a **system service** using systemd. |

**3. Functionalities**

**Monitors failed login attempts** (prevents brute force attacks).  
 **Tracks running processes** (detects unauthorized software execution).  
 **Logs network activity** (identifies suspicious connections).  
 **Stores logs persistently** at /var/log/rtos\_security.log.  
 **Runs as a systemd service** (automatically starts on boot).

**4. Technology Used**

**• Programming Languages:** Python 3

**• Libraries and Tools:**

**psutil** – Process and network monitoring  
 **logging** – Log management  
 **systemd** – Background service setup

**• Other Tools:**

**GitHub** – Version control  
 **Nano/Vim** – Code editing  
 **Bash** – Terminal scripting

**5. Flow Diagram**

A diagram of a login attack

AI-generated content may be incorrect.

**6. Revision Tracking on GitHub**

* **Repository Name:** RTOS-event-logger
* **GitHub Link:** <https://github.com/rai-1819/RTOS-event-logger>

**7. Conclusion and Future Scope**

**Conclusion:**

The RTOS Logger successfully records key security events in **real time**, helping system administrators identify potential security threats. Its **modular structure** makes it flexible for integration with advanced **Intrusion Detection Systems (IDS).**

**Future Scope:**

**Integrate email alerts** for security breaches  
 **Enhance with AI-driven anomaly detection**  
 **Add GUI-based real-time log monitoring**

**8. References**

* **Official Python Documentation** – <https://docs.python.org/3/>
* **Psutil Library Docs** – <https://psutil.readthedocs.io/en/latest/>
* **Kali Linux Security Logs** – <https://www.kali.org/docs/>

**Appendix**

**A. AI-Generated Project Breakdown Report**

***Project Scope***

*A lightweight* ***Real-Time OS Security Event Logger*** *that:*

* *Monitors* ***system-level security events*** *(e.g., failed logins, process executions, network connections).*
* *Logs events in a structured format for analysis.*
* *Runs as a* ***background daemon/service****.*

***Tech Stack & Approach***

* ***Language****: Python (for speed & simplicity)*
* ***Modules****: psutil, os, logging, subprocess, datetime*
* ***Core Functions****:*
  + ***Monitor failed login attempts*** *(Linux: /var/log/auth.log, Windows: Event Viewer)*
  + ***Track running processes*** *(psutil)*
  + ***Detect network connections*** *(psutil.net\_connections())*
  + ***Log all events in a structured format*** *(timestamped .log file)*

***Steps to Follow***

1. ***Set Up a Basic Logger****: Create a file-based logging system.*
2. ***Monitor Security Events****: Implement real-time monitoring using Python.*
3. ***Run as a Background Daemon****: Ensure it runs continuously.*
4. ***(Optional) Display Insights****: Show simple analytics on logs.*

**B. Problem Statement**

**Design a logger that monitors and records OS-level security events, providing insights into potential vulnerabilities.**

**C. Solution / Code**

python

CopyEdit

import psutil

import os

import time

import logging

from datetime import datetime

# Log File Setup

LOG\_FILE = "/var/log/rtos\_security.log"

logging.basicConfig(filename=LOG\_FILE, level=logging.INFO, format="%(asctime)s - %(message)s")

# Function to Monitor Failed Login Attempts (Linux Only)

def monitor\_failed\_logins():

auth\_log = "/var/log/auth.log"

if os.path.exists(auth\_log):

with open(auth\_log, "r") as f:

lines = f.readlines()

for line in lines[-10:]: # Read last 10 lines for new attempts

if "Failed password" in line:

logging.info("[FAILED LOGIN] " + line.strip())

# Function to Monitor Running Processes

def monitor\_processes():

for proc in psutil.process\_iter(attrs=['pid', 'name', 'username']):

logging.info(f"[PROCESS] PID: {proc.info['pid']}, Name: {proc.info['name']}, User: {proc.info['username']}")

# Function to Monitor Network Connections

def monitor\_network():

for conn in psutil.net\_connections(kind='inet'):

logging.info(f"[NETWORK] PID: {conn.pid}, Status: {conn.status}, IP: {conn.laddr}")

# Main Function to Continuously Monitor

def monitor\_system():

logging.info("=== RTOS Security Logger Started ===")

while True:

monitor\_failed\_logins()

monitor\_processes()

monitor\_network()

time.sleep(10) # Log every 10 seconds

if \_\_name\_\_ == "\_\_main\_\_":

monitor\_system()

**Student Name:** Rai Singh

**Registration no:** 12303277

**Section:** K23HP  
**OS Used:** Kali Linux  
**Language:** Python