ACADEMIC TASK - 2 CSE 316

(OPERATING SYSTEMS) COMPUTER

SCIENCE AND ENGINEERING

Section – K23HP

Submitted by: Rai Singh

Reg No.- 12303277

Roll – 07

Contributors- Suraj

Reg No. 12323150

Roll No. 49

Submitted to: AMANDEEP KAUR

LOVELY PROFESSIONAL UNIVERSITY

LOVELY PROFESSIONAL UNIVERSITY

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

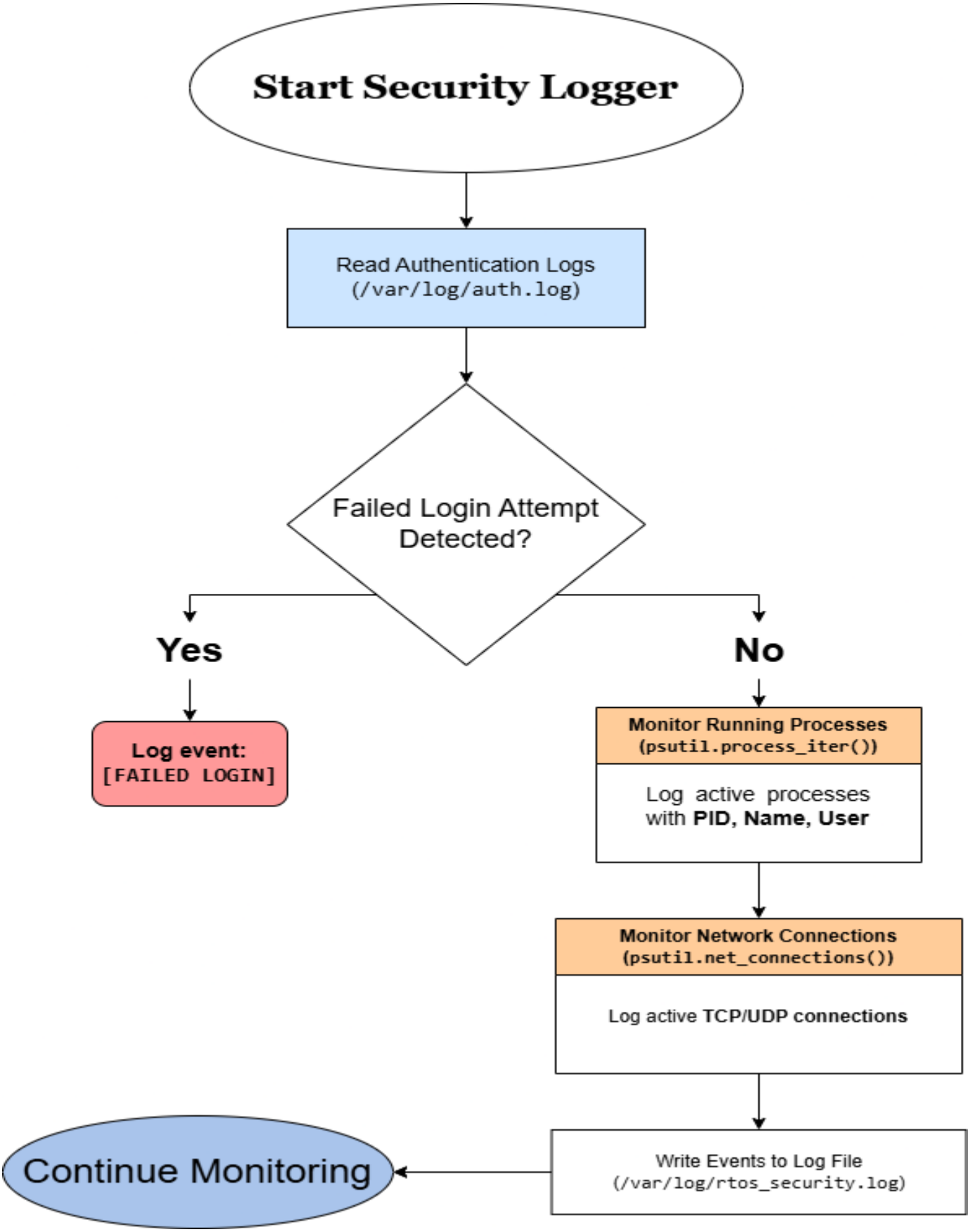
• **Libraries and Tools:**

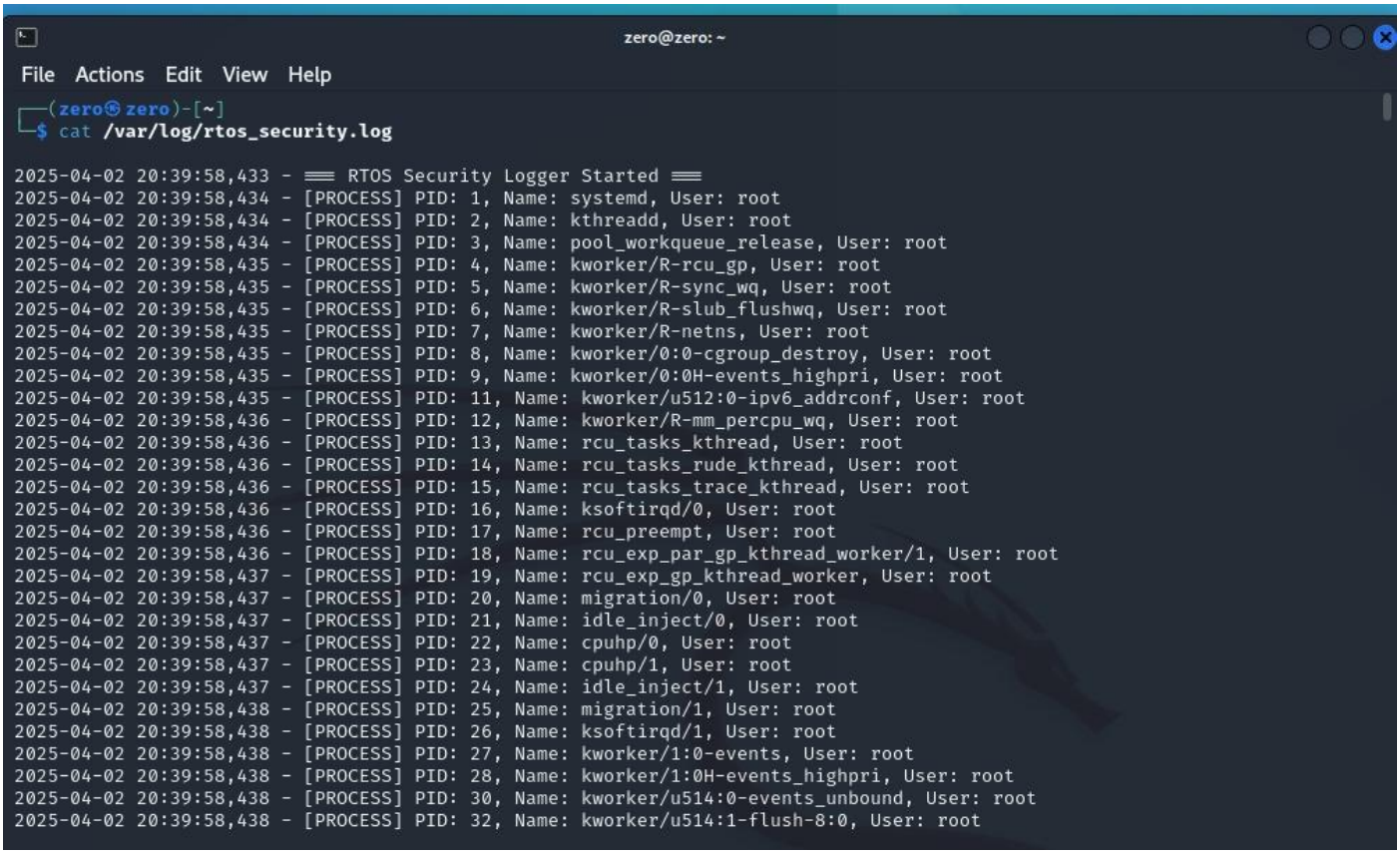
psutil – Process and network monitoring
logging – Log management
systemd – Background service setup

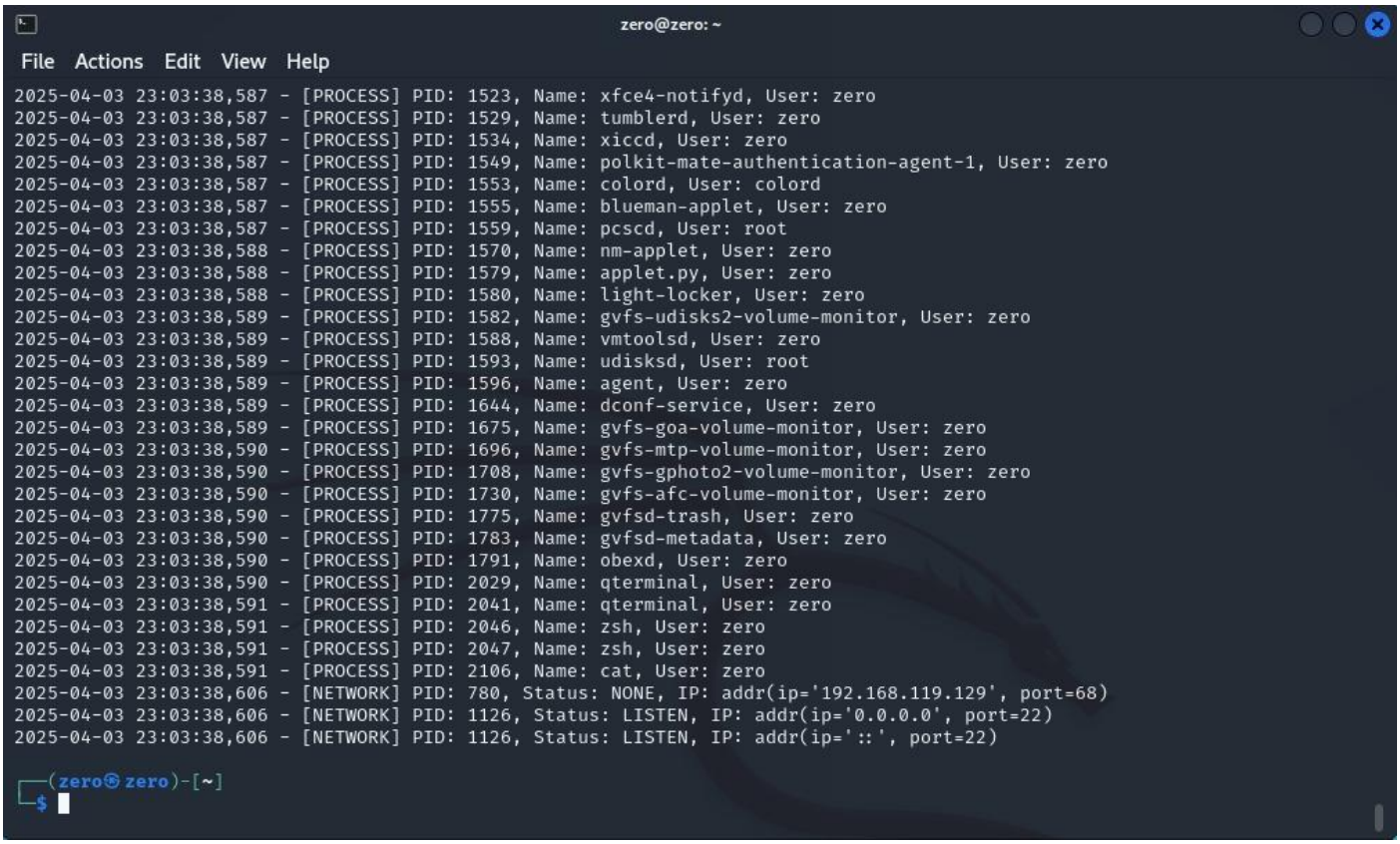
• **Other Tools:**

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

5. Flow Diagram and screenshots




zero@zero: ~
File Actions Edit View Help
(zero㉿zero)-[~]
$ cat /var/log/rtos_security.log
2025-04-02 20:39:58,433 - ═══ RTOS Security Logger Started ═══
2025-04-02 20:39:58,434 - [PROCESS] PID: 1, Name: systemd, User: root
2025-04-02 20:39:58,434 - [PROCESS] PID: 2, Name: kthreadd, User: root
2025-04-02 20:39:58,434 - [PROCESS] PID: 3, Name: pool_workqueue_release, User: root
2025-04-02 20:39:58,435 - [PROCESS] PID: 4, Name: kworker/R-rcu_gp, User: root
2025-04-02 20:39:58,435 - [PROCESS] PID: 5, Name: kworker/R-sync_wq, User: root
2025-04-02 20:39:58,435 - [PROCESS] PID: 6, Name: kworker/R-slub_flushwq, User: root
2025-04-02 20:39:58,435 - [PROCESS] PID: 7, Name: kworker/R-netns, User: root
2025-04-02 20:39:58,435 - [PROCESS] PID: 8, Name: kworker/0:0-cgroup_destroy, User: root
2025-04-02 20:39:58,435 - [PROCESS] PID: 9, Name: kworker/0:0H-events_highpri, User: root
2025-04-02 20:39:58,435 - [PROCESS] PID: 11, Name: kworker/u512:0-ipv6_addrconf, User: root
2025-04-02 20:39:58,436 - [PROCESS] PID: 12, Name: kworker/R-mm_percpu_wq, User: root
2025-04-02 20:39:58,436 - [PROCESS] PID: 13, Name: rcu_tasks_kthread, User: root
2025-04-02 20:39:58,436 - [PROCESS] PID: 14, Name: rcu_tasks_rude_kthread, User: root
2025-04-02 20:39:58,436 - [PROCESS] PID: 15, Name: rcu_tasks_trace_kthread, User: root
2025-04-02 20:39:58,436 - [PROCESS] PID: 16, Name: ksoftirqd/0, User: root
2025-04-02 20:39:58,436 - [PROCESS] PID: 17, Name: rcu_preempt, User: root
2025-04-02 20:39:58,436 - [PROCESS] PID: 18, Name: rcu_exp_par_gp_kthread_worker/1, User: root
2025-04-02 20:39:58,437 - [PROCESS] PID: 19, Name: rcu_exp_gp_kthread_worker, User: root
2025-04-02 20:39:58,437 - [PROCESS] PID: 20, Name: migration/0, User: root
2025-04-02 20:39:58,437 - [PROCESS] PID: 21, Name: idle_inject/0, User: root
2025-04-02 20:39:58,437 - [PROCESS] PID: 22, Name: cpuhp/0, User: root
2025-04-02 20:39:58,437 - [PROCESS] PID: 23, Name: cpuhp/1, User: root
2025-04-02 20:39:58,437 - [PROCESS] PID: 24, Name: idle_inject/1, User: root
2025-04-02 20:39:58,438 - [PROCESS] PID: 25, Name: migration/1, User: root
2025-04-02 20:39:58,438 - [PROCESS] PID: 26, Name: ksoftirqd/1, User: root
2025-04-02 20:39:58,438 - [PROCESS] PID: 27, Name: kworker/1:0-events, User: root
2025-04-02 20:39:58,438 - [PROCESS] PID: 28, Name: kworker/1:0H-events_highpri, User: root
2025-04-02 20:39:58,438 - [PROCESS] PID: 30, Name: kworker/u514:0-events_unbound, User: root
2025-04-02 20:39:58,438 - [PROCESS] PID: 32, Name: kworker/u514:1-flush-8:0, User: root


zero@zero: ~
File Actions Edit View Help
2025-04-03 23:03:38,587 - [PROCESS] PID: 1523, Name: xfce4-notifyd, User: zero
2025-04-03 23:03:38,587 - [PROCESS] PID: 1529, Name: tumblerd, User: zero
2025-04-03 23:03:38,587 - [PROCESS] PID: 1534, Name: xiccd, User: zero
2025-04-03 23:03:38,587 - [PROCESS] PID: 1549, Name: polkit-mate-authentication-agent-1, User: zero
2025-04-03 23:03:38,587 - [PROCESS] PID: 1553, Name: colord, User: colord
2025-04-03 23:03:38,587 - [PROCESS] PID: 1555, Name: blueman-applet, User: zero
2025-04-03 23:03:38,587 - [PROCESS] PID: 1559, Name: pcscd, User: root
2025-04-03 23:03:38,588 - [PROCESS] PID: 1570, Name: nm-applet, User: zero
2025-04-03 23:03:38,588 - [PROCESS] PID: 1579, Name: applet.py, User: zero
2025-04-03 23:03:38,588 - [PROCESS] PID: 1580, Name: light-locker, User: zero
2025-04-03 23:03:38,589 - [PROCESS] PID: 1582, Name: gvfs-udisks2-volume-monitor, User: zero
2025-04-03 23:03:38,589 - [PROCESS] PID: 1588, Name: vmtoolsd, User: zero
2025-04-03 23:03:38,589 - [PROCESS] PID: 1593, Name: udisksd, User: root
2025-04-03 23:03:38,589 - [PROCESS] PID: 1596, Name: agent, User: zero
2025-04-03 23:03:38,589 - [PROCESS] PID: 1644, Name: dconf-service, User: zero
2025-04-03 23:03:38,589 - [PROCESS] PID: 1675, Name: gvfs-goa-volume-monitor, User: zero
2025-04-03 23:03:38,590 - [PROCESS] PID: 1696, Name: gvfs-mtp-volume-monitor, User: zero
2025-04-03 23:03:38,590 - [PROCESS] PID: 1708, Name: gvfs-gphoto2-volume-monitor, User: zero
2025-04-03 23:03:38,590 - [PROCESS] PID: 1730, Name: gvfs-afc-volume-monitor, User: zero
2025-04-03 23:03:38,590 - [PROCESS] PID: 1775, Name: gvfsd-trash, User: zero
2025-04-03 23:03:38,590 - [PROCESS] PID: 1783, Name: gvfsd-metadata, User: zero
2025-04-03 23:03:38,590 - [PROCESS] PID: 1791, Name: obexd, User: zero
2025-04-03 23:03:38,590 - [PROCESS] PID: 2029, Name: qterminal, User: zero
2025-04-03 23:03:38,591 - [PROCESS] PID: 2041, Name: qterminal, User: zero
2025-04-03 23:03:38,591 - [PROCESS] PID: 2046, Name: zsh, User: zero
2025-04-03 23:03:38,591 - [PROCESS] PID: 2047, Name: zsh, User: zero
2025-04-03 23:03:38,591 - [PROCESS] PID: 2106, Name: cat, User: zero
2025-04-03 23:03:38,606 - [NETWORK] PID: 780, Status: NONE, IP: addr(ip='192.168.119.129', port=68)
2025-04-03 23:03:38,606 - [NETWORK] PID: 1126, Status: LISTEN, IP: addr(ip='0.0.0.0', port=22)
2025-04-03 23:03:38,606 - [NETWORK] PID: 1126, Status: LISTEN, IP: addr(ip='::', port=22)
(zero㉿zero)-[~]
$

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()
```

OS Used: Kali Linux
Language: Python