Local Threat Analyzer

Name: Rohan Kailas Langhe

Project Title: Local System Threat Analyzer (Python-Based)

Objective

To design and develop a lightweight local threat detection tool that analyzes running system processes for suspicious behavior, including high resource usage and potential malware presence. The project simulates a basic endpoint monitoring system for detection, alerting, and logging of potential threats in real-time.

Lab Setup

- Platform: Windows 10 / 11
- Programming Language: Python 3.11
- Execution Environment: Terminal-based (CLI)
- Dependencies:
  - psutil (process monitoring)
  - tabulate (formatted output)
  - csv (log output)

Tools/Technologies Used

- Python 3
- psutil
- tabulate
- pyinstaller (for packaging as .exe)

Detection Workflow

Step 1: Process Scanning

- The tool uses psutil to list all running processes, extracting PID, process name, CPU usage, memory consumption, and file path.

Step 2: Threat Identification

Each process is evaluated against the following criteria:

- CPU usage > 50%
- Memory usage > 500MB
- Executable path is missing or suspicious
- Process name matches known malicious signatures (e.g., svch0st.exe, backdoor.exe)

Step 3: Real-Time Monitoring

- A refresh loop runs every 5 seconds (customizable), continuously scanning the system and updating the display.

Step 4: Logging

- All flagged suspicious processes are saved into a timestamped .csv log file containing process metadata and threat status.

Sample Output

```
PID   Name            CPU%    Memory (MB)   Path              Status
1234  svch0st.exe      87.3%   762          C:\Windows\...    SUSPICIOUS
4321  explorer.exe     12.4%   142          C:\Windows\...    Normal
```

Example CSV Log

```
PID,Name,CPU%,Memory (MB),Path,Status,Timestamp
```

```
1234,svch0st.exe,87.3,762,C:/Windows/System32, SUSPICIOUS,2025-04-29_13-20-45
```

Outcomes & Learnings

- Learned practical use of process monitoring tools and system resource auditing using Python.
- Built an alerting system for anomalous behavior without relying on third-party tools.
- Gained experience in structuring CLI-based security tools with logging capabilities.
- Practiced clean logging, custom thresholds, and safe code execution.

Conclusion

This project replicates a basic host-based intrusion detection mechanism focused on local process monitoring. It demonstrates how Python can be used to build efficient threat-detection tools suitable for endpoint security on budget-constrained systems or learning labs.

Next Steps

- Integrate hash-based malware signature detection using VirusTotal API.
- Add logging to a centralized SQLite or ElasticSearch instance.
- Expand detection to include suspicious network activity (e.g., outbound connections).
- Build a lightweight GUI version using Tkinter or PyQt for non-technical users.