Forensic Log Analyzer CLI Tool

Day-1

The project began with an exploration of the .vlog file format. Sample session logs were reviewed to understand their structure, pattern, and metadata components such as timestamps, event types, user actions, and target paths. A key observation was that each line followed a consistent format including a hexadecimal ID, timestamp, event type, user or IP, and a target. This day was foundation for building a parser by identifying the components required for extraction.

Day-2

On the second day, python script was implemented using regular expressions(re) to parse structured .vlog entries. The parser successfully extracted fields such as timestamp, event type, action, actor (user/IP), and target path. It also differentiated valid entries from malformed ones, saving invalid lines to a separate logs.csv for auditability. This script ensured that all clean logs could be transformed into structured data for further analysis.

Day-3

The structured log entries were divided into categories: process, file, and user events based on the XR- tag in the event type. A timeline was generated by sorting all events chronologically and saving them to timeline.csv. This timeline served as a clean, organized dataset representing system activity, which would be crucial for anomaly detection and visualization.

Day-4

A rule-based approach was implemented to identify suspicious activity. Specifically, the tool flagged scenarios where a user executed a KILL command on a shadow process (XR-SHDW) followed by a DELETE operation (XR-DEL) within a 10-second window. These sequences were considered potentially malicious and were saved in a separate file, suspected\_anomalies.csv. This enabled quick identification of harmful behavior hidden in routine logs.

Day-5

Visual representations of the log data were created using Matplotlib and Plotly. Graphs such as event\_frequency.png showed the volume of log entries over time, user\_activity.png highlighted activity per user, and anomalies\_highlighted.png visually marked suspicious sequences. Interactive HTML versions were also generated using Plotly for deeper insights, enabling dynamic exploration of user patterns and event trends.

Day-6

All scripts were integrated into a single command-line tool, cli\_tool.py, using Python’s argparse library. The CLI accepted input directory paths and optional flags like --timeline, --alerts, and --summary. This made the tool flexible and easy to use in forensic labs. The --timeline flag generated structured and invalid logs, --alerts ran anomaly detection and saved plots, while --summary printed key insights to the terminal.

CONCLUSION

After seven days, the project successfully evolved from raw log analysis to a complete forensic investigation CLI tool. It automates log parsing, identifies anomalies, and provides investigators with both visual and structured evidence — all packaged in an easy-to-use interface.

RESULTS



