Design Report - SOC Log Analysis Toolkit

Name: Anthony Tast
Date: November 4, 2025

Credit: Initial drafts generated with AI (Claude Sonnet 4.5), modified and tested by me

Program Architechture

Program has two main components that work together:

Component 1: Parser - Reads CSV files and checks if the data is valid

- Reads log files line by line
- Checks order of timestamps
- Makes sure all required fields are present
- Keeps track of errors with line numbers

Component 2: Analytics - Analyzes the data

- Counts how many of each event type occurred
- Finds most frequent events
- Detects suspicious activity (like privilege escalation)
- Searches for access to sensitive files

Design Decisions

ArrayList

I chose to store events in an **ArrayList** because:

- It is dynamic and uses less memory
- It's fast when iterating through all the events in order
- Duplicates can be removed later when needed, instead of always removing them
    - HashSet automatically removes duplicates, saving time. But I decided against it because HashSet uses more memory and sometimes duplicates need to be counted (like for frequency analysis)

Handling Invalid Data

When the parser finds invalid data, it immediately rejects it and explains why

- Prints exact line number and error
- Invalid data doesn't crash the program
- Only valid data is analyzed

Examples of validation:

- Process IDs must be positive numbers
- Privilege must be either "user" or "root"

- File permissions must be 3 digits
- IP addresses must be in the right format
- Ports must be between 1 and 65535

Removing Duplicates

All events are logged and duplicates are only removed when required by/for a specific method

- Some analytics need to count duplicates while other analytics need unique events only

Complexity

Parser Speed

- Reading and validating each line takes about the same amount of time
- If there are **n** records, it takes **O(n)** time to process them all
- This means it's linear - double the data, double the time

Analytics Speed

Simple operations (O(n) - iterate list once):

- Removing duplicates
- Counting event types
- Finding privilege escalations
- Searching for file access (r, w, etc.)

Slower operations (O(n log n) due to sorting):

- Finding top K frequent events
- Ranking processes by activity
- Detecting high-frequency processes

Memory Usage

The program stores all events in memory, making it **O(n)** space complexity

Testing

Testing Covers:

- All 8 event types (read, write, execute, fork, etc.)
- All validation rules (bad timestamps, invalid privileges, wrong formats)
- Edge cases (empty files, single events, weird inputs)
- Files from the `data/` folder

Test results:

- All 87 tests pass
- The program correctly reads 19 valid events from sample data
- It correctly rejects 6 invalid records, explains why for each, and continues processing valid events

Summary

The program successfully parses security log files, validates all data, and provides analytics to detect suspicious activity. It uses memory efficiently with ArrayList storage and has an efficient time complexity of **O(n)** to handle large datasets. The design is simple, reliable, and catches all invalid data before analysis.