School of Computing & Information Technology

CSCI262 System Security
SIM-2025-S4

Assignment 3 (14 marks, worth 14%)

Due 23 Nov 2025 by 9:00 pm Singapore time

Overview

You need to design and implement an Email system event modeler & intrusion detection system in accordance with the system descriptions below. The implementation is to be in C, C++, Java, or Python. While there are concrete details on the form of the initial input, and certain inputs along the way, the format of intermediate data is up to each group.

You need to provide a report in a file Report.pdf covering the various points through this assignment where information is required. This report should be broken into sections associated with the components as follows:

- Initial input.
- Activity engine and logs.
- Analysis engine.
- Alert engine.

Initial Input

You only need command line options at the setup phase, some user input is required later.

IDS Events.txt Stats.txt Days

Events.txt and Stats.txt define the formats and the distributions of the events to be modelled. Days is an integer used in the next section.

Here goes an example **Events.txt** file. This file describes the events and some of their parameters.

5
Logins:D:0::2:
Time online:C:0:1440:2:
Emails sent:D:0::1:
Emails opened:D:0::1:
Emails deteled:D:0::2:

The first line contains the number of events being monitored. Each subsequent line is of the form

Event name:[CD]:minimum:maximum:weight:

C and D represent continuous and discrete events respectively. Discrete events must take integer values and occur one at a time, continuous events don't need to take an integer value and an occurrence of that event may be of any value. The minimum and maximum specify the allowed range for that event type across a day. Continuous events need to be recorded in two decimal places. The weights are used in the alert engine and will always be positive integers.

The file **Stats.txt** contains the distributions to be modelled for the events. Here goes an example Stats.txt file.

5
Logins:4:1.5:
Time online:150.5:25.00:
Emails sent:10:3:
Emails opened:12:4.5:
Emails deteled:7:2.25:

The first line again contains the number of events being monitored. Each subsequent line is of the form

Event name:mean:standard deviation:

Your program should appropriately report events and statistics read in, as evidence this phase works. You should include in your report a description of:

1. How you are going to store the events and statistics internally.
2. Potential inconsistencies between Events.txt and Stats.txt. You should attempt to detect those inconsistencies. If there are inconsistencies you are aware of but haven't attempted to detect them, note this in your report.

Activity Engine and the Logs

Instructions for Event Generation and Logging with the Activity Engine

1. **Starting Event Generation**

   - After completing the initial setup and loading the base files, start the activity engine to generate and log events.
   - Ensure that your program provides feedback on its progress without producing excessive detail.

2. **Generating Consistent Statistics**

   - Generate events that are **approximately consistent** with the statistics specified in Stats.txt.
   - Log events over the number of days specified during the initial setup of the IDS.

3. **Logging Options**

   - You may choose to store the events in separate files for each day or in a single log file, according to your preference. This collection of events will serve as baseline data for the system.

4. **Reporting Requirements**
   Include the following descriptions in your report:

   - **Event Generation Process**:
     Explain how events are generated to align approximately with the specified distribution in Stats.txt. Describe any distinctions between handling discrete and continuous events.
   - **Log File Structure**:
     - Specify the name and format of the log file and justify your choice of format.
     - Ensure the log file is human-readable, as it will be used for subsequent parts of the program.

Analysis Engine

Your program should indicate when event generation is complete and when it is starting the analysis phase. At this point, you will measure the baseline data for each event and calculate the associated statistics.

1. **Data Processing and Storage**:

    - Compute daily totals for each event and save them in a data file.
    - Calculate the mean and standard deviation for each event based on this baseline data.
    - Provide progress updates as appropriate.

2. **Handling Incomplete Data Consistency**:

    - If data generation does not fully match a given distribution, the analysis engine can still read from and report on the log file.

3. **Reporting Requirements**:

    - Include in your report the name and format of the file that stores daily totals and statistical data for each event.

Alert Engine

Consistency Check and Intrusion Detection Workflow

1. **Overview of the Alert Engine**

   The alert engine monitors consistency between "live data" and baseline statistics to detect potential anomalies. This process consists of three main phases: input setup, data generation, and anomaly detection.

2. **Input Setup**

   - Prompt the user to select a file containing new statistics. This file will follow the same format as the previously used Stats.txt but may include different parameters for specific events.
   - Request the number of days for which to generate activity data.

3. **Data Generation**

   - Use the **activity engine** to simulate and produce event data for the specified number of days.
   - Run the analysis engine to calculate daily totals based on the simulated data, which will serve as input for alert detection.

4. **Anomaly Detection Process**

   - For each day, calculate an **anomaly counter** by following these steps:
     i. **Compute Standard Deviation**: For each event, determine how many standard deviations the daily event value deviates from its baseline mean, which was calculated from previous data.
     ii. **Apply Event Weights**: Multiply each deviation by its assigned weight, as specified in Events.txt.
     iii. **Sum Deviations**: Add up all weighted deviations to obtain the total anomaly counter for the day.

5. **Example of Anomaly Calculation**

   - Suppose the baseline mean for daily logins is 4, with a standard deviation of 1.5.
   - On a particular day, there is only 1 login, which is 2 standard deviations below the mean.
   - If the login event weight is 2, then this event contributes $2 \times 2 = 42$ to the anomaly counter.

6. **Threshold Comparison and Reporting**

   - **Threshold Calculation**: Set the threshold for detecting an anomaly as $2 \times (sum\ of\ weights)$, where weights are defined in Events.txt.
   - **Daily Reporting**: For each day:
     - Output the threshold value and the calculated anomaly counter.
     - State whether the day is "okay" or "flagged" as an alert, based on whether the anomaly counter meets or exceeds the threshold.

7. **Repeating the Process**

   - Once all days are processed, return to the beginning of this phase so the user can load another statistics file and specify a new number of days for analysis.
   - Provide an option to quit.

Notes on submission

1. Submission is via Moodle. Everything will need to be uploaded in a single zip file. Please don't put subdirectories in your submission. In addition to addressing the specified points, you can include anything of significance in your report. You should report on any parts not completed. be marked.

2. Include the compilation instructions with your submission (i.e., provide a readme.txt file).

3. Late submissions will be marked with a 25% deduction for each day, including days over the weekend.
4. Submissions more than three days late will not be marked unless an extension has been granted.

5. If you need an extension apply through SOLS, if possible **before** the assignment deadline.

6. Plagiarism is treated seriously. Students involved will likely receive zero.