**Name: Akilesh Jayakumar**

**UOW ID: 7901240**

**Assignment 3**

**Report on Intrusion Detection System Implementation**

**Initial Input**

In IDS assignment, I have implemented a command-line interface for the initial input. The program takes three arguments: the filenames for `Events.txt` and `Stats.txt`, and the number of days for which the simulation is to run. This approach ensures that the user can quickly start the program with all necessary inputs without additional prompts, enhancing usability.

Note: Since the sample Events.txt file provided in the assignment paper has empty “maximum” values, I have decided to place the value “0” in my Events.txt file.

A black screen with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

**Activity Simulation Engine and the Logs**

For the activity engine, I have designed a function to generate a dataset for each event type across the specified number of days. The dataset generation is based on the mean and standard deviation provided in `Stats.txt`. The program handles both discrete and continuous events, ensuring that generated data aligns with the event's nature. The generated data is then logged in a file named `ActivityLogs.txt`. This file is overwritten for each run to maintain the latest simulation data, preventing clutter from multiple log files.

Screenshot of `ActivityLogs.txt`:

A screenshot of a computer

Description automatically generated

**Analysis Engine**

The analysis engine reads the generated `ActivityLogs.txt` file to compute the daily totals of each event. It then calculates the mean and standard deviation for these totals, which are stored in `BaselineStats.txt`. This file serves as a reference point for detecting anomalies in subsequent data generated by the alert engine.

Screenshot of Analysis Engine:

A screenshot of a computer

Description automatically generated

**Alert Engine**

The alert engine is the core of the IDS, where it compares live data against baseline statistics to detect anomalies. It prompts the user for a new `Stats.txt` file and a number of days, simulating new events based on these inputs. The engine calculates an anomaly counter for each day, comparing it against a threshold derived from event weights. Days where the anomaly counter exceeds the threshold are flagged as potential intrusions. The results, including anomaly counters and flagged days, are written to `anomalyCounter.txt`.

Screenshot of Alert Engine:

A screenshot of a computer screen

Description automatically generated

Screenshot of anomalyCounter.txt`:

A screenshot of a computer

Description automatically generated

**Handling of Events and Statistics**

Internally, events and statistics are stored as lists of strings, which are parsed and processed as needed. This method allows for flexible handling of different event types and easy integration with Python's file handling and statistical functions.

**Detecting Inconsistencies**

I implemented a consistency check between `Events.txt` and `Stats.txt` to ensure that the events listed in both files match in number and names. This check is crucial to prevent errors during data generation and analysis phases.

**Process of Generating Events**

The event generation process differs between discrete and continuous events. For discrete events, integer values are generated within the specified range, while for continuous events, values are generated with up to two decimal places. This distinction ensures the data aligns with the nature of each event type.

**Log File Format and Justification**

The log file, `ActivityLogs.txt`, is structured to be human-readable with clear demarcations for each day and event. This format facilitates easy reading and debugging. Additionally, the file format aligns with the requirements for the analysis engine, ensuring seamless data processing.

**Name and Format of Statistical Data File**

The statistical data, including daily totals and calculated mean and standard deviation, are stored in `BaselineStats.txt`. This file uses a straightforward format that lists each event with its corresponding statistical values, making it easy to comprehend and utilize for anomaly detection.

While implementing the IDS, I was mindful of user interaction, data integrity, and efficient data processing. The system's modular design, coupled with input validation and clear data processing, shows my approach to implementing an IDS.