School of Computing & Information Technology

CSCI262/CSCI862 System Security
Spring 2018

Assignment 3 (24 marks, worth 12%)

Due 11:55pm Monday 22^{nd} October 2018 (Start of Week 13)

The groups

Students should form into groups of 2 or 3. Students need not be in the same lab at Wollongong, nor do they need to be in the same code (262 vs 862), although students in Wollongong cannot team up with students in Liverpool. If students are from different labs they will need to agree on a lab they plan to present at.

There will be an assignment demonstration in the week 13 lab at Wollongong, not for Liverpool since there won't be a lecture in Week 13 due to the trade show in Wollongong.

One person from each group can email the subject coordinator a list of the group members and which lab you plan to present in. `Please cc in the other student or students in your group`. After Friday 14^{th} September 2018 we will start allocating people into groups so if you want to choose your group please let me know before then. You can use the forum on Moodle to help find people to work with. If you are contacting people it makes sense to let them know which language you would want/are able to code in. If you know which language you won't to work in you can let us know and we can help people get into groups.

Each group needs to submit the code and a report. Only one student from each group needs to submit. Submission is via Moodle, other than for the demonstration in the lab. The same code submitted via Moodle should be demonstrated in the lab and the lab demonstration is not the only marking that will take place.

Overview

Each group needs to design and implement a honeypot event modeller & IDS/auditor system in accordance with the descriptions below. It's a honeypot in the sense that you are to be able to produce events consistent with provided statistics, so that if those statistics accurately represented the real world your system and the real world would be indistinguishable to a certain level of detail. 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.

The implementation is to be in C, C++ or Java. Your program must compile on Banshee with instructions you provide in a file `Readme.txt`. Sample input instructions will be specified for a program compiled to `Traffic`.

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:

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

Initial Input

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

```
Traffic Vehicles.txt Stats.txt Days
java Traffic Vehicles.txt Stats.txt Days
```

`Days` is an integer used in the next section. All activity is associated with a single road.

Here goes an example `Vehicles.txt` file. This file describes the vehicles and some of their parameters.

```
6
Bus:0:LLLDDD:3:2:
Motorbike:0:LLDDLL:1:1:
Red car:1:LDLLDL:1:1:
Elephant:1:LD:2:5:
Taxi:0:LDDL:0:2:
Emergency:0:LLDD:3:0:
```

The first line contains the number of vehicle types being monitored, those are the only vehicles that travel on the road. Each subsequent line is of the form

```
Vehicle name:Parking flag:Registration format:Volume weight:Speed weight:
```

The `Parking flag` indicates whether the vehicle type is allowed to park at the side of the road, with a 0 used to indicate they cannot and a 1 used to indicate they can. The `Registration format` uses L for letter and D for digit to indicate the format of the registration plate for the vehicle. For example, for the data above, a Bus might have a plate ABC012, a Taxi X44T, and an Elephant M1. The Vehicle name is simply that, a name, and you should not attempt to modify the behaviour based on the name.

The `Volume weight` and `Speed weight` are used in the alert engine and will always be non–negative integers. They indicate the significance of anomalous behaviour by a particular Vehicle type. For example, Emergency vehicles are allowed to speed, so the speed weight is 0; it's not a problem; while Elephants speeding should be seen as a significant problem, particular since the speed set in the example, see below, is 60 km/h and African Elephants can only reach about 40 km/h.

The file `Stats.txt` contains the distributions to be modelled for the events. This is distinct from the above to facilitate an extension to multiple roads. Here goes an example `Stats.txt` file.

```
6 5 60 20
Bus:3:1:40:10:
Motorbike:10:3:55:5:
Red car:20:4:50:4:
Elephant:2:1:10:10:
Taxi:5:2:53:7:
Emergency:1:0.5:60:10:
```

The four numbers on the first line are all positive integers. They represent, respectively:

- The number of vehicle types being monitored.
- The length of the road in kilometres.
- The speed limit in kilometres per hour. This is the same across the whole road.
- The number of parking spaces available.

Each subsequent line is of the form

`Vehicle type:Number mean:Number standard deviation:Speed mean: Speed standard deviation:`

The means and standard deviations do not need to be integers but will be specified to at most 2 decimal places.

Your program should appropriately report vehicle types and statistics read in, as evidence this phase works. If you pick up problems with the reading in these files, so they don't exist or an incorrectly formatted, you should report the problem and abort.

You should include in your report a description of:

1. How you are going to store the vehicle types and statistics internally.
2. Potential inconsistencies within and between `Vehicles.txt` and `Stats.txt`. You should attempt to detect (at least some of) 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

Once the intial setup has taken place, and you have read in the base files, the activity engine should start generating and logging events.

Time should be stepped in 1 minute blocks. Your program should give some indication as to what is happening, without being verbose. So you might indicate that you have started this phase and as it is processing indicate the currently day being generated.

There are five types of events. The first is associated with vehicle generation, the other four with existing vehicle activity.

1. Vehicle arrival, always at the start of the road. This is only up to 23:00 on any given day.
2. Vehicle departure via a side road. The vehicle is then out of the road system.

3. Vehicle departure via the end of the road. The vehicle is then out of the road system.

4. Vehicle parks or stops parking.

5. Vehicle moves and possibly changes speed.

You should think about appropriate probabilities for each of those activities. The statistics associated with vehicle volume are tied to the vehicle arrival. The statistics associated with vehicle speed are tied to the speed of the vehicle when it arrives on the road. You only test breaches of speed limit using the average speed across the whole road based on the times, so you don't need to record the speed of the vehicle at each time step.

You are attempting to produce statistics approximately consistent with the statistics specified in the statistics file. You should log for the number of `Days` specified at the initial running of `Traffic`. You can, if you like, store the activities in distinct files for each day, or in a single log file. This collection of events forms the baseline data for the system.

You can clear all activity at midnight. Every vehicle left on the road, if there are any, simply disappears ⌣̈.

You should include in your report a description of:

1. The process used to generate events approximately consistent with the particular distribution. While the vehicle arrivals are discrete events, the speed of the vehicle is effectively continuous so the way you generate something in accordance with the distribution will likely differ.

2. The name and format of the log file, with justification for the format. You will need to be able to read the log entries for subsequent parts of the program. The log file needs to be human readable.

3. Any alarms that may be raised during the activity, so an immediate detection of a problem.

Analysis Engine

Your program should indicate it has completed event generation and is going to begin analyis. You can now measure the baseline data for the traffic and determine the statistics associated with the baseline.

Produce totals for each vehicle type for each day, store that in a data file, and determine the mean and standard deviation associated with that vehicle volume and speed across that data. Report what is happening as you consider appropriate. Note again the speed statistics are based just on the initial arrival speed of the vehicle.

Your analysis engine should also test for average speed breaches for each vehicle that goes off the end of the road, not for those that depart by side streets. You calculate the average speed based of it's arrival and departure times and you don't need to take into account any parking the vehicle may do. For each day, you can produce a list of vehicles that have breached the average speed limit.

Even if you are unable to produce data consistent with a given distribution you can still have the analysis engine reading and reporting on the log file.

You should include in your report a description of:

1. Specify the file containing the daily totals for the events.

2. Possible anomalies in reading the logs.

3. Possible anomalies in determining the statistics.

Alert Engine

The alert engine is used to check consistency between "live data" and the base line statistics.

Once this phase is reached you should prompt the user for a file, containing new statistics, and a number of days. The statistics file has the same format as `Stats.txt` from earlier but will generally be different. You should run your activity engine and produce data for the number of days specified. Use the analysis engine to produce daily totals, those are used in alert detection.

For each day generated you need to report on whether there is are intrusions detected by comparing anomaly counters with thresholds. You calculate anomaly counters by adding up the weighted number of standard deviations each specific tested event value is from the mean for that event, where the standard deviation and mean are those you have generated from the base data and reported, and the weights are taken from the original `Vehicles.txt` file. There are to be distinct analyses and reports for the Vehicle volume and for the Vehicle speed, so there is an anomaly counter for each. We will go through an example of this in class.

For example, if the mean number of Elephants per day is 2 and the standard deviation is 1; then if we get 1 Elephant in a day we are 1 standard deviation from the mean. Referrring back to the weight of the Elephant vehicle type we see it was 2 so the Elephant contributes 2 to our overall anomaly counter. There would be a similar analysis for Elephant speed.

The thresholds for detecting an intrusion are 2∗(Sums of weights) where the weights are taken from `Vehicles.txt`, and they are across speeds or volumes depending on which threshold you are looking at. If an anomaly counter is greater than associated threshold you should report this as an anomaly.

You should output the thresholds at the start, since they are fixed, and for each day give the values of the two anomaly counters as well as reporting a Volume anomaly and/or a Speed anomaly or No anomaly.

Once the alert engine part has finished you should return to the start of this phase, so another set of statistics and number of days can be considered. An option to quit should be provided.

Some final notes

In addition to addressing the specified points, you can include anything of significance in your report. You should identify any parts not completed.

The data files will be generated on Banshee and used on Banshee. When you are transferring files from Moodle to Banshee for use, be careful that control characters aren't added at the end of lines and so on. The utility `dos2unix` may help sort out the format if you are unsure; and if you have no idea what this means, please ask!

Submission

Submission is by Moodle, in a single zip file without directories in it, by 11:55pm Monday 22^{nd} October 2018. Only one person per team needs to submit. The lab demonstrations will take place in the Week 13 labs and we expect every group member to attend.

Your code must compile on Banshee with the instructions you provide. If it doesn't you will likely be given zero for the coding part of this assignment. For this program in particular weI expect the main function to be a clear readable indication of the process.