ECE318: Programming Principles for Engineers

Individual project specification

Date of post: 13/10/2023

General info & important deadlines:

1) This is an ***individual piece of work***, hence you will have to design and implement it ***on your own.*** No group projects will be accepted.
2) **Submission deadline: 20/11/2023** (Monday November 20th, 2023)
   - Time: 17:00
   - Submission via Teams
   - What to submit: Code and UML diagram in a Zip file.

3) **Demo dates: 22-23/11/2023** (Wednesday and Thursday 22nd and 23rd November 2023)
   - 17:10-18:00 (Wednesday November 22nd)
   - 18:10-19:30 (Thursday November 23rd)

Project Description & Specification – Cyber Threat Incident Dashboard

You are a cybersecurity research engineer and you are asked to ***design and implement a Cyber Threat Incident Dashboard (CTID) software*** to be used by operators in a network operation centre (NOC). The software should be informative but also interactive with the user. The requirements as discussed with the operators of the NOC are as follows:

The software must:

a) Be implemented under OOP principles.
b) Access, read, and in general, manipulate cyber incidents from the ***malw_sample_set_2.csv*** (file is provided).
c) **Randomly** read a sample of random records (i.e., not linearly picked) in the order of 10 from that file. An operator should be able to enter the sampling rate manually to the program.
   a. i.e., a user may request 10, 100, 1000, 10000 and up to 100000 per sample to read each time.
d) Provide an output depicting a report of the total numbers of benign and malicious incidents.
e) Provide an output with the top 15 incidents in each category related to malicious activity.
   a. Top ranking incidents per category of malicious activity are related to:
      i. ***Android_Adware***: high number of packets flow/second and high number of total Fwd packets (columns U and H)
      ii. ***Android_Scareware***: high flow duration (column G)

iii. ***Android_SMS_Malware***: highest packet length variance and highest average packet size (columns AC and AD)

f) Provide a detailed view of any selected malicious activity with all the related fields of that incident in a dedicated pop-up output.
g) Allow the operator to manually include new events and access them using an incident ID.

The software should:

a) Have GUI output indicating all reports/logs as/when needed.
    a. GUI technology/library is entirely your own decision – (e.g., either local GUI or web-based using Spring Boot, REST, JavaFX, Swing etc..)
b) Interface with a database
    a. Advised to use JDBC.
c) Provide a view of malicious incidents based on the following rules:
    a. Protocol identifier
        i. e.g., UDP for 17 or TCP for 6
    b. Label
        i. i.e., based on the four labels / 3 are malicious + 1 is benign
    c. Destination port using ranges or single values
        i. e.g., 443, 4000<range<5000
d) Be optimal in terms of computational complexity – (be able to explain)
    a. i.e., response time to operator in the order of seconds.
        i. Optimal use of data structures and sorting/searching
        ii. Awareness of memory management

Explanation of the dataset *malw_sample_test.csv*:

The dataset was taken originally by ***Subhadeep Chakraborty. (2023). Android Malware Detection [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/4987461*** and contains four labels namely Android_Adware, Android_Scareware, Android_SMS_Malware and Benign. The dataset includes 355630 entries or instances (rows) with 85 columns originally that were reduced to 34 for the purpose of this coursework. The data has been scrapped from the CIC repository. Presently, the data has the following label distribution:
Android_Adware =>147443 instances, Android_Scareware => 117082 instances, Android_SMS_Malware => 67397 instances, Benign => 23708 instances

Project Rules & Marking

Requirements:

- Individual submission – the work should be developed solely by you and with no one else!
    - Submission of code + UML diagrams over Teams on November 20th by 17:00 via Teams as a .zip file

Marking Scheme:

Assessment over a demo mini-session using the code you submitted –

a. **Duration 5-7 minutes**

b. Explanation and justification of UML diagrams – 10%

i. Class diagram – 5%
ii. Use case diagram – 5%

c. Code explanation of the basic functionalities – 30%

i. Class definitions – (10%)
ii. Class dependencies – (10%)
iii. Methods – (10%)

d. Functionality Demo – 60%:

i. Demonstrate reading of random samples per user sampling rate (i.e., 10, 100 etc.) – (10%)
ii. Demonstrate breakdown to individual types of incidents over a GUI – (10%) ; if no GUI then 5%
iii. Demonstrate that you can insert a new incident and then access it. (20%)
    1. Wrong input checks and exception handling
    2. Search for incidents based on
        a. Protocol type
        b. Source port
        c. Destination port
iv. Demonstrate top ranked incidents based on their individual properties as defined earlier in the program requirements. (15%)
v. ***Bonus:*** Extra for GUI-based popups on exception handling (5%)

Penalties:

Late submission:

- Same day late submission (i.e., after 17:00 and before 23:59 on Nov 20th):
    - reduction of 10% from overall mark
- Next day late submission:
    - Reduction of 20% from overall mark
- 2+ day submission until 5 days:
    - Reduction of 50% from overall mark
- 5+ day submission:
    - Rejected and total 100% reduction from overall mark

No show policy:

- Students that do not appear for the demo with no justification of their absence will be automatically marked with 0.
    - Need to inform the ECE admin office cc'ing Prof. Marnerides in your email and explaining why you will be absent that day.

Plagiarism:

- Students submitting copied diagrams or code that they cannot explain will be investigated for plagiarism and will be automatically assigned with a 0 mark. If the case is severe a further case in the departmental council will be raised.

Notes:

1) students **requesting an extension with the appropriate justification** need to ask **at least 1 week in advance** of the submission deadline.

2) students with **last minute requests for extension without justification** will be automatically **rejected**
    a. unless an unexpected critical circumstance has emerged (e.g., unforeseen family tragedy, illness etc) that will be evaluated based on the justification provided by the student.

Tips:

- **Tip1:** a sample is the number of rows in the .csv. – it is advisable to first open the .csv in MS Excel to visualise and identify the column names and alphabetical index.
- **Tip 2:** Indicative Classes: Incident (superclass), Adware, SMS, Scareware, Benign
    - **Incident:** Columns A up to F (A-F)
    - **Adware:** columns U and H → i.e., A-F, U and H
    - **SMS:** columns AC and AD → i.e., A-F, AC and AD
    - **Scareware:** column G → i.e., A-F and G
    - **Benign:** all apart those for the malicious incidents (i.e., G, U, H, AC, AD)
    - **Note: these are only indicative classes, you may choose to develop your own classes based on your own design if you can justify them.**
- **Tip3:** when you insert a new incident, fields that are not related to the specific subclass should hold a zero or null value accordingly.
- **Tip4:** you are free to create your own indexing (e.g., local indexing through list/collection manipulation in your program, .csv manipulation on the database side etc.)
- **Tip5:** within the explanation during your demo, you should justify design choices that are aligned with your UML diagram – e.g., any notions of polymorphism or inheritance.
- **Tip6:** you should be able to explain confidently, how your data is stored within a DB, how it is accessed, how it is stored locally on your program (i.e., data structures and algorithms used) during runtime and how specific properties are implemented (e.g., random fetching of records).