Midterm Exam

CS 698M: Topics in Malware Analysis and Intrusion Detection

Take Home exam: Release: 18th February 2020

Due: March 1st, 2020, 11:55 PM (Since this is an exam – no late submission will be allowed)

Total Marks: 100

The midterm exam will be take-home programming project with 2 distinct deliverables. Each deliverable has 50 marks.

Description: You are given two sets of data – and each set contains again two distinct labeled sets.

Set 1: A directory containing files for static analysis of malware files – each malware labeled by its hash value in a separate directory which contains 3 different text files – opcodes, structure, and strings. Even though malwares are further classified into various malware classes, you may put them together as just a single class – malware. The same analysis information has been made available for the benign files. Once you download the zipped-up files and extract the directories – you must programmatically extract features for each malware as well as for each benign file. You should get 5113 feature vectors labelled as malware, and 5125 feature vectors labelled as benign files. You may have to do some feature engineering to decide which features can do better classification. This is where your knowledge of the class material will come handy. Once you have selected features and trained a model you will produce your first deliverable as described later in this document.

Set 2: A directory containing files for dynamic analysis of malware files – into json files which are labeled with the hash values of the malware. Even though malwares are further classified into various malware classes, you may put them together as just a single class – malware. The same analysis information has been made available for the benign files. Once you download the zipped-up files and extract the directories – you must programmatically extra features for each malware as well as for each benign file. You should get 5113 feature vectors labelled as malware, and 5125 feature vectors labelled as benign files. You may have to do some feature engineering to decide which features can do better classification. This is where your knowledge of the class material will come handy. Once you have selected features and trained a model you will produce your second deliverable as described later in this document.

Note: To train the model do not use all 5113 + 5125 files as you will need to test the various figures of efficacy of your models. So, keep 25% of malware and 25% of benign file data for testing purposes. We will be testing your models on 1000 new malware+ benign file data – in each case – where the data will be supplied in the same format as given to you. However, your model would not have seen these files during your training or testing phases.

Deliverable 1: Create a program named static.xxx (where xxx extension may be py , java or whatever language you use to write your code). This program should take as input the full path to a directory containing static analysis information for 1000 or so files (mix of malware and benign ware). Then your program should extract feature vectors from these files – do the feature reduction – and run your model on the feature vector – for each file. At the end you will output a .csv file with two columns – one hash of the file you test, and M/B in the second column. In other words, your model will be run on all the feature vectors in the order the data for these files are present in the data given – and once your model classifies that file as malware it will write an M, and if it classifies it as benign then it will write a B. The format of the output file will be and named static.csv and stored in the current directory.

File_Hash	Predicted Label
0a0ee0aa381260d43987e98dd1a6f4bab11164e876f21db6ddb1db7c319c5cf8	B
0a21ef18ba03622736a8edd5390afbab6088dcacc3d5877eb0b28206285f569d	M
0a2adcac2b16b02d475e9d47b4772b77b0b4269132f07557c7ef6081727585da	B
0a56a947d9c0be507b6aa0e2b569ca7eed39e5e802c8cf78be71adda9d324eae	M
0be400c66a6171749a459a5ef7fe1e0ead5380a1a97e5062b85da81e200c23ca	B

Deliverable 2: Create a program named dynamic.xxx (where xxx extension may be py , java or whatever language you use to write your code). This program should take as input the name of a directory containing .json files containing dynamic analysis information for 1000 or so files (mix of malware and benign ware). Then your program should extract feature vectors from these files – do the feature reduction – and run your model on the feature vector – for each file. At the end you will output a .csv file with two columns – one hash of the file you test, and M/B in the second column. In other words, your model will be run on all the feature vectors in the order the data for these files are present in the data given – and once your model classifies that file as malware it will write an M, and if it classifies it as benign then it will write a B. The format of the output file will be and named dynamic.csv and stored in the current directory.

We will be computing the following figures of efficacy of your models on the test data we use.

$$accuracy = (True\ Positives + True\ Negatives) / Total\ Points$$

precision= True Positives / (True Positives + False Positives)

recall= True Positives / (True Positives + False Negatives)

*F1 Score= 2 * Precision * Recall / (Precision + Recall)*

What do you deliver on canvas on or before deadline?

For each deliverable (1 and 2) – make separate directory – containing all your code. Remove all data files from the directory due to space constraint. The directory should contain code, makefile and README file which may describe any issues your code may have to comply with the requirements of input/output as specified above. Put the two directories under a directory that is your rollnumber. Zip up the top-level directory and upload on canvas.

How will you be marked?

For each deliverable

1. Program satisfies Input/Output requirements so the examination script can automatically run your code and obtain the output string of M/Bs -- 15 points
2. The figures of merits are not too divergent from standard solution to the problem – 15 points
3. On inspection, the code seems logically written, and the feature engineering and model training seems logical – 15 points
4. Code quality -- 5 points

All code will be checked with automated code plagiarism checker – any sign of plagiarism either between students, or from Internet resources are strictly prohibited. However, use of ML libraries are permitted provided in the README file – the libraries used and dependencies on libraries are mentioned.