Malware Analysis by Combining Multiple Detectors

Shankara Narayanan V

Introduction

< Introduction about the RaDaR dataset >

4 detectors from the paper Malware Analysis by Combining Multiple Detectors by Massimo Ficco are implemented on the RaDaR dataset. The implementational details and results are discussed in this report.

AFD – API Frequency Detector

In the RaDaR dataset, each hash value has an associated operation. The operation here is simply a system call performed by the binary being analyzed. A new dataframe is formed with the columns as the unique operations found across the dataset and the index of the dataframe are all the unique hash values. The corresponding cells contain the proportion of calls made by the respective hash and operation pair. The values in the cell range from 0 (not called) to 1 (only one called by the hash).

ASD – API Sequence Detector

The API Sequence Detector requires all the API calls (System calls in this case) by a particular hash. The chain of system calls are captured in a dataframe, passed to a software for aligning all the sequences to produce maximum alignment (ClustalX) and the aligned system call chains are parsed into numbers and passed to the neural network for further classification. The ClustalX software accepts only alphabets to be a part of the chains which needs to be aligned hence the top 26 system calls are filtered and other records are dropped. In this process, the dataset size reduces from 12 million records to 11.8 million records.

The encoded system call chains are then parsed to the FASTA format and fed into the ClustalX software. After the software runs the MSA (Multiple Sequence Alignment) algorithm, the aligned sequences can be downloaded and parsed back into a dataframe. The dataframe is then fitted with an ordinal encoder to convert the alphabets back to numbers facilitating training on a neural network.

NTD – Network Traffic Detector

The NTD requires the port numbers of the packets being sent by the hash values. Similar to API Frequency Detector, the columns of the dataframe will contain the port numbers and the indices of the dataframe contains the hash values of the binaries. The cells will contain the proportion of times a corresponding hash value sends a packet of the corresponding port

number. The values in the cells range from 0 (no traffic by that hash on that port) to 1 (all traffic by that hash on that port).

MCD – Markov Chain Detector

This detector uses the transition probability of system calls and classifies a binary as malware or benign. The columns of the input dataframe contain all the possible system call transitions present in the dataset. The corresponding cell will be filled according to the formula

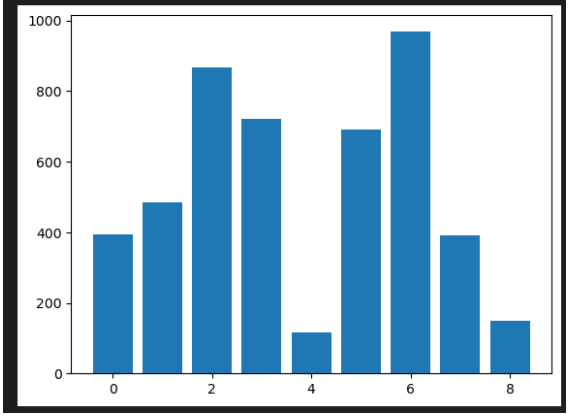
$$\frac{O(c_i \rightarrow c_j)}{\sum_{c_z=1}^{n} O(c_i \rightarrow c_z)}$$

Here, the numerator is the total number of times system call c_i -> c_j happened and the denominator is the total number of all possible system call transitions starting from c_i to any other system call invoked by that particular hash value.

Oversampling using SMOTE

The dataset needs to be oversampled by SMOTE (Synthetic Minority Oversampling TEchnique) method to avoid poor modelling of minority classes. In the RaDaR dataset, there is an inherent class imbalance

```
Class=0, n=393 (8.223%)
Class=8, n=148 (3.097%)
Class=1, n=484 (10.128%)
Class=2, n=867 (18.142%)
Class=3, n=720 (15.066%)
Class=4, n=116 (2.427%)
Class=5, n=691 (14.459%)
Class=6, n=968 (20.255%)
Class=7, n=392 (8.203%)
```

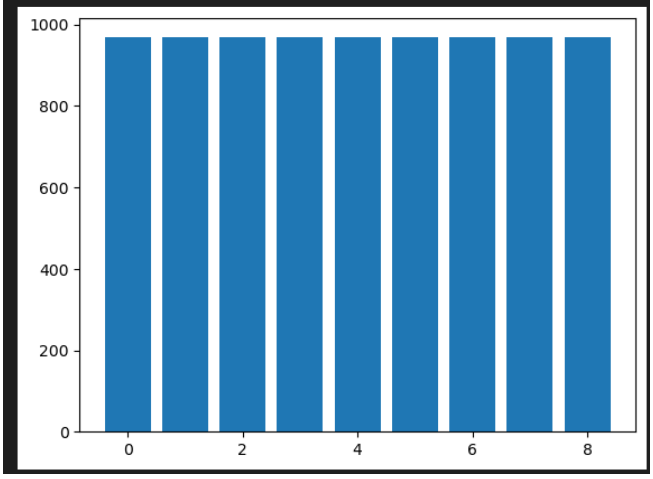


The python implementation of the SMOTE technique is available under imblearn library. The code snippet for the same is given below

```
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)
```

After oversampling the classes are equally represented as follows

```
Class=0, n=968 (11.111%)
Class=8, n=968 (11.111%)
Class=1, n=968 (11.111%)
Class=2, n=968 (11.111%)
Class=3, n=968 (11.111%)
Class=4, n=968 (11.111%)
Class=5, n=968 (11.111%)
Class=6, n=968 (11.111%)
Class=7, n=968 (11.111%)
```



Before oversampling the model may give better accuracy but that is not truly representative of the performance of the model across all the classes. The accuracy may be expected to drop after oversampling but prediction precision and recalls for all the classes will quite high compared to the imbalanced model.

The final set of models

Generic model

The generic model classifies all the samples as either benign or malware. Because of this reason, all the encodings with values other than 0 are considered as 1. This brings about a case of class imbalance.

```
Percentage of ones:  88.88888888888889
Percentage of zeros:  11.11111111111111
```

This can again be fixed using the SMOTE technique

```
Percentage of ones:  50.0
Percentage of zeros:  50.0
```

Then a neural network is fit on the balanced dataset, with BinaryCrossentropy as the loss function and Adam optimizer. After 100 epochs, the test loss and accuracy is as below,

```
Test loss:  0.35030415654182434
Test accuracy:  0.8247256278991699
```

Banker classifier model

A specialized detector is made with Markov Chain Detector as the base model for classifying Banker type malware. The input to this model contains records that are either benign or Banker malware. After fitting a shallow neural network, the following results are observed.

```
Test loss:  0.2590281069278717
Test accuracy:  0.8324742317199707
```

Spyware classifier model

A specialized detector is made with Markov Chain Detector as the base model for classifying Spyware type malware. The input to this model contains records that are either benign or Spyware malware. After fitting a shallow neural network, the following results are observed.

```
Test loss:  0.38887566328048706
Test accuracy:  0.8608247637748718
```

Backdoor classifier model

A specialized detector is made with Markov Chain Detector as the base model for classifying Backdoor type malware. The input to this model contains records that are either benign or Backdoor malware. After fitting a shallow neural network, the following results are observed.

```
Test loss:  0.564724862575531
Test accuracy:  0.8144329786300659
```

Ransomeware classifier model

A specialized detector is made with Markov Chain Detector as the base model for classifying Ransomware type malware. The input to this model contains records that are either benign or Ransomware malware. After fitting a shallow neural network, the following results are observed.

```
Test loss:  0.5075554847717285
Test accuracy:  0.7938144207000732
```

Potentially Unwanted Applications classifier model

A specialized detector is made with Markov Chain Detector as the base model for classifying Potentially Unwanted Applications type malware. The input to this model contains records that are either benign or Potentially Unwanted Applications malware. After fitting a shallow neural network, the following results are observed.

```
Test loss:  0.5488463640213013
Test accuracy:  0.7654638886451721
```

Downloader classifier model

A specialized detector is made with Markov Chain Detector as the base model for classifying Downloader type malware. The input to this model contains records that are either benign or

Downloader malware. After fitting a shallow neural network, the following results are observed.

```
Test loss:  0.5849368572235107
Test accuracy:  0.8221649527549744
```

Deceptor classifier model

A specialized detector is made with Markov Chain Detector as the base model for classifying Deceptor type malware. The input to this model contains records that are either benign or Deceptor malware. After fitting a shallow neural network, the following results are observed.

```
Test loss:  0.5844281911849976
Test accuracy:  0.7319587469100952
```

Cryptominer classifier model

A specialized detector is made with Markov Chain Detector as the base model for classifying Cryptominer type malware. The input to this model contains records that are either benign or Cryptominer malware. After fitting a shallow neural network, the following results are observed.

```
Test loss:  0.1886872947216034
Test accuracy:  0.969072163105011
```

Meta-Learner

The final meta learner takes input from a new dataframe. This dataframe contains 9 columns, each column is the prediction on the test split of the main dataset from the base models mentioned above. The last column is the target column from the test split dataframe. After fitting a shallow neural network as a meta leaner, the following results are observed.

```
Accuracy: 0.9248422384262085
```

Consolidated results table

Final Ensemble Model -> Markov Chain Detector chosen as base model

Model	Validation Accuracy after 10 epochs
Generic detector (Malware or Benign)	82.47%
Specialized detector for Banker	83.24%
Specialized detector for Spyware	86.08%
Specialized detector for Backdoor	81.44%
Specialized detector for Ransomware	79.38%
Specialized detector for PUA	76.54%
Specialized detector for Downloader	82.21%
Specialized detector forDeceptor	73.19%
Specialized detector forCryptominer	96.91%
Final meta learner	**92.48%**