List of contents

INTRODUCTION

Malware

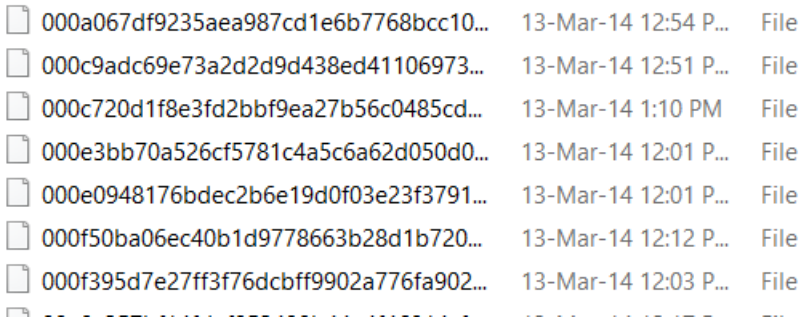
Cyber attacks and malware are one of the biggest threats on the internet . Malware is shorthand for malicious software. It is software developed by cyber attackers with the intentions of gaining access or causing damage to a computer or network, often while the victims remains oblivious to the fact there has been a compromise. A common alternative description of malware is Computer virus although are big differences between there types malicious programs. In this Report we are going to discusses the method to detect if it is a malware or not a malware. Machine learning is suitable possibility to provide that.

There are a lot of methods how to detect the malware but below is going to be introduced how is analyzed by SVM(Support vector machine) and Random Forest classifier which both of them are very powerful methods.

DREBIN DATASET

This dataset is public and available for anyone. Is used to extract features from every sample application from a dataset. Every application creates one data file with the belonging features. The dataset contains more than 120.000 hamware and 5.560 malware applications.

To detect if the sample is hamware or malware there is a sha256_family.csv file in the drebin dataset.



In that file are listed all malware filenames. By comparing the sample name and the malware list it is possible to divide the dataset into malware and hamware. The csv file contains also the malware family to every sample name. The name of every file is a SHA1 the hash code.

The encoded features are presented with a row. Every row represent one feature. Every example can contain different features and values in every feature. The value of every feature can be detected after the ' :: '.

```
feature::android.hardware.touchscreen
url::https://cc.blueshoemobile.com/cc/
permission::android.permission.ACCESS_FINE_LOCATION
api_call::org/apache/http/impl/client/DefaultHttpClient
permission::android.permission.CALL_PHONE
activity::com.mobilegrub.android.OrderInstructionsActivity
permission::android.permission.INTERNET
real_permission::android.permission.ACCESS_FINE_LOCATION
```

The features are divided into eight features classes

Prefix	***Set***
feature	*1 : Hardware Components*
permission	*2 : Requested Permission*
activity *service_reciver* *provider* *service*	*3 : App Components*
intent	*4 : Filtered Intents*
api_call	*5 : Restricted API calls*
real_premission	*6 : Used Permission*
call	*7 : Suspicious API calls*
url	*8 : Network Addresses*

LIBRARIES

The following open source libraries were used in the code.

1. Numpy- scipy for reading data and building matrices
2. Scikit-learn for machine learning algorithms and utilities
3. Matplotlib for ploting

FEATURE SELECTION AND EXTRACTION

In the code has been used both of features methods.

The feature selection, this technique is used for selecting the features which explains the most of the target variables (has a correlation with the target variable). This test is ran just before the model is applied on the data

To explain it better I want to give an example. There are 10 features and 1 target variables, 9 features explain 90 % of the target variables and 10 features together explains 91 % of the target variables. So the 1 variable is not making much sense for us. So you remove it before modeling. We can also call it as Predictor importance.

Now lets talk about feature extraction. Is the process of transforming the input data into a set of features which can vary well represent the input data. It is a special form of the dimensionality reduction. When the data is to large to processed, the data will be transformed into a reduced representation set of features. The process of transforming the input data into the set of features is called FEATURES EXTRACTION.

```
"""
Create feature vectors
"""
FEATURES_SET = {
    "feature": 1,
    "permission": 2,
    "activity": 3,
    "service_receiver": 3,
    "provider": 3,
    "service": 3,
    "intent": 4,
    "api_call": 5,
    "real_permission": 6,
    "call": 7,
    "url": 8
}
```

Here has been created the features vectors

CLASSIFICATION

The main intention of evaluating the dataset is in one binary output decision malware or hamware. To get to that solution there are many possible ways. Because we have a dataset with the binary output to every sample out of the csv file it can be use supervised learning method. The dataset provides features of every sample, thus that should be the input of the learning algorithms.

D = Dataset

Xi → Features Values

Yi → Decision malware or hamware

CLASSIFIER

There are a lot of classifier methods but here has been used the SVM(Support vector machine) and Random forest classifier. I could use the Naïve Bayes method but it can not read all the files in drebin dataset. I have used the libraries.

The SVM performs classification by finding the hyperplane that maximizes the margin between the two classes. The SVM has been tested and able to outperform logistic regression only with default hyperparameters.

Random Forest A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with

replacement if bootstrap=True (default). The RF algorithm usually perform very well in practice. However this algorithm is complicated to tune. So only two parameters n_estimator max_features were decide to tune. This action increase the accuracy.

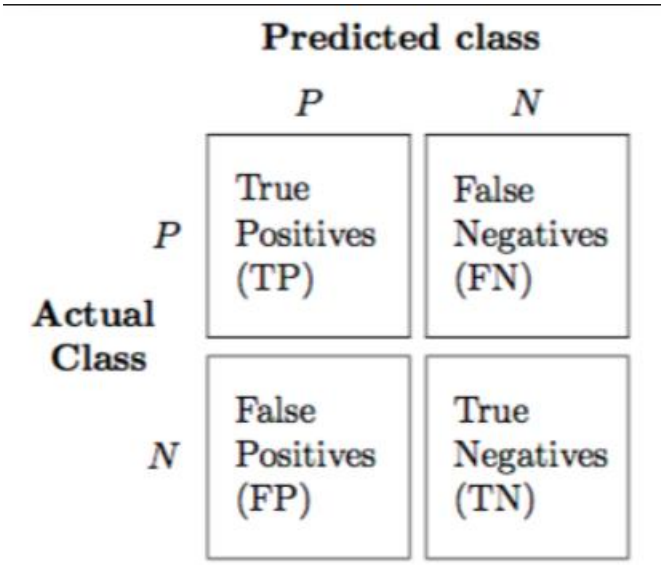
Performance

	F1 Score	Accuracy
SVM	0.904	0.904
Random Forest	0.937	0.937

EVALUATION PROCEDURE AND RESULTS

Confusion Matrix

The confusion matrix is a useful table to evaluate a machine learning algorithm. Every sample classify belongs to one case of the confusion matrix. After the execution the matrix contains the amount of cases at the execution of the algorithms.



Interpret the matrix in our case for Malware or not

TP -> It is malware, I correctly detected it.

FP -> It is not malware, but I thought it was

TN -> It is not a malware, and I thought sot too

FN -> it is malware, but I didn't detect it

```
[[1482  181]
 [ 139 1534]]
```

Result of Confusion Matrix default parameters

```
[[1568   95]
 [ 192 1481]]
```

Result of Confusion Matrix given parameters

Two different method are used to detect the malware the accuracy is almost the same between these two Random forest performed better than support vector machine , although they both out performed logistic regression.

CONCLUSION

Machine learning algorithms are very useful in the context of malware classification. Some of them seems to perform very well in practice given the right feature vectors. They also provide interesting insights about how an application should be examined for malware detection . for example, random forest suggested the number of required hardware components and suspicious API calls are very important.