O.S android-malware

Hillel Ohayon 325300820
Tom Koltunov 323929190

1 Introduction

In this assignment, we examined our classifier and found a few weak spots that could help us exploit them and create an efficient attack. We explored the academic paper that gives a better point of view to what's happening in the model when it processes the .apk files and output .txt files in turn.

2 Weak spot

There are the two main weak spots that we explored in the classifier, we will exploit the weak spots by creating an attack that uses them to decrease the recall metric.

2.1 Structure data type

When a folder which contain apk files is passed as an argument to the classifier, it goes over them by the order and that is to say it labels them by the order that they were given.
In the class we learned about "Structure break attack" that makes an unexpected change over time in the order of files, which can lead to huge forecasting errors and unreliability of the model.

2.2 The effect of training set size

In the academic pepper, we saw an interesting fact. Given the importance of the training phase, it could be argued that the size of training set that they have used in large-scale experiments is too small compared to the size of the testing set. They found a fact that has shown that the Recall rates actually decreases when the size of the training set increases.
That is a significant weak spot and we can exploit it by controlling the size of the datasets and not by changing the features. To find out the weak spot, we created a python file which use to split the folders of the dataset to a random subset in size K. After splitting the folders several times, we saw that when we increase the train data, the model's accuracy and the recall significantly decrease and also the recall.

3 Method of Attacking using aforementioned weak spots

As we've seen, If we give more and more train data to our model it makes it's recall drop We'll do this along with the shuffling technique to create an attack on the CBSD model. When thinking of ways to implement this, I'll create a model that whenever CBSD receives training data it preprocesses the trained data and shuffles the labels according to the constraints that make sure that the model isn't in a state of over-fitting for example.In conclusion Every time CBSD is trained, our attack will augment the data so CBSD's recall will drop.

4 GIT

The second assignment is published in the follow github link:(ignore the clone of the classifier he isn't different from the classifier from the link in the mudel.)// Link to the github: https://github.com/HILLELOH/csbd-O.S-android