Evading the Machine Learning Detector: A Virus Perspective

Group 10
Yuan Xiao, Xiaokuan Zhang

04/27/2017

Abstract

1 Introduction

1.1 Problem Description

Our group wants to solve the problem of classifying hand-written digits and black-and-white rectangular pixel-displayed letters. We have two datasets from UCI machine learning repository [3]. One is hand-written digits dataset [1]. Another is letter recognition dataset [2]. We will introduce them in more detail in Section **??**.

1.2 Classification Algorithms

1.3 Hypothesis

1.4 Originality

1.5 Result Summary

2 Background

3 Methodology

4 Experiments

The whole experiment is made up of four steps. First, data preprocessing. Second, model training with LibSVM and testing with unmodified test set. Thus we have a criterion for evaluation. Third, data modification in the test set and testing using trained libSVM model. Fourth, performance comparison. The methods for

model training and data modification are already described in section 3 and the rest parts will be discussed in the this section.

4.1 Original Dataset

The dataset includes instances of malicious executables (computer virus) as well as non-malicious executables (normal programs). The features are extracted from real-world malicious and non-malicious executables. The dataset was published in March 2016 in UCI Machine Learning Repository. Within one year, the web hits of the dataset has already reached over 200,000. It is obvious that the dataset arouses great interest in the machine learning community.

The dataset consists of 373 instances, of which 301 are malicious and 72 are benign. Each instance has 500 hex features and 30 DLL features. Notice that on the UCI webpage it claims that there are 13 DLL features but we found from the raw data that there are actually 30 of them. All the attributes are binary, meaning a certain feature exists or not.

However, the primary weakness of the dataset is that the sample number is relatively low compared to other datasets. However, the goal of our project is not to exhaustively find the best machine learning model to classify malicious and non-malicious executables. Thus we think the low sample amount is tolerable. In addition, after searching online, we thought the UCI dataset is the only one available of such type of data.

4.2 Data Preprocessing

The raw data is generally LibSVM-format conformant. The class attribute at the beginning of an instance marks whether it is benign or malicious. -1 stands for a malicious instance and +1 means benign. However, at the end of each instance there is an additional -1. We need to remove it before passing it to LibSVM as input.

In addition, we randomly seperate the dataset into two sets. One consists of 80% of the dataset as the training set while the rest as the test set.

4.3 Experiment Conditions

After data preprocessing, we pass the training set to LibSVM to train a classification model. With the different cores, we will have different models. The following implementations apply to all of the models.

Given a trained classification model, we apply it to the test set first and get the test results as a criterion for later evaluation. With the unmodified test set, we expect the performance of the model to classify malicious instances to be high enough. Otherwise, the model itself does not make sense and it is pointless to fool an inaccurate classification model.

After that, we apply different attribute modification methods to the test set. Notice that only malicious instances are modified since our only goal is to disguise a malicious executable as a benign one. With the different modified test sets, we re-do the testing phase with unmodified classification model. Then we compare the new test results with those of original test set and see whether any malicious instances successfully escape the detection.

In the end, we compare the escape results of different modification methods to find which of them perform best and how to achieve a best performance. Meanwhile, we also do comparison of such modifications with existing anti-antivirus techniques for real-world feasibility discussion.

4.4 Results

4.5 Result Analysis

5 Discussion

5.1 Real-world Feasibility

In this section we briefly discusses the feasibility of our modification towards the malicious executables and compare them with existing anti-antivirus methods deployed by real-world virus.

In reality, it is much more difficult to remove features than to add features. The features to remove mostly denote the malicious behavior of a virus and are thus hard to take out. In contrast, we can easily add features as dummy code that will never be executed.

Our modification schemes in fact serves as a combination of two existing anti-antivirus methods. The first is to disguise as popular file formats such as .pdf or .docx or programs such as calc.exe or notepad.exe. The second is polymorphic virus. It mutates on each copy by adding different types of NOP instructions.

References

[1] Optical recognition of handwritten digits data set. `http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits`.

[2] Optical recognition of handwritten digits data set. `https://archive.ics.uci.edu/ml/datasets/Letter+Recognition`.

[3] M. Lichman. UCI machine learning repository, 2013.