Poster: Automatically Evading Classifiers

A Case Study on Structural Feature-based PDF Malware Classifiers

Weilin Xu, Yanjun Qi, and David Evans
University of Virginia
{xuweilin, yanjun, evans}@virginia.edu

***Abstract*—Machine learning methods are widely used in security tasks. However, the robustness of these models against motivated adversaries is unclear. In this work, we propose a generic method that simulates evasion attempts to evaluate the robustness of classifiers under attack. We report results from experiments automatically generating malware variants to evade classifiers, from which we have observed non-robust features result in vulnerable classifiers. This suggests the proposed evasion simulation method will help to improve the robustness of classifiers by locating weak spots of learning models.**

I. Introduction

Machine learning models are popular in security tasks such as malware detection, network intrusion detection and spam detection. From data scientists' perspective, these models are effective, since they achieve extremely high accuracies. For example, Dahl *et al.* trained an ensemble deep neural network with dynamic features for the Win32 malware classification that achieves 99.58% accuracy [5]. Šrndic *et al.* trained a SVM-RBF model with accuracy of 99.958% in a PDF malware classification task by using structural path features [8].

However, it is important to understand that these results are specific for given test datasets. Unlike the application of machine learning in other fields, security tasks involve adversaries responding to the classifier. For example in malware detection, attackers will always try to generate new malware samples that have different patterns to evade existing classifiers. This breaks the assumption of machine learning models that training set and testing set share the same data distribution, consequently decreasing the accuracy in practice. As a result, it will be beneficial to model attackers' efforts in evaluating the robustness of classifiers in security tasks.

In this work, we propose a generic method to evaluate the robustness of a classifier by simulating attackers' efforts in evasion attacks. Taking the idea from *genetic programming* (GP), we don't need to design any manipulation strategy on malicious samples with security experts' knowledge. Instead, we perform random manipulations and then evaluate the generated variants to select positive ones. By repeating this procedure iteratively, we aim to find evasive variants.

The proposed method has been demonstrated to work effectively on two PDF malware classifiers by successfully generating evasive variants. The evasive malware exhibits the same malicious behavior as the original sample, however, they have widely different patterns in feature spaces, resulting in being classified as benign by the machine learning-based model. Further examination shows it is because of the employment of non-robust features, which directs a hopeful way to build more robust classifiers by eliminating these features.

II. Approach

Genetic programming (GP) is a type of evolution algorithm, originally developed for automatically generating computer programs. Recently, Le Gous *et al.* showed that GP can also work on manipulating programs in fixing software bugs [6]. We propose a GP-based method that can manipulate existing malicious samples to evade classifiers. Our goal is to simulate what an attacker would do to evade the malware classifier.

The GP-based procedure is illustrated in Figure 1. First, we initiate a population of variants by randomly manipulating a seed malicious sample that exhibits malicious behavior and is classified as malicious by the target classifier. Our method aims to find an evasive sample that preserves the malicious behavior but is mis-classified as benign by the target classifier.

Subsequently, the population is evaluated by a target classifier as well as an oracle. If any variant is classified as benign but the oracle reports that it behaves maliciously it is deemed an evasive sample. Otherwise, some variants are selected but others are eliminated based on a fitness measure. Next the selected variants are randomly manipulated by mutation and crossover operators to produce next generation of the population. The loop continues until an evasive sample is found or a threshold number of iterations is reached.

The fitness function is designed as monotonically increasing as the success of evasion increases in order to eliminate the hopeless variants in each generation. There should be a threshold value of fitness score that indicates the evasion attempt succeeds, which is determined by the output structures of the oracle and the target classifier.

III. PDF Malware Experiment

We conducted an experiment using our approach on two PDF malware classifiers, respectively PDFRATE and Hidost. PDFRATE is a Random-Forest model using meta data and

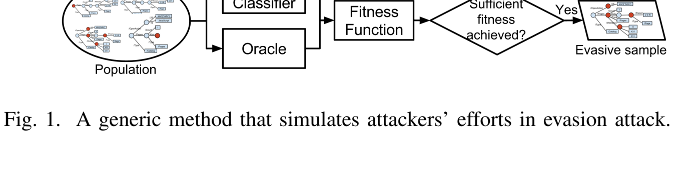

Fig. 1. A generic method that simulates attackers' efforts in evasion attack.

structural information as features [9], while Hidost is a SVM learning model employing structural paths as features [8].

A. Implementation

To implement the method in PDF malware's case, we need to understand PDF format and define an oracle as well as a fitness function. PDF file is organized as a tree-like structure of objects internally. An object can be any visible element in a PDF document, or can be Javascript code and other embedded executables. PDF malware typically contains objects of malicious payloads in random locations of the tree-like structure. We use *pdfrw* [2], a python-based open source library to convert a PDF file to the tree-like structure and vice versa, so the variants can be manipulated by GP operators and can be saved as the equivalent file representation.

We use Cuckoo sandbox [3] as the oracle. Cuckoo sandbox run each submitted sample in a virtual machine with a PDF reader, and reports the malicious behaviors that match the known signatures.

Let's formulate the oracle as a binary function $oracle(x) = 1$ if the maliciousness is preserved and $oracle(x) = 0$ otherwise. We define the evasion threshold as 0, then the fitness function can be designed based on the output structure of the target classifier. As PDFRATE outputs confidence value of maliciousness from 0 to 1, we define the fitness function as $f_{pdfrate}(x) = (0.5 - pdfrate(x)) \times oracle(x)$. In contrast, as Hidost outputs positive (negative) distance of a benign (malicious) sample to hyperplane, the fitness function is defined as $f_{hidost}(x) = hidost(x) \times oracle(x)$.

In terms of GP parameters, we specify the population size as 48, mutation rate as 0.3, crossover rate as 0.0 and the threshold iterations as 10.

B. Results

We selected 50 malware samples from Contagio archive [1], all of which are recognized as malware by Cuckoo sandbox (with a virtual machine of Windows 7 and Adobe Reader 8.1.1) as well as the two classifiers, and can be correctly repacked by *pdfrw*.

After an approximately 16 hours of execution, the method generates variants for all 50 malware samples that evade PDFRATE. For Hidost, it took about 14 hours to achieve an evasion rate of 100% on the 50 samples.

By comparing the pairs of original malware samples and their evasive variants in feature space, we found classifiers rely on some features that can be easily manipulated by an attacker without losing the malicious behavior. For example, the number of font objects is not relevant to malicious behavior, but it is the most critical feature for some evasive samples against PDFRATE. The classifier learns this feature because most of the malware in the training set contain no font objects as the malware authors are too lazy to insert any text. However, this is an artifact of the malware samples used to train the classifier, not an inherent property for malicious PDFs. It is trivial to add more font objects to an existing PDF malware sample to evade the classifier. Our future work will use the automated tool for finding evasive sample to improve the robustness of classifiers. We intend to explore both retraining the classifier using the find evasive samples, as well as using the evasive sample to identify non-robust feature that should be removed from the classifier.

IV. Related Work

Evasion attacks against machine learning classifiers have been discussed by Biggio *et al.* from the angle of classification models [4] and other malware classifier authors such as Šrndic *et al.* [8]. However, these studies assumed that attackers can only insert features. In fact, the experiments in our work show attackers also have the ability of removing features, which leads to evade classifiers.

Šrndic *et al.* demonstrated how PDFRate could be evaded by exploiting an implementation flaw in the feature extraction [9]. Our method does not rely on any implementation flaw. Instead, it captures the weak spots in model's feature space.

In addition, Maiorca *et al.* proposed reverse-mimicry attacks against PDF malware classifiers [7]. The attack is generic to a class of classifiers based on structural features. However, the hand-crafted attack only works on malware with simple payloads. In contrast, our GP-based method is automatic and does not have this limitation.

V. Conclusion

We propose a generic method that automatically evades classifiers in security tasks. The experiments on real classifiers have shown the traditional approach of building machine learning classifiers fails against determined adversaries. As future work, we are going to improve the efficiency of the method and better utilize the information from evasive samples to build more robust classifiers.

References

[1] http://contagiodump.blogspot.de/2010/08/malicious-documents-archive-for.html.

[2] https://code.google.com/p/pdfrw/.

[3] http://www.cuckoosandbox.org/.

[4] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *ECML-KDD*. 2013.

[5] George E Dahl, Jack W Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *ICASSP*, 2013.

[6] Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, and Westley Weimer. Genprog: A generic method for automatic software repair. *IEEE Trans. on Software Engineering*, 2012.

[7] Davide Maiorca, Igino Corona, and Giorgio Giacinto. Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious pdf files detection. In *ASIACCS*, 2013.

[8] Nedim Šrndic and Pavel Laskov. Detection of malicious pdf files based on hierarchical document structure. In *NDSS*, 2013.

[9] Nedim Šrndic and Pavel Laskov. Practical evasion of a learning-based classifier: A case study. In *Oakland*, 2014.