Poster: Attacking Malware Classifiers by Crafting Gradient-Attacks that Preserve Functionality

Raphael Labaca-Castro
Research Institute CODE
Bundeswehr University Munich
Munich, Germany
raphael.labaca@unibw.de

Battista Biggio
Dept. of Electrical and Electronic Engineering
University of Cagliari
Cagliari, Italy
battista.biggio@unica.it

Gabi Dreo Rodosek
Research Institute CODE
Bundeswehr University Munich
Munich, Germany
gabi.dreo@unibw.de

ABSTRACT

Machine learning has proved to be a promising technology to determine whether a piece of software is malicious or benign. However, the accuracy of this approach comes sometimes at the expense of its robustness and probing these systems against adversarial examples is not always a priority. In this work, we present a gradient-based approach that can carefully generate valid executable malicious files that are classified as benign by state-of-the-art detectors. Initial results demonstrate that our approach is able to automatically find optimal adversarial examples in a more efficient way, which can provide a good support for building more robust models in the future.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; Software reverse engineering; • **Computing methodologies →** *Neural networks.*

ACM Reference Format:
Raphael Labaca-Castro, Battista Biggio, and Gabi Dreo Rodosek. 2019. Poster: Attacking Malware Classifiers by Crafting Gradient-Attacks that Preserve Functionality. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3319535.3363257

1 INTRODUCTION

Deep neural networks have been widely implemented for malware classification [4, 7, 10]. Yet, problems still persist when they are confronted to adversarial examples. These carefully crafted perturbations injected into malicious software can make the classifier label malware as actual benign software.

Even though most of the literature has been focusing in the image domain, adversarial learning for malware evasion is also a promising area of research with recent contributions in the literature [1, 3]. Nevertheless, there are important differences between these two domains, among others, the fact that datasets are more accessible in the image domain (e.g.; MNIST, ImageNet) and the nature of the input file. In the image domain files can be easily manipulated with perturbations that are often invisible to the human eye without causing major issues to the structure of the files. On the other hand, the malware domain is more complex to work with when generating adversarial examples. The lower entropy of the data, although encryption and compression often return higher entropy scores, and the fact that many values can be binary instead of continue real numbers, such as API calls, can affect the ability to generate valid adversarial examples. The complexity can vary greatly and depends on the structure of the input file. However, evading the classifier is not only the important point since modifying the structure of malicious files plays a big role. Hence, the perturbations need to be carefully created in order to preserve the functionality of each malware sample.

In this work, we present how to build evasive examples against static state-of-the-art classifiers. We implemented a convolutional neural network (CNN) to perform the malware classification [8]. The model was trained using the EMBER dataset, which consists of 2351 features representations of more than one million portable executable (PE) files. Among them 300.000 malicious, 300.000 benign, 300.000 unlabeled, and 200.000 test samples [2].

In order to generate the adversarial examples we implemented an internal module, which is able to take a PE malicious file as input and perform the injection of a sequence of perturbations in order to achieve an evasive malware [6]. In this case, instead of relying on finding the optimal random sequence of perturbations to achieve misclassification, we analyze the value of the gradient for every perturbation injected and use it to maximize the evasion rate of the adversarial example generated. In this way, our approach is able to conveniently identify which of the transformations return a better score and, hence, generates faster adversarial examples for any given malicious file provided as input.

Furthermore, we intend to analyze the cross-evasion results of our optimal adversarial examples by testing them against state-of-the-art classifiers of the literature [2]. This demonstrates how successful it is to bring results from one classifier to another without the need to create tailored adversarial examples for each classifier targeted.

2 RELATED WORK

Biggio et al. [3] proposed a simple and effective gradient approach that is able to evade PDF file classifiers. Given that PDF files have a flexible logical structure, they are a good option for adversaries to use as infection vector by adding malicious routines. Several approaches with different levels of knowledge were evaluated and

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11.
https://doi.org/10.1145/3319535.3363257

one of the takeaways is that the false negative rate increases the more the PDF will be modified. Support Vector Machines (SVM) and neural networks were evaluated but the approach can be applicable to any other model with differentiable functions.

In [1] the authors proposed the use of a reinforcement learning (RL) agent to find the best perturbations applied on PE files. It is reported that approximately a fourth of the files achieved evasion. In addition, retraining the malware detector using the adversarial examples can reduce effectiveness of the attack by 33% according to the authors. However, they acknowledge that these results can depend on the model and dataset used. Furthermore, even though the perturbations applied are meant to be preserving-functionality transformations they do not provide an additional mechanism to make sure the generated output files are valid.

In AIMED [5] we proposed the use of genetic programming (GP) to find optimal sequences of perturbations that will be injected into the malicious file in order to generate adversarial examples. As an optimization strategy, GP seems to converge much faster than classic stochastic approaches and the number of corrupt files generated is drastically reduced. Furthermore, an extra control is added in order to ensure the functionality of the evasive files that were automatically created. However, this approach can lead to local minima and maxima, which can prevent finding a suitable vector of perturbation for some of the input samples. Another important point is that unlike PDFs, by working with PE files, the false negative rate of the queried model does not seem to necessarily increase the more perturbations the file receive. What in fact increases is the likelihood of a corrupt file despite of its detection rate. Thus, even working in the same domain, having different file structures, such as PDF and PE, forces to further adjust the evasion strategy.

3 METHODOLOGY

CNNs are a particular type of feed-forward neural networks where the pattern of connectivity used with its neurons resembles the biological visual cortex. The architecture consist of three types of layers: convolutional, pooling, and fully-connected. Unlike regular neural networks, convolutional networks do not have only fully-connected layers. The convolutional layer will compute a dot product between the weights and a part of the input. Then, the information from the input will be passed along the pooling layer where it will be downsampled and, finally, the fully-connected layer will calculate the scores [9]. The CNN is trained on the EMBER dataset using 1.1 million instances and 2351 features. There are eight groups of them, which include parsed and agnostic-format features. The former focuses on general and header information of the file as well as imported and exported functions whereas the latter refers to string information and byte-histograms [2]. The performance of the malware classifier after being trained on the dataset is displayed on Fig. 2 along with the train loss. We report the results calculated for 40 epochs and used the area under the receiver operating characteristic curve (ROC AUC) metric to measure the rate of successful classification by the CNN as it makes easier for further comparison with other models. The reported performance consists of a f-score of 94% at 5% false positive and 7% false negative rates. Additional tuning can still be performed to improve, mostly, the true positive rates when comparing to the performance of the

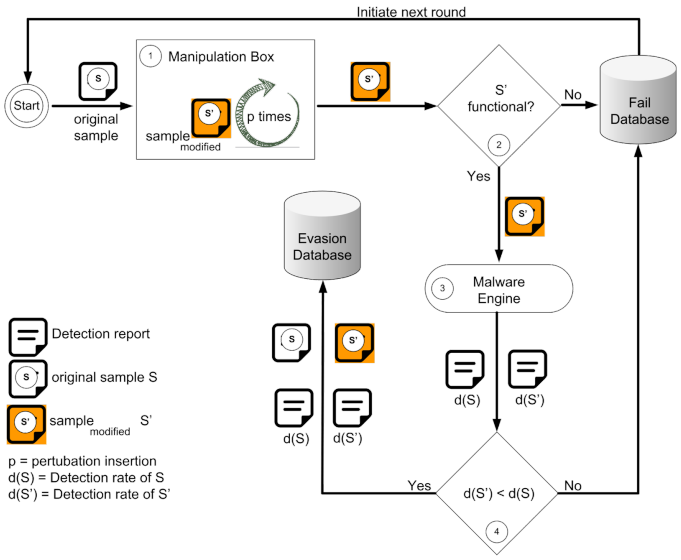

Figure 1: Workflow to generate adversarial examples [6]

adversarial examples against further classifiers as well as to further stabilize the loss function output.

3.1 Perturbation Injection

We implement, as depicted in Fig. 1, an internal module to generate malicious adversarial samples that are able to evade the targeted malware classifier [6]. During Step 1 a malware sample s is sent to the manipulation box where a sequence of defined perturbations will be injected. Once the process is done, it will generate a modified sample s' and it proceeds to the next step to check whether the file is still valid. In Step 2 a sandbox will be used to dynamically test the file. In case s' is corrupt, the malware is dismissed and the process is restarted. Otherwise, it triggers Step 3, where the sample is checked for detection. Both s and s' are sent to Step 4 and its detection results will be compared. Assuming the process was successful, $d(s') < d(s)$, an adversarial example is produced that has a lower detection rate than the initial malware sample. This workflow will be useful when generating the adversarial examples needed to optimize the gradient-based attack.

3.2 Gradient-based Attack

Following the workflow in Fig. 1 we inject defined perturbations to a previously detected malware in order to achieve misclassification by the model. In this case, instead of attacking a black-box classifier we will be targeting a CNN from which we have full knowledge of its parameters and hyperparameters. That allows us to extract the gradient information of every perturbation and calculate which one of them maximizes the likelihood of evasion.

We characterize the notations as per [3] where a classification model is defined as $f : \mathcal{S} \longmapsto \mathcal{Y}$ in the feature space $s \in \mathcal{S}$. The samples represented in the feature space are assigned to a label $y \in \mathcal{Y} = 0, 1$ where 0 denotes a benign and 1 a malicious PE file. A continuous discriminant function $g : \mathcal{S} \in \mathbb{R}$ is used to output the labels. Therefore, we assume $f(s) = 0$ if $g(s) < 0$ and $f(s) = 1$ if

$g(s) > 0$. For any sample s, the strategy is to minimize $g(\cdot)$ or the estimation of it $\hat{g}(\cdot)$ in order to achieve an adversarial sample s' based on the following expression:

$$e = \arg\min_{s} \hat{g}(\cdot) \;\; s.t. \;\; d(s, s') \leq d_{max} \tag{1}$$

We selected gradient descent as the technique to approach this problem. Despite of its good performance, it is also known that optimizing locally can be prone to failure due to the nature of the discriminant function. However, given our rather large training dataset we expect to avoid such issues. All the attacks performed have the d_{max} constraint as the transformations created must always respect the maximum distance: $d : \mathcal{S} \times \mathcal{S} \longmapsto \mathbb{R}^+$. The correct distance requires domain knowledge and, in the PE domain, needs to properly reflect the limitations of the perturbations.

4 SOLUTION

We propose a solution based on generating valid PE files that are able to evade malware classification (Fig. 2). Windows portable executables are widely used by adversaries in order to perpetrate attacks and are therefore one of the most important infection vectors. Nevertheless, manipulating them with actions such as removing objects can quickly lead to corrupt files. Thus, we implement perturbations only as additions or modifications to the PE.

The algorithm returns a score for each of the perturbations and calculate the value of the gradient for the next iteration. This approach leads us to calculate in a much more efficient way the best evasive sequence vector of perturbations that needs to be injected into a PE file in order to achieve an evasion. An important aspect is that the functionality can be preserved after the process and the system only outputs valid modified versions of the input malware.

Until now, most of the approaches mentioned worked with black-box classifiers, which brings us a realistic scenario from an adversary perspective but it prevents us to understand how the classifier responds specifically to each of the transformations. Conversely, attacks on white-box models do not normally produce real files. Therefore, our approach can be useful to not only achieve better and faster adversarial examples but also to further understand the weaknesses of the model in order to build more robust classifiers for malware detection. Furthermore, the valid PE files generated can be used for additional purposes. First, for retraining the model in order to determine what is the ratio of improvement when the classifier is able to learn from the adversarial examples generated to evade it. Second, to evaluate cross-evasion capabilities of the adversarial examples.

5 CONCLUSION

In this work, we proposed the ability of generating valid adversarial malware examples against convolutional neural networks using gradient information. By implementing attacks on the EMBER dataset, we determined that using the gradient to find optimal evasions is a promising strategy to improve adversarial attacks on real malware. Our initial experiments showed that we are able to use that information to find optimal sequences of transformations without rendering the malware sample corrupt.

As future work, we aim to improve the efficiency of the system by using another neural network to estimate the distribution instead of extracting the features and calculating the gradient for each malware sample.

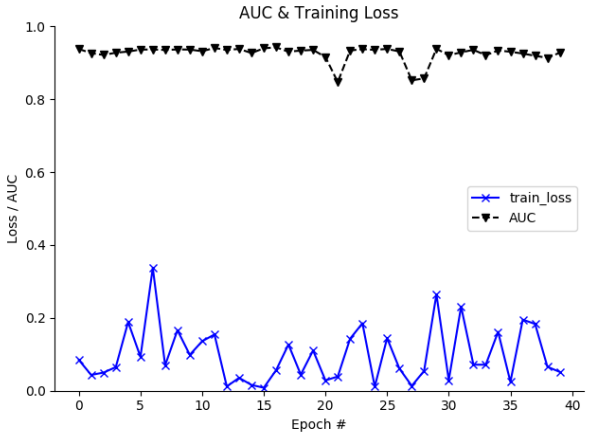

Figure 2: ROC AUC & training loss reported by classifier

6 ACKNOWLEDGMENTS

The authors would like to thank the Chair for Communication Systems and Network Security as well as the Research Institute CODE for their comments and improvements. Research supported, in parts, by EU H2020 Project CONCORDIA GA 830927.

REFERENCES

[1] Hyrum S. Anderson, Anant Kharkar, Bobby Filar, David Evans, and Phil Roth. 2018. Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning. *CoRR* abs/1801.08917 (2018). arXiv:1801.08917 http://arxiv.org/abs/1801.08917

[2] Hyrum S. Anderson and Phil Roth. 2018. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *CoRR* abs/1804.04637 (2018). arXiv:1804.04637 http://arxiv.org/abs/1804.04637

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, Heidelberg, Germany, 387–402.

[4] Bojan Kolosnjaji, Apostolis Zarras, George Webster, and Claudia Eckert. 2016. Deep learning for classification of malware system call sequences. In *Australasian Conference on Artificial Intelligence*. Springer, Heidelberg, Germany, 137–149.

[5] Raphael Labaca Castro, Corinna Schmitt, and Gabi Dreo Rodosek. 2019. AIMED: Evolving Malware with Genetic Programming to Evade Detection. In *2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, New York, NY, USA, 1–x, Paper presented on Aug 5, 2019.

[6] Raphael Labaca Castro, Corinna Schmitt, and Gabi Dreo Rodosek. 2019. ARMED: How Automatic Malware Modifications Can Evade Static Detection?. In *5th International Conference on Information Management (ICIM)*. IEEE, New York, NY, USA, 20–27.

[7] Razvan Pascanu, Jack W Stokes, Hermineh Sanossian, Mady Marinescu, and Anil Thomas. 2015. Malware classification with recurrent networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, New York, NY, USA, 1916–1920.

[8] Subhojeet Pramanik. 2018. Malware detection using Convolutional Neural Networks. https://github.com/subho406/Malware-detection-using-Convolutional-Neural-Networks.

[9] Shun Tobiyama, Yukiko Yamaguchi, Hajime Shimada, Tomonori Ikuse, and Takeshi Yagi. 2016. Malware detection with deep neural network using process behavior. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2. IEEE, New York, NY, USA, 577–582.

[10] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. 2017. Malware traffic classification using convolutional neural network for representation learning. In *2017 International Conference on Information Networking (ICOIN)*. IEEE, New York, NY, USA, 712–717.