Poster: Exploring Adversarial Examples in Malware Detection

Octavian Suciu
University of Maryland, College Park
osuciu@umiacs.umd.edu

Scott E. Coull
FireEye, Inc.
scott.coull@fireeye.com

Jeffrey Johns
FireEye, Inc.
jeffrey.johns@fireeye.com

***Abstract*—The convolutional neural network (CNN) architecture is increasingly being applied to new domains, such as malware detection, where it is able to learn malicious behavior from raw bytes extracted from executables. These architectures reach impressive performance with no feature engineering effort involved, but their robustness against active attackers is yet to be understood. Such malware detectors could face a new attack vector in the form of adversarial interference with the classification model. Existing evasion attacks intended to cause misclassification on test-time instances, which have been extensively studied for image classifiers, are not applicable because of the input semantics that prevents arbitrary changes to the binaries. This paper explores the area of adversarial examples for malware detection. By training an existing model on a production-scale dataset, we show that some previous attacks are less effective than initially reported, while simultaneously highlighting architectural weaknesses that facilitate new attack strategies for malware classification. Finally, we explore how generalizable different attack strategies are, the trade-offs when aiming to increase their effectiveness, and the transferability of single-step attacks. This poster is associated with an accepted paper from the Deep Learning and Security Workshop at S&P.**

I. Introduction

The popularity of convolutional neural network (CNN) classifiers has lead to their adoption in fields which have been historically adversarial, such as malware detection. Recent advances in adversarial machine learning have highlighted weaknesses of classifiers when faced with adversarial samples. One such class of attacks is evasion, which acts on test-time instances. The instances, also called adversarial examples, are modified by the attacker such that they are misclassified by the victim classifier even though they still resemble their original representation. State-of-the-art attacks focus mainly on image classifiers, where attacks add small perturbations to input pixels that lead to a large shift in the victim classifier feature space, potentially shifting it across the classification decision boundary.

In the context of malware detection, adversarial examples could represent an additional attack vector for an attacker determined to evade such a system. However, domain-specific challenges limit the applicability of existing attacks designed against image classifiers on this task. First, the strict semantics of binary files disallows arbitrary perturbations in the input space. This is because there is a structural interdependence between adjacent bytes, and any change to a byte value could potentially break the functionality of the executable. Second, limited availability of representative datasets or robust public models limits the generality of existing studies.

This paper sheds light on the generalization property of adversarial examples against CNN-based malware detectors by focusing on an existing classifier called MalConv [1]. The architecturea uses an embedding layer which is passed through a gated convolutional layer, followed by a temporal maxpooling layer, before a final fully connected layer.

By training on a production-scale dataset of 12.5 million Portable Executable (PE) files, we are able to observe interesting properties of adversarial attacks, showing that their effectiveness could be misestimated when small datasets are used for training, and that single-step attacks are more effective against robust models trained on larger datasets.

II. Datasets.

To evaluate the success of evasion attacks against the MalConv architecture we utilize three datasets. First, we collected 16.3M PE files from a variety of sources, including VirusTotal, Reversing Labs, and proprietary FireEye data. The data was used to create a production-quality dataset of 12.5M training samples and 3.8M testing samples, which we refer to as the *Full* dataset. Second, we utilize the *EMBER* dataset [2], which is a publicly available dataset comprised of 1.1M PE files, out of which 900K are used for training. On this dataset, we use the pre-trained MalConv model released with the dataset. In addition, we also created a smaller dataset whose size and distribution is more in line with prior attacks proposed by Kolosnjaji et al. [3], which we refer to as the *Mini* dataset. The Mini dataset was created by sampling 4,000 goodware and 4,598 malware samples from the Full dataset.

III. Attack Strategies

We utilize two attack strategies throughout our study, each with its own set of trade-offs.

a) Append Attacks: These attacks address the semantic integrity constraints of PE files by appending adversarial noise to the original file.

Benign Append: The attack takes bytes from the beginning of benign instances and appends them to the end of a malicious instance. The intuition behind this attack is that leading bytes of a file, and especially the PE headers, are the most influential towards the classification decision [1]. Therefore, it signals whether the maliciousness of the target could be offset by appending highly influential benign bytes.

FGM Append: We propose the "one-shot" FGM Append attack, an adaptation of the Fast Gradient Method (FGM) originally described in [4]. Our attack appends random bytes

# Bytes	Benign Append			FGM Append		
	Mini	EMBER	Full	Mini	EMBER	Full
500	4%	0%	0%	1%	13%	13%
2,000	5%	1%	0%	2%	18%	30%
5,000	6%	2%	1%	2%	26%	52%
10,000	9%	2%	1%	1%	33%	71%

TABLE I: Success Rate of the Append attacks for increased number of added bytes.

to the original sample and updates them using a policy dictated by FGM.

b) Slack Attacks: Besides the inability to append bytes to files that already exceed the model's maximum size, they also need to offset a large fraction of the original discriminative features. We therefore propose an attack strategy that exploits the existing bytes of binaries *with no side effects on the functionality of the program.*

Slack FGM: Our strategy defines a set of slack bytes where an attack algorithm is allowed to freely modify bytes in the existing binary without breaking the PE. This strategy extracts the gaps between neighboring PE sections of an executable. The gaps are inserted by the compiler and exist due to misalignments between the virtual addresses and the multipliers over the block sizes on disk. Once identified, the slack bytes are then modified using the FGM approach.

IV. Results

We randomly pick 400 candidate instances from the test set that are correctly classified as malware by the victim and measure the effectiveness of the attacks using the Success Rate (SR): the percentage of adversarial samples that successfully evaded detection.

a) Append Attacks: As shown in Table I, the SR of the Benign Append attack seems to progressively increase with the number of added bytes on the Mini dataset, but fails to show the same behavior on the EMBER and Full datasets. Conversely, in the FGM Append attack we observe that the attack fails on the Mini dataset, while reaching up to 33% SR on EMBER and 71% SR on the Full datasets. This paradoxical behavior highlights the importance of large, robust datasets in evaluating adversarial attacks. One reason for the discrepancy in attack behaviors is that the MalConv model trained using the Mini dataset (modeled after the dataset used by Kolosnjaji et al.) has a severe overfitting problem and the single gradient evaluation does not provide enough information for the sequence of byte changes made in the attack.Aside from the methodological issues surrounding dataset size and composition, our results also show that even a robustly trained MalConv classifier is vulnerable to append attacks when given a sufficiently large degree of freedom.

b) Slack Attacks: In Figure 1 we evaluate the Slack FGM attack by varying the percentage of available slack bytes that are modified. This is achieved by modifying the ϵ parameter of FGM, which in turn affects the magnitude of the changes performed on the bytes. The upper bound for the SR is 15% on EMBER for an attack where 14% (291/2103) slack bytes were modified on average, while on Full we achieve 28% SR for 58% (1117/1930). While the attack is more successful against Full than EMBER, it also succeeds in modifying a proportionally larger number of bytes.

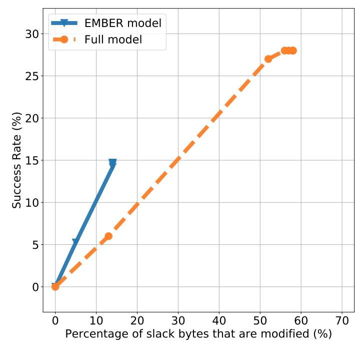

Fig. 1: Slack FGM attack SR for increasing leverage.

Compared to FGM Append which could achieve a higher SR, Slack FGM requires a much smaller number of byte modifications. The results confirm our initial intuition that the coarse nature of MalConv's features requires consideration of the surrounding contextual bytes within the convolutional window. In the slack attack, we make use of existing contextual bytes to amplify the power of our FGM attack without having to generate a full convolutional window using appended bytes.

c) Attack Transferability: We further analyze the transferability of attack samples generated for one (source) model against another (target). We run two experiments with EMBER and Full alternately acting as source and target, and evaluate FGM Append and Slack FGM attacks on samples that successfully evade the source model and for which the original (pre-attack) sample is correctly classified by the target model. At most 2/400 samples evade the target model for each set of experiments, indicating that these *single-step samples are not transferable between models.* The findings are not in line with prior observations on adversarial examples for image classification, where single-step samples were found to successfully transfer across models [5].

V. Conclusion

In this paper, we explored the space of adversarial examples against deep learning-based malware detectors. Our experiments indicate that the effectiveness of adversarial attacks on models trained using small datasets does not always generalize to robust models. The attacks we propose highlight the threat of adversarial examples as an alternative to evasion techniques such as runtime packing.

References

[1] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas, "Malware detection by eating a whole exe," *arXiv preprint arXiv:1710.09435*, 2017.

[2] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," *ArXiv e-prints*, Apr. 2018.

[3] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," *26th European Signal Processing Conference (EUSIPCO '18)*, 2018.

[4] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[5] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.