Poster: Training GANs to Generate Adversarial Examples Against Malware Classification

Raphael Labaca Castro, Corinna Schmitt, and Gabi Dreo Rodosek

Research Institute CODE
Bundeswehr University Munich, Germany
{Raphael.Labaca, Corinna.Schmitt, Gabi.Dreo}@unibw.de

***Abstract*—Detecting new malware using machine learning has been increasingly used lately, yet recent research has proven that deep neural networks report unexpected behavior when confronted with adversarial examples. Implementing Generative Adversarial Network (GAN) has proved to be a powerful technique in the image processing domain and it can be similarly extended to further domains such as malware evasion. While the concept is fairly straight forward for image processing, manipulating portable executable (PE) files can be challenging given its binary nature and the fact that perturbations can render the file corrupt. Hence, most of research proposed in the literature work with limited malware representations and dismissed the actual files. Our hypothesis is that generating valid PE files can be more effective for adversarial learning and the use of machine learning for malware classification. Therefore, we designed an approach using GAN to generate malware adversarial examples by injecting byte-level perturbations, which are able to bypass state-of-the-art classifiers.**

***Index Terms*—Malware, GAN, adversarial learning**

I. Introduction

Generative Adversarial Networks have been extensively studied since introduced by Goodfellow et al. [1] in 2014. The idea is to train two models simultaneously, a $generator$ network that is responsible for generating new examples whereas a $discriminator$ evaluates the probability that these examples are real instead of fake examples coming from the generator. Hence, both have different goals, the generator is looking for an example to bypass the discriminator while the discriminator needs to identify which examples are coming from the generator. Both are optimizing opposite functions in a zero-sum game.

GANs have been observed to successfully perform in a large number of domains but most of the work has been done in image processing including highly realistic representations of objects and humans faces [2]. Recently, implementing GANs to generate and divert malicious attacks has attracted more attention. In DeepDGA [3] the authors attempt to bypass a detector of web domain generation algorithm that identifies human-generated domains from its automatically generated counterparts.

Rigaki et al. [4] proposed to adapt malware communication to force misclassification of new generation Intrusion Prevention Systems. Thus, they adjusted the code of a malware to mimic the network traffic of the Facebook chat application. Their work suggest that GANs can be successful at modifying malware traffic in order to remain undetectable.

Specifically focused in malware is MalGAN [5], which aims to generate malicious software that is misclassified as benign. In this case, the authors implement a surrogate detector, which will fit the given black box model in order to compute the gradient that will be used by the GAN to create the adversarial examples. While the authors report full evasion for almost all cases, the approach requires strong assumptions that are less related with real-case scenarios. For instance, it implies that the adversaries have the ability to fully identify the feature space. Moreover, it focus mostly on API features, which is a very extended way to represent portable executable (PE) files, yet limited to bypass real malware classifiers alone.

Unlike the image domain, adding a random string of perturbations on a feature representation of a PE file is likely to bypass classification, though is not enough to build an adversarial example.

Therefore, we propose a GAN that is able to work with real PE files instead of feature representations based on automatic byte-level modifications [6]. We build on the work of Hu et al. [5] to implement a generator and discriminator, which will then be able to inject and analyze real perturbations on files and deploy adversarial examples.

II. Design

We decided to implement a GAN that is able to inject automatic byte-level perturbations into PE files [6]. To be able to generate adversarial examples, we needed to combine feature representation with real perturbations injected into the malware examples. In order to achieve that, we define the noise, as a vector of nine byte-level perturbations. Both the vector of perturbations as well as the malware file are sent to the generator to create the adversarial examples. Those along with benign examples from the ground-truth set are fed into the black box detector.

Both $generator$ and $discriminator$ are feed-forward neural networks. As depicted in Fig. 1, the generator receives as input a feature representation from a malware file m as well as a vector of byte-level perturbations, $noise$. As output, an adversarial example m' is generated and sent to the black box classifier along with benign samples b. The black box model classifies both m' and b and returns the labels as $d(m', b)$

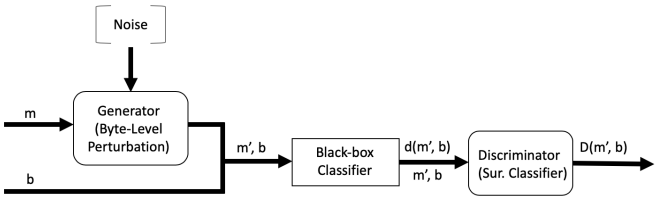

Fig. 1. GAN-based model architecture

along with a feature representation of the malicious and benign files to be used as input to the discriminator. Afterwards, the discriminator acts like a surrogate classifier and returns the probability that the input is malicious, noted as $D(m', b)$.

As feature representations, we implement a vector of 2350 features including metadata from the header, section, and import and export tables as well as file strings. These features provide an extensive overview of each malware sample. Finally, the discriminator will need to fit the black box model in order to reverse-engineer it and create a surrogate model that behaves similarly.

Unlike regular GAN approaches, the generator will create random sequence of perturbation comprising nine different possibilities each injection. Thus, we have a vector of length 10 with nine possibilities each, 9^{10} different combinations of perturbation vectors could be injected to each of the PE files to generate adversarial examples.

On the other hand, the discriminator will need to fit a black box detector. In our case, in order to experience the behavior of real models, we implement [7] a previously-trained GBDT, which has state-of-the-art detection rates. Hence, the capabilities of the surrogate will extend to creating new adversarial examples that will likely to not only bypass the black box detector but also cross-evade different classifiers.

III. Solution

We propose a modified version of a malware analysis environment, which is able to return valid PE binaries as adversarial examples after all perturbations were injected [6]. In order to generate PE files, we adjusted our generator to be able to perform valid byte-level perturbations to the malware sample before sending the output to the black box classifier.

To train the GAN both malware and benign software are needed. Every malware file is represented as a 2350-feature vector and then fed to the generator. In order to optimize our approach, the following steps need to be taken following the architecture:

(1) Define feed-forward neural network topology based on the examples, features needed and complexity of the task. Many nodes will lead to a more complex network that probably overfits, yet too few can lead to errors since the complexity is too high for a small number of nodes. Regarding the number of layers perhaps some of the best pieces of advice on this matter comes from Yoshua Bengio, where he points out the importance of keep adding layers until the test error does not improve anymore [8]. (2) The malware detector is a Gradient Boosted Decision Tree (GBDT), which acts as a black box classifier with reported $ROC - AUC = 0.993$ [7]. While training a multilayer perceptron in a binary vector resembling API calls as black box can deploy higher evasion rates, we speculate important aspects from malware behavior are overlooked and therefore choose a rich feature representation. (3) Fit the black box classifier into a feed-forward neural network in order to be able to update the weights of the surrogate model and perform gradient descent. (4) Similarly, update the generator weights with the gradient information of the surrogate classifier.

IV. Conclusion

Recently, generative adversarial networks are shaping the way adversarial examples are created in a plethora of applications. Malware adversaries can leverage this technology to improve malware generation despite of the classifier used in a real scenario.

Here we present a GAN-based approach, which aims to generate real malware mutations applying GAN algorithms to previous malicious software in order to bypass static malware classifiers.

While evasion rates are almost perfect for simple cases, preliminary results showed that more complex scenarios, which resemble the real world more closely, are feasible. We observed that byte-level perturbations can be combined with the power of GANs in order to extend the attack vector of previously detected PE malware.

Acknowledgment

The authors would like to thank the Chair for Communication Systems and Network Security and the research institute CODE for their comments and improvements.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[2] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *arXiv preprint arXiv:1812.04948*, 2018.

[3] H. S. Anderson, J. Woodbridge, and B. Filar, "Deepdga: Adversarially-tuned domain generation and detection," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*. ACM, 2016, pp. 13–21.

[4] M. Rigaki and S. Garcia, "Bringing a gan to a knife-fight: Adapting malware communication to avoid detection," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 70–75.

[5] W. Hu and Y. Tan, "Generating Adversarial Malware Examples for Black-box Attacks based on GAN," *Computing Research Repository*, February 20, 2017.

[6] R. Labaca Castro and G. D. Rodosek, "ARMED: How Automatic Malware Modifications Can Evade Static Detection?" in *5th International Conference on Information Management (ICIM)*. New York, NY, USA: IEEE, 02 2019, pp. 1–x, Acceptance notification on February 22, 2019.

[7] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth, "Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning," *Computing Research Repository*, pp. 1–9, January 30, 2018.

[8] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.