AdvML@KDD'19 Workshop Challenge Problem: **Robust Malware Detection**

Overview

The bulk of adversarial machine learning research has been focused on crafting attacks and defenses for image classification. In this challenge, we put adversarial machine learning in the context of robust malware detection. In the era of modern cyber warfare, cyber adversaries craft adversarial malicious code that can evade malware detectors [4]. The problem of crafting adversarial examples in the malware classification setup is more challenging than image classifications: malware adversarial examples must not only fool the classifier, they must also ensure that their adversarial perturbations do not alter the malicious payload.

The gist for this challenge is to *defend* against adversarial attacks by building robust detectors and/or *attack* robust malware detectors based on binary indicators of imported functions used by the malware. The challenge has two tracks:

1. **Defend** Track: Build high-accuracy deep models that are robust to adversarial attacks.
2. **Attack** Track: Craft adversarial malicious PEs that evades detection on adversarially trained models [1].

A toolkit for both tracks can be found at `https://github.com/ALFA-group/malware_challenge`.

Setup

Dataset. The dataset is composed of Portable Executable (PE) files, a file format for executables in 32-bit and 64-bit Windows operating systems. The PE format encapsulates information necessary for Windows OS to manage the wrapped code, and this information can be extracted in order to construct feature representations. PE files are a natural choice due to their structure and widespread use as malware. In this challenge, we chose binary indicator vectors of the PE's imported functions to be the input of a malware detector. I.e., each PE is represented as a binary indicator vector, where each entry corresponds to a unique Windows API call. The value "1" in an entry denotes the presence of the corresponding API call. In our dataset, we observe a total of $22,761$ unique API calls, thus each PE file is represented as a binary vector $\mathbf{x} \in \mathcal{X} = \{0,1\}^{22761}$. We used the LIEF[1] library to parse each PE.[2] In line with the challenge's two tracks, we release two datasets, namley *defend.zip* and *attack.zip*, respectively. Represented by their binary indicator vectors, the *defend* dataset contains $15,200$ bengin PEs and $15,200$ malicious PEs. The *attack* dataset contains $3,800$ malicious PEs.

[1] `https://lief.quarkslab.com/`

[2] Note that other established parsing tools can be used (e.g., [2]) and a myriad of features can be used to represent PEs.

Adversarial Perturbations. To craft a functionality-preserving adversarial version $\mathbf{x}_{adv}$ of a malicious PE $\mathbf{x}$, the following condition must be satisfied.

$$\mathbf{x} \wedge \mathbf{x}_{adv} = \mathbf{x} \,.$$

That is, perturbations that preserve malicious functionality correspond to setting unset bits in the binary feature vector $\mathbf{x}$ of the malware at hand. Acting as an attacker/adversary, you may only import additional functions that are not used by the PE and you cannot remove any existing ones. Note that we consider adversarial perturbations only for *malicious* PEs. The assumption here is that authors of benign applications have no interest in having their binaries misclassified as malware.

Model Architecture. To build a robust malware detector, participants can use any deep model architecture they see fit. However, the model's input layer should be of $22,761$ coordinates (this corresponds to the dimensionality of PEs indicator vectors) and the output layer should be a two-neuron `nn.LogSoftmax(dim=1)` layer.[3].

Track 1: Defend Challenge

Participants in the defend track are required to construct robust models given the *defend* dataset. The challenge's code repository demonstrates hardening malware detectors with *adversarial training* [1]. Participants must submit their trained model's **.pt** file.[4] The trained model can be saved as shown in Listing 1.

```
torch.save(the_model, PATH)
```

Listing 1: Saving a model

For evaluation, we will assess the model's performance on a test set of benign and malicious (and adversarial versions of them) PEs. Participants' solutions will be evaluated based on their F1 score against their strongest adversaries as shown in Listing 2

```
model = torch.load(PATH)
holdout_set = load_holdout_set()
f1_score = eval(model, holdout_set)
```

Listing 2: Evaluating a model

Track 2: Attack Challenge

Given the *attack* dataset—which consists of 3800 malicious PEs, participants are required to craft adversarial versions that will evade detection by a secret model. That is, the participants are asked to craft *black-box, zero-query, transfer-only* attacks. To this end, we will release the secret model's architecture but retain its weights. Further, the participants will be provided with a naturally pre-trained version of the model. Participants could use the model in addition to any other models they may develop to craft adversarial versions of the 3800 malicious PEs. The challenge's code repository demonstrates the evasion rate of functionality-preserving perturbations

[3] E.g., see the `build_ff_classifier` function at `https://bit.ly/2GfHTlW`

[4] See `https://bit.ly/2D7byMe`

on a naturally trained model. Participants must submit their 3800 crafted adversarial examples in the form of an `.npy` file (stored as an 3800×22761 Numpy array), where the ith row of the array corresponds to the ith malicious PE of the *attack* dataset. Note that the entries of the array should be either 0 or 1. Participants' solutions will be evaluated based on their evasion rate on the secret model.

Submission

For the *defend* track, participants should submit their trained models to the email below. For the *attack* track, participants should submit their crafted adversarial malware versions to the same email. If files are too big, participants may share a download link.

Important Dates.

- Release of Defense Challenge Data/Code: 17 April 2019
- Challenge Submission Deadline: 15 July 2019

The winners of the challenge will be announced at the workshop and will receive cash prizes sponsored by the MIT-IBM Watson AI Lab. The winners will also be invited to submit a technical report/poster about their techniques. For paper submission to the workshop (deadline: **5 May 2019**), please refer to the workshop's website: `https://sites.google.com/view/advml`.

More Details/Suggestions. A series of related blog posts can be found at `https://bit.ly/2KNe0Oj`. If you have any questions about the challenge, please send an email to advmalwarechallenge@gmail.com

Acknowledgment. The outline of this challenge is inspired by the AICS 2019 workshop challenge problem and built on the work in [1, 3]. This work was supported by the MIT-IBM Watson AI Lab and CSAIL CyberSecurity Initiative.

References

[1] Abdullah Al-Dujaili, Alex Huang, Erik Hemberg, and Una-May O'Reilly. Adversarial deep learning for robust detection of binary encoded malware. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 76–82. IEEE, 2018.

[2] Ero Carrera. pefile: a python module to read and work with pe (portable executable) files. https://github.com/erocarrera/pefile, 2018.

[3] Alex Huang, Abdullah Al-Dujaili, Erik Hemberg, and Una-May O'Reilly. On visual hallmarks of robustness to adversarial malware. *arXiv preprint arXiv:1805.03553*, 2018.

[4] Weilin Xu, Yanjun Qi, and David Evans. Automatically evading classifiers. In *Proceedings of the 2016 network and distributed systems symposium*, pages 21–24, 2016.