Adversarial Malware Generation Using GANs

Constructing "Sheep's Clothing" for Malware

Zayd Hammoudeh

Department of Computer Science, University of Oregon

zayd@cs.uoregon.edu

Background

- **Motivation**: Security companies are increasingly relying on *brittle* machine learning-based techniques for malware detection
- Malware detection is binary classification task over Boolean feature vectors [3, 4]
- Each Boolean feature represents whether a particular API/system call/DLL is used
- Malware detectors lack robustness in particular when a malicious actor constructs *adversarial examples* designed to avoid detection
- Adversarial examples are impossible to construct manually in particular for high dimensional feature spaces

Problem Setting

- **Goal**: Develop an automated system to adversarially generate malware that appear benign to a blackbox malware detector
- **Input**: An existing malware program already flagged malicious by the detector
- **Output**: A "benign-looking" adversarially transformed malware sample that *preserves all of the original malware's functionality*

Our Solution

Adversarially trained generator $G(\mathbf{m},\mathbf{z})$ constructed from a neural network with two hidden layers

- $\mathbf{m}$: A malware program's Boolean feature vector
- $\mathbf{z}$: Latent vector of Z i.i.d. random variables from distribution $\mathcal{U}(0,1)$, where Z is a positive-integer hyperparameter

The adversarially generated malware output vector, $\mathbf{m}'$, defined as:

$$\mathbf{m}' = \text{sgn}\Big\{\max\big\{G(\mathbf{m},\mathbf{z}),\mathbf{m}\big\} - 0.5\Big\}$$

Role of Element-wise Operations in Definition of $\mathbf{m}'$:

- max : Diffentiable bitwise-OR that preserves the original malware's functionality by ensuring all system calls remain enabled
- sgn : Binarizes output vector

GAN Training Architecture

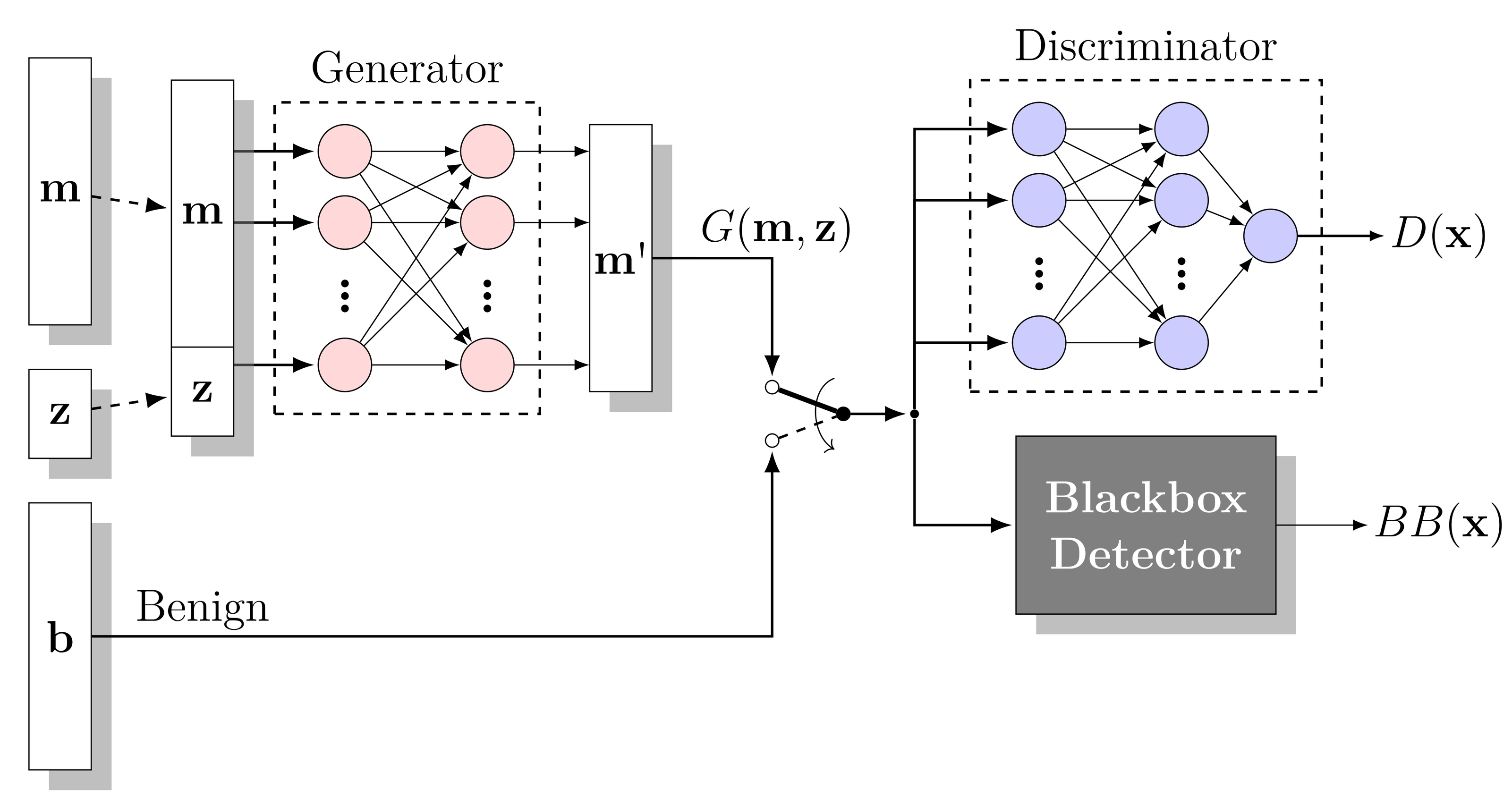

Figure 1: Adversarial Malware Generator Training Architecture

Generative adversarial network (GAN)-based architecture inspired by [2]

Three Primary Components:

- **Blackbox Detector** (BB): Emulates a static third-party malware detector
- **Generator** (G): Neural network that learns to *adversarially generate* malware programs that appear benign to the blackbox detector
- **Discriminator** (D): Neural network that learns to mimic the behavior of the blackbox detector to assist training the generator

Generator Loss Function:

$$L_G = \mathbb{E}\Big\{\ln D(G(\mathbf{m},\mathbf{z}))\Big\}$$

Discriminator Loss Function:

$$L_D = -\mathbb{E}_{BB(\mathbf{x})=\text{Malware}}\Big\{\ln D(\mathbf{x})\Big\} - \mathbb{E}_{BB(\mathbf{x})=\text{Benign}}\Big\{\ln\left(1 - D(\mathbf{x})\right)\Big\}$$

Implementation

- : https://github.com/ZaydH/MalwareGAN
- Fully open-source implementation in PyTorch
- Full **CUDA** support for >30× training and testing

Experimental Results

Dataset:

- SLEIPNIR dataset [1] of 54,620 programs (∼65% malware)
- # Boolean Features per Program: 22,761
- Test Set Holdout: 10% of malware & benign programs

Modeling the Blackbox Detector:

- Utilize off-the-shelf tools based on both parametric and non-parametric binary classification algorithms
- Random forests create highly non-linear decision boundaries, which can be very difficult to mimic with relatively limited training data

Summary of Results in Table 1:

- **Primary Takeaway**: *Our adversarially generated malware looks more benign than actual benign programs*
- SLEIPNIR dataset is very challenging as blackbox detector has a *high false positive rate* on benign test set
- Blackbox detector correctly classifies >93% of *original* malware
- Only 0–3% of *adversarially generated* malware was detectable irrespective of the blackbox classification algorithm

Blackbox Algorithm	Benign	Malware	
		Orig.	Adver. Gen.
Logistic Regression	10.9	94.4	**0**
Multilayer Perceptron	9.2	93.5	**0**
Decision Tree	11.2	94.2	**0.01**
Random Forest	9.7	94.6	**2.6**

Table 1: Percentage of Each Test Set Classified as Malicious

References

[1] Abdullah Al-Dujaili et al. "Adversarial Deep Learning for Robust Detection of Binary Encoded Malware". In: *arXiv preprint arXiv:1801.02950* (2018).

[2] Weiwei Hu and Ying Tan. "Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN". In: *arXiv Preprint* abs/1702.05983 (2017).

[3] J. Zico Kolter and Marcus A. Maloof. "Learning to Detect and Classify Malicious Executables in the Wild". In: *Journal of Machine Learning Research* (2006).

[4] Matthew G. Schultz et al. "Data Mining Methods for Detection of New Malicious Executables". In: *Proceedings of Symposium on Security & Privacy*. 2001.