Mal-D2GAN: Double-Detector based GAN for Malware Generation

Nam Hoang Thanh
Academy of Cryptography Techniques
Hanoi, Vietnam
hoangthanhnam@actvn.edu.vn

Trung Pham Duy
Academy of Cryptography Techniques
Hanoi, Vietnam
trungpd@actvn.edu.vn

Lam Bui Thu*
Academy of Cryptography Techniques
Hanoi, Vietnam
lambt@actvn.edu.vn

Abstract—Machine learning (ML) has been developed to detect malware in recent years. Most researchers focused their efforts on improving the detection performance but ignored the robustness of the ML models. In addition, many machine learning algorithms are very vulnerable to intentional attacks. To solve these problems, adversarial malware examples are generated by GANs to enhance the robustness of the malware detector. However, since current GAN models suffer from limitations such as unstable training and weak adversarial examples, we propose the Mal-D2GAN model to address these problems. Specifically, the Mal-D2GAN architecture was designed with double-detector and a least square loss function and tested on a dataset of 20,000 samples. The results show that the Mal-D2GAN model reduced the detection accuracy (true positive rate) in 8 malware detectors. The performance was then compared with that of the existing MalGAN and Mal- LSGAN models.

Keywords– malware detection, adversarial malware examples, generative adversarial network

I. INTRODUCTION

Malware, or malicious software, refers to programs or codes intentionally designed to cause harm to personal computers, servers, or computer networks. The primary purpose of malware is to execute unauthorized actions that can lead to various illicit activities. Therefore, malware detection is an important step in the incident response process for any organization, system, or enterprise. Malware detection is divided into two main types: signature-based detection and behavior-based detection [1]

Signature-based detection is a technique that compares files or network traffic with a database of known malware signatures. A signature is a unique pattern or fingerprint of bytes that identifies a specific malware. Signature-based detection can quickly and accurately detect known malware variants, but it can not detect new or unknown malware with no signature in the database. In contrast, behavior-based detection can detect new or unknown malware with no signature and malware that tries to hide or disguise its signature. Behavior-based detection is a technique that analyzes the actions or behavior of files or network traffic to detect malware. Behavior-based detection does not rely on predefined signatures, but on heuristic rules or machine- learning algorithms that can identify suspicious or malicious activities [2].

Lam Bui Thu is the corresponding author.

Machine learning [3] can learn from the behavior of software, not just rely on pre-known signatures. This allows the detection of new variants of malware without frequent updates to signatures. However, machine learning-based malware detection systems face the drawback of susceptibility to adversarial malware samples [4]. Adversarial samples generated from Generative Adversarial Networks (GANs) [5] have led to numerous adversarial attacks on machine learning-based malware detection systems [6]. To create adversarial malware examples to bypass ML based malware detector, Hu et al [7] proposed the MalGAN model which uses adversarial samples to bypass malware detectors (a.k.a black-box detectors). The purpose of this work to enhance the robustness of malware detector. In studies such as Mal-LSGAN [8] and LSGAN- AT [9], improvements using integrated activation functions were introduced to enhance the ability to bypass black-box detectors. However, recent GAN models suffer from limitations such as unstable training and weak adversarial examples. In this paper, we propose the Mal-D2GAN model with double-detector and a least square loss function and test on a dataset of 20,000 samples. The experimental results show that adversarial examples generated by our proposed model can bypass malware black-box detectors with 8 common machine learning algorithms. The proposed method's results are higher than the MalGAN and Mal- LSGAN models proposed.

The rest of the paper is organized as follows. We first review the related work in Section 2. Section 3 presents the proposed double detectors-based GAN for malware generation. Section 4 describes the experiment and results. We conclude the paper with a conclusion and indicate possible research directions in Section 5.

II. RELATED WORKS

Hu et al [7] introduced an algorithm based on GANs called MalGAN, which creates adversarial malware examples that can evade black-box machine-learning models. MalGAN uses a substitute detector to fit the black- box malware detection, and it is trained to minimize the detection probability of adversarial malware samples predicted by the substitute detector. MalGAN's advantage over traditional gradient-based adversarial sample generation algorithms is its ability to reduce the detection rate to nearly zero, making retraining-based defenses against adversarial samples less effective.

In the proposed Mal-LSGAN model to tackle these weaknesses. By using a least square (LS) loss function [10] and new activation function combinations, Mal-LSGAN achieves a higher Attack Success Rate (ASR) and a lower True Positive Rate (TPR) in 6 ML detectors [8].

In the proposed AAGAN [11], an automated Android malware generation system based on Generative Adversarial Networks (GAN) that can successfully deceive current ML detectors. Their experiment results indicate that adversarial examples generated by their system can flip the prediction of the state-of-the-art detection algorithms in 99% of cases using a real-world dataset. To defend against AE attacks, they improve the robustness of our detection system by alternatively retraining with these newly generated AEs. Surprisingly, after retraining five times, AAGAN can achieve an 89% success rate in bypassing our malware detection system [12].

Shahpasand el al [13] show a model for generating highly effective adversarial samples against machine learning-based malware detectors. The gap covered by this contribution is the threshold we provide on the maximum number of changes to an original malware sample. They used the Drebin dataset, choosing a subset of features that have the highest influence on classification with the same performance as the complete feature set. Their result shows that having a high performance for classifier accuracy on the original samples, the proposed model for generating adversarial samples can attack the classifiers with up to a 99% success rate but only shows that the model can attack four different types of classifiers with a high chance of evasion.

III. THE DOUBLE-DETECTORS BASED GAN FOR MALWARE GENERATION

The proposed model includes 4 main blocks: generator, black-box detector, substitute detector and additional detector.

Fig. 1. The architecture of Mal-D2GAN

A. Black-boxdetector

This external system utilizes machine learning-based algorithms for malware detection. We assume that the malware author is only aware of the types of features used by the black-box detector. They do not know which machine learning algorithm is employed, nor do they have access

to the trained model parameters. The malware author can receive the detection results of their programs from the black-box detector. In this model, the black-box detectors uses 8 classification algorithms such as Random forest (RF), Linear Regression (LR), Decision Tree (DT), Multilayer Perceptron (MLP), Support Vector Machine (SVM), AdaBoost (AB), GradientBoosting (GB), and K-nearest- Neighbor (KNN).

B. Generator

This block generates adversarial malware data by transforming a malware feature vector into its adversarial sample. It requires a combination of the malware feature vector $\mathbf{m}$ and a noise vector $\mathbf{z}$ as input. Here, we use the design advantages of the DCGAN [14] model for the generator block. The convolution layers in DCGAN are replaced with fully connected layers. Malware is an M-dimensional binary vector, with each element representing the presence or absence of a feature. $\mathbf{z}$ is a Z-dimensional vector with Z being a hyperparameter. Each element of $\mathbf{z}$ is a random number sampled from a uniform distribution in the range $[0, 1)$. The purpose of $\mathbf{z}$ is to allow the generator to create diverse adversarial samples from a single malware feature vector. The structure of the generator is shown in Table I.

TABLE I
THE GENERATOR (G) NETWORK STRUCTURE

Layer Type	Output Shape
Input Layer (Malware)	(None, 160)
Input Layer (Noise)	(None, 10)
Concatenate (Malware + Noise)	(None, 170)
LeakyReLU (Dense)	(None, 256)
BatchNormalization	(None, 256)
LeakyReLU (Dense)	(None, 256)
BatchNormalization	(None, 256)
LeakyReLU (Dense)	(None, 256)
BatchNormalization	(None, 256)
Sigmoid (Dense)	(None, 160)
Maximum	(None, 160)

The input vector is fed into a multi-layer feed-forward neural network with weights θ_g. The output layer of this network has M neurons, and the activation function used by the last layer is sigmoid, which limits the output to the range $(0, 1)$. The hidden layers use LeakyReLU as the activation function due to its better performance and high learning cost parameters. The loss function used in the generator block is mean square error (MSE). The output of this generator network is denoted as o.

Since the malware feature values are binary, a binary conversion process is applied to o depending on whether the element values are greater than 0.5 or not, creating a binary vector o'. When creating adversarial samples for binary malware features, we exclusively focus on adding unrelated features to the original malware. The removal of any feature from the original malware could potentially disrupt its functionality. For example, if the API "DeleteFile" is removed from a program, the program will not be able to perform normal deleting functions, and the malware may be detected.

The non-zero elements of the binary vector o' are used as unrelated features to add to the original malware. The final adversarial example can be represented as $m' = m \mid o'$, where "|" is the element-wise binary OR operation. m' is a binary vector, so the gradient cannot propagate back from the substitute detector to the generator. A smooth function G is defined to receive gradient information from the substitute detector, as shown in:

$$G_{\theta_g}(m, z) \;= max_(m, o) \tag{1}$$

C. Detectors

1) Substitute Detector: Substitute detector is employed to approximate its behavior and supply gradient information for training the generator. Previously in , when there was no additional detector, substitute detector was a multi-layer feed-forward neural network with weights θ_d that received the program feature vector $\mathbf{m}$ as input. It classifies the program as either benign or malicious. We denote the probability of $\mathbf{x}$ being predicted as malware as $D_{\theta_d}(x)$.

The substitute detector network plays a crucial role in enabling gradient-based adversarial training by serving as a differentiable approximation of the actual detection system.

TABLE II
THE NETWORK STRUCTURE OF THE SUBSTITUTE DETECTOR

Layer Type	Output Shape
Input Layer (Malware)	(None, 160)
Sigmoid (Dense)	(None, 256)
Sigmoid (Dense)	(None, 1)

The training data for the substitute detector includes adversarial malware samples generated by the generator and benign samples from a benign dataset collected from various sources. Importantly, the training data for the substitute detector is all labeled by the black-box detector. The actual labels of the training data are not used to train the substitute detector. The goal of the substitute detector is to fit the black-box detector.

The black-box detector will first detect this training data and label it as either malicious or benign. Then, these samples are used to train the substitute detector.

2) Additional Detector: In the new Mal-D2GAN model, an additional detector block is designed to work alongside the substitute detector to more effectively detect malware generated by the generator. Experimental results have demonstrated this effectiveness.

TABLE III
NETWORK STRUCTURE OF THE SUBSTITUTE DETECTOR

Layer Type	Output Shape
Input Layer (Malware)	(None, 160)
Sigmoid (Dense)	(None, 256)
Sigmoid (Dense)	(None, 1)

The data used to train additional detector will include benign data from the training set as well as data generated directly from the generator block. The input for both substitute detector and additional detector is malware represented as an M-dimensional vector. Substitute detector and additional detector will be designed to work together to predict whether the received data is malicious or benign.

3) Adversarial example generation: To train MalGAN, the malware author should first collect two datasets: one containing malicious code samples and another containing benign samples.

The combined loss function of substitute detector (L_{D_1}) and additional detector (L_{D_2}) is defined as in:

$$L_{D_1} = \frac{1}{2}\mathbb{E}_{x \in BB_{Benign}}\left[\left(D_{1\theta_{d_1}}(x)\right)^2\right]$$
$$+ \frac{1}{2}\mathbb{E}_{x \in BB_{Malware}}\left[\left(D_{1\theta_{d_1}}(x) - 1\right)^2\right] \tag{2}$$

$$L_{D_2} = \frac{1}{2}\mathbb{E}_{x \in S_{Benign}}\left[\left(D_{2\theta_{d_2}}(x)\right)^2\right]$$
$$+ \frac{1}{2}\mathbb{E}_{x \in G_{Adversarial}}\left[\left(D_{2\theta_{d_2}}(x) - 1\right)^2\right] \tag{3}$$

$$L_D = \alpha L_{D_1} + (1 - \alpha)L_{D_2} \tag{4}$$

BB_{Benign} represents a set of programs identified as non-malicious by the black-box detector, and $BB_{Malware}$ represents a set of programs detected as malware by the black-box detector. S_{Benign} is the original dataset of benign samples labeled to train the additional detector, and $G_{Adversarial}$ refers to malware data generated by the generator for training the additional detector. To train the detectors, L_D needs to be minimized with respect to the weights of the detectors. In (3), α is used to represent the importance of L_{D_1} and L_{D_2}.

The loss function of the generator is defined as:

$$L_G = \frac{1}{2}\,\mathbb{E}_{m \in S_{\text{Malware}},\ z \sim \text{p}(0,1)}\left[\left(D_{\theta_{d_1 d_2}}\left(G_{\theta_g}(m, z)\right)\right)^2\right] \tag{5}$$

Algorithm 1 The Training Process of Mal-D2GAN

1: **while** not converging **do**
2:     Sample a batch of malware M
3:     Generate adversarial samples M' from Generator for M
4:     Sample a batch of benign samples B
5:     Label M' and B using black-box detector
6:     Label S_{Benign} and $G_{Adversarial}$ using additional detector
7:     Update weights θ_{d1} and θ_{d2} of substitute detector and additional detector by descending along the gradients $\nabla_{\theta_1} L_{D1}$, $\nabla_{\theta_2} L_{D2}$
8:     Update weights θ_g of generator by descending along the gradients $\nabla_\theta L_G$
9: **end while**

$S_{Malware}$ represents a dataset of actual malware and is not labeled by the black-box detector. L_G is minimized with respect to the weights of the generator during training. Minimizing L_G reduces the probability of the generated malware

being predicted as harmful, thereby encouraging the substitute detector to classify malware as benign. Because the substitute detector aims to align with the black-box detector, training the generator will further deceive the black-box detector. The entire training process of Mal-D2GAN is illustrated in Algorithm 1.

IV. EXPERIMENTS AND RESULTS

A. Experimental Setup

The malware dataset used in the experiment was collected from various sources such as VirusShare [15], consisting of files in Portable Executable (PE) format. Each malware file, after being processed, is represented as a 160-dimensional API feature vector.

The dataset contains 20,000 samples, of which 70% are malware files and 30% are benign files. In the experiment, the value of α was selected as 0.5.

To build the malware dataset, a behavior-based malware analysis system called Cuckoo Sandbox [16] is required. This system helps to analyze malware samples and provides results in a JSON file format.

Fig. 2. Collecting the report file from the Cuckoo sandbox

To build the malware dataset, a behavior-based malware analysis system called Cuckoo Sandbox [16] is required. This system helps to analyze malware samples and provides results in a JSON file format.

There are many features, including static features and malicious features, but for the experiment, we will select about 160 prioritized features. We use the Random Forest classifier algorithm to determine the scores of the 160 dynamic features.

The testing process of the Mal-D2GAN model uses a method for data splitting as follows: 80% of the dataset is used as the training set, and 20% of the dataset is used as the test set. The black-box detector block will use the training set data for training.

We conducted experiments on a computer with an Intel Core i5-12500H processor, 16GB of RAM, running Windows 11 and Anaconda [17], using 20 epochs fortraining and 5 epochs for retraining [18].

B. Experimental Results

We will analyze the scenario where Mal-D2GAN and black-box detector use the same training set. In malware detection, the true positive rate (TPR) [19] represents the rate of correctly detecting malware. After conducting adversarial attacks, the generated adversarial malware samples gain the ability to bypass detection algorithms. In the table below, a dataset with 20,000 samples is used to test this case [20].

TABLE IV
TRUE POSITIVE RATE (IN PERCENTAGE) ON ORIGINAL SAMPLES AND ADVERSARIAL EXAMPLES MAL-D2GAN, MAL-LSGAN, MALGAN- ON TRAINING SET. "ADVER." REPRESENTS ADVERSARIAL EXAMPLES

Classifier	Mal-D2GAN		Mal-LSGAN		MalGAN	
	Original	Adver	Original	Adver	Original	Adver
RF	97.25	2.28	97.75	20.14	97.02	31.68
LR	94.69	0.00	94.52	20.39	94.50	0.61
DT	97.35	0.00	94.64	0.98	97.40	6.60
SVM	96.10	0.07	96.26	20.61	96.34	5.84
MLP	98.31	0.16	97.80	20.93	97.11	35.09
AB	93.66	0.00	93.86	14.42	94.01	16.02
GB	97.35	0.37	97.35	14.38	97.40	49.86
KNN	98.10	0.15	98.15	1.19	98.81	63.34

TABLE V
TRUE POSITIVE RATE (IN PERCENTAGE) ON ORIGINAL SAMPLES AND ADVERSARIAL EXAMPLES WHEN MAL-D2GAN, MAL-LSGAN AND MALGAN ON THE TEST SET. "ADVER." REPRESENTS ADVERSARIAL EXAMPLES

Classifier	Mal-D2GAN		Mal-LSGAN		MalGAN	
	Original	Adver	Original	Adver	Original	Adver
RF	94.74	2.38	98.16	20.59	97.10	31.80
LR	94.32	0.10	94.17	21.03	94.90	0.43
DT	94.45	0.10	93.80	1.20	97.36	6.33
SVM	95.82	0.00	95.78	19.71	96.22	5.20
MLP	96.48	0.18	96.22	20.33	97.54	37.15
AB	92.04	0.00	92.89	13.99	93.51	15.93
GB	96.11	0.18	96.00	14.07	97.06	51.33
KNN	97.50	0.16	97.80	0.95	97.44	61.31

Overall, the results in Table IV and Table V show that the Mal-D2GAN model reduced the detection accuracy (true positive rate) across 8 malware detection algorithms. The experimental results of the Mal-D2GAN model, along with comparisons with existing models such as Mal-LSGAN and MalGAN, demonstrate its effectiveness. It is evident that our Mal-D2GAN model surpasses the malware detection benchmarks with high performance metrics. Across various algorithms such as RF, LR, DT, SVM, MLP, AB, GB, and KNN, the true positive rate (TPR) mostly decreases to near 0%. In comparison, for Mal-LSGAN, only DT approaches 0%, while for MalGAN, only LR decreases to nearly 0%. Specifically, the MLP classifier in the Mal-D2GAN model shows a dramatic drop in TPR from 98.31% to 0.16%. RF

also decreases significantly to 2.28%, and DT drops rapidly from 97.35% to 0%.

These experimental results demonstrated the impact of the additional detector in the Mal-D2GAN model, which dramatically reduces the detection accuracy (true positive rate) in all 8 malware detectors.

C. Retraining the Black-box Detector

In this section, we will analyze the performance of Mal-D2GAN under the retraining-based defensive [6] approach. If an anti-malware vendor collects enough adversarial malware examples, they can retrain the black-box detector on these adversarial examples to learn their signature and detect them.

We use RF, DT, AB, GB, and KNN as the black-box detectors due to their good performance. After retraining the black-box detector 5 times, it can detect all adversarial examples, as shown in the middle column, namely "Before retraining GAN model" [21].

TABLE VI
TRUE POSITIVE RATE (IN PERCENTAGE) ON THE ADVERSARIAL EXAMPLES
AFTER THE BLACK-BOX DETECTOR IS RETRAINED

Classifier	Mal-D2GAN		Mal-LSGAN		MalGAN	
	Before	After	Before	After	Before	After
RF	100	0.29	100	100	100	16.96
DT	100	0.88	100	20.07	100	36.90
AB	100	0.00	100	14.37	100	20.72
GB	100	0.00	100	100	100	43.61
KNN	100	14.66	100	11.14	100	87.29

However, once antivirus vendors release the updated black-box detector publicly, malware authors will have access to it and can retrain Mal-D2GAN to target the new black-box detector. Following retraining, Mal-D2GAN generates new adversarial examples that remain undetected by RF, DT, AB, GB, and KNN, as indicated in the "After retraining model."

Table VI shows the results of Mal-LSGAN after undergoing 5 rounds of retraining. It demonstrates that Mal-LSGAN performs well with DT, AB, and KNN. Meanwhile, the performance outcomes of MalGAN also plateaued at average effectiveness levels after the model retraining process.

V. CONCLUSION AND FUTURE WORK

In this paper, we develop a novel Mal-D2GAN model to enhance the robustness of the malware detector. The Mal-D2GAN architecture was designed with a double-detector and a least square loss function and tested on a dataset of 20,000 samples. The results show that the Mal-D2GAN model reduced the detection accuracy (true positive rate) in 8 malware detectors. This performance was then compared with that of the existing MalGAN and Mal-LSGAN models. The results also show that after retraining Mal-D2GAN, new adversarial examples remain undetected.

Additionally, detailed experiments demonstrate that combining the two detectors enhances the quality of adversarial sample generation. Moving forward, our goal is to further experiment with retraining using algorithms such as MLP, LR,

and SVM, focusing on generating high-quality malware and testing it across various datasets.

REFERENCES

[1] A. Ray, *Cybersecurity for Connected Medical Devices*. Elsevier Inc, 2022.
[2] M. Goyal and R. Kumar, "The pipeline process of signature-based and behavior-based malware detection," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, India, 2020.
[3] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.
[4] X. Ling *et al.*, "Adversarial attacks against windows pe malware detection: A survey of the state-of-the-art," https://arxiv.org/abs/2112.12310v5, 2023, arXiv:2112.12310v5 [cs.CR], 17 Feb 2023.
[5] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems (NeurIPS)*, vol. 2, 2014.
[6] O. Ibitoye *et al.*, "The threat of adversarial attacks against machine learning," https://arxiv.org/abs/1911.02621v3, 2023, arXiv:1911.02621v3 [cs.CR], 21 Mar 2023.
[7] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on gan," in *Data Mining and Big Data: 7th International Conference (DMBD 2022)*, 2022, pp. 409–423.
[8] J. Wang, X. Chang, J. Mišić, V. B. Mišić, Y. Wang, and J. Zhang, "Mal-lsgan: An effective adversarial malware example generation model," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.
[9] J. Wang, X. Chang, Y. Wang, R. J. Rodríguez, and J. Zhang, "LSGAN-AT: enhancing malware detector robustness against adversarial examples," *Cybersecurity*, vol. 4, no. 1, p. 38, 2021. [Online]. Available: https://doi.org/10.1186/s42400-021-00102-9
[10] X. Mao *et al.*, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2794–2802.
[11] V. H. Trung *et al.*, "Aagan: Android malware generation system based on generative adversarial network," *Vietnam Journal of Computer Science*, vol. 11, no. 2, pp. 275–299, 2024.
[12] A. Creswell *et al.*, "Generative adversarial networks: An overview," https://arxiv.org/abs/1710.07035, 2017, arXiv:1710.07035v1 [cs.CV], 19 Oct 2017.
[13] M. Shahpasand *et al.*, "Adversarial attacks on mobile malware detection," in *2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile)*, February 2019, pp. 24–24.
[14] A. Radford *et al.*, "Unsupervised representation learning with deep convolutional generative adversarial networks," https://arxiv.org/abs/1511.06434, 2015.
[15] "Virusshare.com - because sharing is caring," https://virusshare.com/, accessed 1 Jan 2024.
[16] "Cuckoo sandbox - automated malware analysis," https://github.com/cuckoosandbox, accessed 1 Jan 2024.
[17] "Anaconda," https://www.anaconda.com/.
[18] D. Foster, *Generative Deep Learning*. O'Reilly Media, 2019.
[19] A. Dunmore *et al.*, "Generative adversarial networks for malware detection: a survey," https://arxiv.org/abs/2302.08558, 2023, arXiv:2302.08558v2 [cs.CR], 24 Feb 2023.
[20] M. Alazab, *Deep Learning Applications for Cyber Security*. Springer, 2019.
[21] A. Y. Mahgoub *et al.*, *Machine Learning Techniques for Cybersecurity*. Springer, 2023.