Leveraging Reinforcement Learning and Generative Adversarial Networks to Craft Mutants of Windows Malware against Black-box Malware Detectors

Phan The Duy
duypt@uit.edu.vn
Information Security Laboratory,
University of Information Technology
Vietnam National University, Ho Chi Minh city
Ho Chi Minh city, Vietnam

Tran Duc Luong
19521815@gm.uit.edu.vn
Information Security Laboratory,
University of Information Technology
Vietnam National University, Ho Chi Minh city
Ho Chi Minh city, Vietnam

Nguyen Hoang Quoc An
18520431@gm.uit.edu.vn
Information Security Laboratory,
University of Information Technology
Vietnam National University, Ho Chi Minh city
Ho Chi Minh city, Vietnam

Nguyen Huu Quyen
18521321@gm.uit.edu.vn
Information Security Laboratory,
University of Information Technology
Vietnam National University, Ho Chi Minh city
Ho Chi Minh city, Vietnam

Nghi Hoang Khoa
khoanh@uit.edu.vn
Information Security Laboratory,
University of Information Technology
Vietnam National University, Ho Chi Minh city
Ho Chi Minh city, Vietnam

Van-Hau Pham
haupv@uit.edu.vn
Information Security Laboratory,
University of Information Technology
Vietnam National University, Ho Chi Minh city
Ho Chi Minh city, Vietnam

ABSTRACT

To build an effective malware detector, it is required to collect a diversity of malware samples and their evolution, since malware authors always try to evade detectors through strategies of malware mutation. So, this paper explores the ability to craft mutants of malware for gathering numerous mutated samples in training a machine learning (ML)-based malware detector. Specifically, we leverage Reinforcement Learning (RL) and Generative Adversarial Networks (GAN) to generate adversarial malware samples against ML-based detectors. The more we use this approach with different targeted antivirus and malware samples in training the RL agent as a malware mutator, the more it learns how to avoid black box malware detectors. The experimental results in real-world dataset indicate that RL can help GAN in crafting variants of malware with executability preservation to evade ML-based detectors and VirusTotal. Finally, this approach can be used as an automated tool for benchmarking the robustness of malware detectors against the metamorphic malwares.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**.

KEYWORDS

Metamorphic malware, Windows malware, evasion attack, generative adversarial networks, reinforcement learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SoICT'22, December 01–03, 2022, Hanoi, Vietnam
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9725-4/22/12…$15.00
https://doi.org/10.1145/3568562.3568636

ACM Reference Format:
Phan The Duy, Tran Duc Luong, Nguyen Hoang Quoc An, Nguyen Huu Quyen, Nghi Hoang Khoa, and Van-Hau Pham. 2022. Leveraging Reinforcement Learning and Generative Adversarial Networks to Craft Mutants of Windows Malware against Black-box Malware Detectors . In *The 11th International Symposium on Information and Communication Technology (SoICT 2022), December 1–3, 2022, Hanoi, Vietnam.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3568562.3568636

1 INTRODUCTION

Malicious software, also known as malware, has grown in great popularity in recent years, posing a considerable threat to Internet users. Numerous forms of malware, such as viruses, worms, ransomware, rootkits, etc., are intentionally designed to damage victim systems, leak sensitive data, or gain unauthorized access to vulnerable systems, etc. [22]. These days, criminals usually make their effort to craft different malware variants which could resist as long as possible in victim machines by evading malware detectors. Therefore, the need for building robust anti-malware engines like antivirus software is becoming increasingly critical.

On the one hand, machine learning (ML) has emerged as an attractive approach [21], [25] for anti-malware vendors in spotting harmful applications. Through learning the distribution and distinctive features of various malware versions from a training dataset, an ML-based detector can generate a desirable outcome of differentiating between malicious and benign software [14], [2].

However, recent works [16], [7] indicated that the state-of-the-art ML-based malware detectors could be susceptible to some types of adversarial attacks. Skillful adversaries have a great interest in manipulating malware samples so as to bypass the deployed ML-based detectors [18], [20], [11]. One of the most advanced techniques to conduct these attacks is utilizing Generative Adversarial Networks (GAN) [15]. As its name implies, GAN is a generative model which can automatically produce a wide range of adversarial examples from the given data. It is constituted of two sub-models: Generator (G) and Discriminator (D). The idea of GAN originates from a zero-sum non-cooperative game. More specifically, G constantly attempts to generate fake data to fool D, while D is responsible for classifying examples as either real or fake. The GAN training process comes to a convergence point when D is completely deceived, or in other words, when G can generate highly plausible data. As a result, the GAN-based adversarial samples are likely to bypass ML-based detectors.

So, in this paper, we implement GAN as a generative malware tool in the context of Windows Portable Executable (PE) files. From malicious PE files, we extract essential features as input vectors for the GAN model to produce adversarial samples. However, in terms of GAN, attackers must have a great awareness of the features that characterize the target ML-based model. This is because the final goal of GAN is to craft adversarial examples that both resemble the original feature space and bypass the detection of the target system. Additionally, the generated GAN-based samples are only available as feature vectors, which are frequently impractical in the real world. Meanwhile, reinforcement learning (RL) could address those issues when conducting these kinds of attacks with executable files regardless of the feature space (structure or implementation of ML-based model, detection features). Therefore, we integrate RL [12] into our framework to craft new maliciousness-preserving PE malware from GAN-based adversarial feature vectors. The generated PE malware files are then used by adversaries against malware detectors in black-box mode, where attackers have no insight into the target system, to demonstrate the feasibility in practice.

Based on the above-mentioned analysis, we explore a new mutation scheme based on the idea of PEsidious [9] that leverages RL and GAN with the express purpose of crafting mutants of Windows malware which are likely to deceive malware detectors. The suggested solution can be applied to other systems like ELF, Android, etc. since the features are unrelated to file format. We desire that this framework would be a fundamental to build up defense mechanism for anti-malware engines in the future.

The remaining parts of this paper are summarized as follows. In Section 2, we mention some previous research work on malware, RL, and GAN. The following section describes the workflow of our proposed framework in detail. The environmental settings and performance evaluation outcomes are shown in Section 4. Finally, Section 5 includes our main conclusion and future work.

2 RELATED WORKS

According to Luca et al.[11], adversarial Windows malware samples could be crafted malware with small perturbations to evade the detectors. It is considered one of the major issues of robust antivirus in recognizing malware variants. In fact, the methods of crafting adversarial malware cause concerns in academia and industry due to the shortcomings of current ML-based malware detectors.

Also, the study [13] outlined a method for getting beyond static feature analysis-based anti-malware models. The attack model consists of two parts: a framework that can automatically detect anti-malware defects and the creation of new malware variants based on general rules. Gym-malware of Endgame research focuses on malicious Windows PE files and will be the basis for constructing other types of malicious code like PDFs and ELF Binaries. And with Open-AI Gym [6], access to this direction will be easier. Research conducted by Fang et al. utilizes the Environment to design Agents of RL. Implementation research papers are not yet available, and therefore the study is only referenced to give us an idea of how Agent of RL implementation is.

Employing GANs for malicious code modification succeeded beyond all expectations, and the work of Hu and Tan was one of the most notable ones. They offer a MalGAN method in which import features vector representations of malicious binary files are fed into GAN to produce plausible feature vectors that can deceive black box detectors. As a consequence, the True Positive Rate (TPR) was reduced from 97% to 0% when using ML algorithms such as Logistic Regression, SVM, Multilayer Perceptron, and VOTE; and from 97% to 20% in case of Random Forest and Decision Tree. However, the study did not provide any source code repositories to examine their results, making it impossible for us to validate. Fortunately, with the suggestion of notable solution implementations, Yanming Lai [1] helped us verify those above results after training.

Additionally, Learn2Evade [5] is the other approach of generating variants of portable document format (PDF) malware to bypass the detection of antivirus software. Therein, GAN is employed to put a minimum perturbation on the original PDF for creating evasive PDF malware. Obviously, these evasive variants raise a serious security concern about the robustness of ML-based detectors in the presence of adversaries against defense techniques.

Regarding the level of data modification, Maiorca et al. [19] stated that the evasion attack against a malware detector can be staged at the input level or the feature level. In the case of input level, the attacker can craft new evasive malware samples with malicious functionality without breaking the original operation of malware files. On the contrary, adversarial attacks only modify the feature values of the malware samples, without creating the corresponding new files. It means that the modification of input data level produces more practical samples than the others. To recognize the mutated malware samples, RL is considered a powerful approach to conduct integrity verification in the generation of adversarial malware examples. Typically, AIMED-RL [17] is an RL-based method to synthesize adversarial examples that lead ML-based detectors to misclassify malware files without corrupting file format or compromising their functionality.

Besides, PEsidious [9] is another work that utilizes RL and GAN to mutate the original malware sample. The authors evaluate the performance of PEsidious on Gradient Boosting algorithm and VirusTotal to show the feasibility to generate a new metamorphic malware bypassing antivirus. Nonetheless, the experimental scenario is not conducted in comprehensive method to clarify the effect of RL or GAN in crafting malware mutants.

To summarize, the difference of our work and PEsidious is the number of targeted ML-based models and the action space for training RL agent. We use more ML algorithms, MalColv model and VirusTotal as blackbox malware detectors, to benchmark their robustness against mutants on different real-world datasets. In addition, we investigate the impact of RL-only and RL-GAN approach to demonstrate the effectiveness of combination method.

[1]Yanming Lai: https://github.com/yanminglai/Malware-GAN

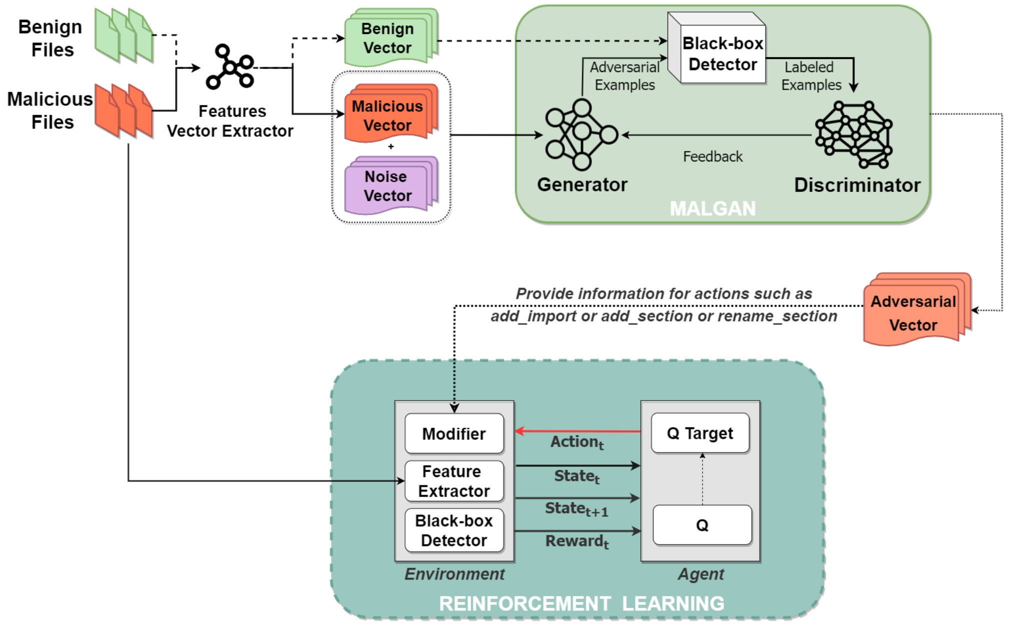

Figure 1: The architecture of crafting PE malware mutants using RL and GANs.

3 METHODOLOGY

There has been a lot of research done over the last decade on how to use Artificial Intelligence (AI) to classify malware and benign files. However, in recent years, adversaries have begun to employ AI to attack these defensive AI models. In the past, notable work in this domain has been done, with researchers using reinforcement learning or generative adversarial networks as their weapons of choice to modify the states of a malware executable to fool antivirus agents. Inspired from the work of Bedang Sen and Chandni Vaya in PESidious [9], our solution employs a combination of deep reinforcement learning and GANs to overcome some limitations encountered when using these approaches independently. Particularly, our work deeply investigate more aspects of efficiency between the fusion of GAN and RL on different settings for mutating an original malware to escape from of the eyes of antivirus.

Figure 1 exhibits the entire architecture of malware mutation framework, which is developed from the MalGAN scheme [16] and the DQEAF model [12]. Additionally, we aim to evade several black-box malware detection algorithms while making adjustments to the RL action space in the experiment. The combination of GAN and RL to create a malware mutation system mainly enhances the evasion of mutated malware and evaluates the effectiveness of crafting malware mutations compared to the existing strategies in MalGAN [16] and DQEAF-based approach [12].

There exists two modules taking over different tasks in the workflow. The first module, MalGAN [16] is responsible for transforming the malicious feature vectors M into adversarial examples A. Specifically, the original PE files would undergo a feature extractor that respectively outputs the benign feature vectors B and malicious ones M. Each of them contains some necessary attributes that we target to serve for RL action space (*add_import, add_section, rename_setion*) such as Import Table, Section Names, etc. Next, M would be mixed with noise vectors Z as input for Generator to generate adversarial examples A. Each feature values of Z is randomly selected from a uniform distribution in range of $[0, 1]$. During MalGAN training stage, adversarial samples from Generator along with benign feature vectors are labeled (benign or malicious) by black-box detector that then are fed into Discriminator learning phase. The main task of Discriminator is imitating the classification ability of black-box detector by computing the loss function between its outcome and the received labels. The Generator obtains this loss value as feedback to cultivate its data production that can bypass the black-box detector as much as possible. The MalGAN training procedure occurs until both black-box detector and Discriminator are deceived by the generated plausible examples A.

As soon as the MalGAN training process finishes, RL module makes use of the GAN-based adversarial vector samples A to craft new PE malware mutants that still preserve malicious behaviors of the original files. And, in this part, we build RL module as Deep Q-network anti-malware Engines Attacking Framework (DQEAF) to take the features of the malware as an input. Furthermore, to apply DQEAF in this problem, all elements will be described as Markov Decision Process (MDP) model as follows:

- State s_t is a vector including the features of malware from PE file.
- Given a state s_t, the DRL agent will take an action a_t. And action space A is a vector of 9 actions that the agent can perform.
- Reward r_t is determined for each training *TURN* based on the label from the classifier and the number of actions taken. And *MAXTURN* is defined, which means that the agent

should claim failure if $MAXTURN$ steps of modification have been taken. Reward r_t is 0 if the label is malicious, and is calculated using **Eq. (1)** when the label is benign.

$$r_t = 20^{-(TURN-1)/MAXTURN} * 100 \tag{1}$$

And the agent is trained to maximize the expectation of the cumulative discounted reward given by **Eq. (2)**

$$R = \sum_{t=1}^{T} \gamma^{t-1} r_t \tag{2}$$

where $\gamma \in (0, 1]$ is a factor discounting future rewards.

- The optimal action-value function $Q^*(s_t, a_t)$ which shown in **Eq. (3)** estimates the expected reward of taking an action a at state s_t.

$$Q^*(s_t, a_t) = E\{r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})\} \tag{3}$$

During the learning process, the agent minimizes the error estimated by the loss function **Eq. (4)** by optimizing the weights θ.

$$l_t(\theta_t) = ((r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{t-1})) - Q(s_t, a_t; \theta_t))^2 \tag{4}$$

Notably, the RL module takes a malware sample as an input to process the actions during trial attempts to bypass antivirus. If the action selected by RL agents corresponding to the feature of PE file header like Section Name or Import Table, the adversarial feature vectors generated by GAN is utilized to create a new mutant of malware sample. The environment and the agent collaborate to mutate the malware sample and calculate the reward, relied on a series of actions during the training time of RL agent. After finishing the training phase, the Q network is leveraged as the optimal policy network to produce the sequence of actions on the original malware sample for crafting a new evasive malware.

4 IMPLEMENTATION AND EXPERIMENTAL RESULTS

4.1 Dataset

We use two datasets from the VirusTotal and VirusShare sources to train the mutation system and the black-box ML-based detector. The VirusTotal dataset includes 7000 backdoor samples used to train mutants and then sifted 200 samples for testing. VirusShare dataset contains samples from malware families like Locker (300 samples), Mediyes (1450 samples), Winwebsec (4400 samples), Zbot (2100 samples), Zeroaccess (690 samples), and Backdoor (500 samples) to evaluate the effectiveness of the combined GAN and RL model. Details of the training dataset are shown in Table 1.

Moreover, we train black-box ML-based detectors through the above two datasets and look for an extensive training repository with more than 1100K samples, Ember (2018); details are in Table 2.

As long as the convenience of testing and experimenting, we also chose to provide test data samples and actual data samples, which are listed in Table 3.

4.2 Implementation

Our system is implemented with Python 3.6.18 on the Ubuntu virtual machine (VM) with configuration of RAM 16 GB and Disk 200 GB. The ML frameworks such as Tensorflow, sklearn are also installed in this VM. We also use PEsidious [2], an open-source to mutate PE malwares to implement our system.

Table 1: The training dataset for GANs and RL

Dataset	Training data	Testing data	Model
VirusTotal	7000	200	RL + GAN
VirusShare	9240	200	RL + GAN

Table 2: The training dataset for black-box detectors

Dataset	Training data	Testing data	Black-box detectors
Ember	900K	200K	MalConv
VirusTotal	7000	200	Random Forest, Gradient Boosting, Decision Tree
VirusShare	9240	200	Random Forest, Gradient Boosting, Decision Tree

Table 3: The testing dataset for the proposed system

	Dataset	Quantity	Category
Testing data	VirusTotal Source	200	Backdoor
Real-world data	MalShare Source	120	MICS

4.2.1 ML-based detectors. In this work, several ML algorithms trained on the datasets mentioned above are used to serve as malware detectors for benchmarking the mutants generated by mutation mechanism. More specifically, it detects malware with static analysis by classifying samples as malicious or benign without executing them, unlike dynamic malware detection, which detects behavior-based malware. Its runtime consists of time-dependent system analysis calls [3], [10], [23]. Although the use of static analysis is known to be of little use against today of sophisticated malware [8], it is nevertheless considered an essential layer of defense in detecting malicious code because it allows malicious files to be detected without execution of anything.

To train ML algorithms for serving as black-box detectors, we set the learning rate to 0.0001, the epoch number of 100 and the batch size to 32. However, apart from others, for the MalConv model, we assign the learning rate to 0.05. To optimize the efficiency of the trained model, we use three options to map the specific features of the data sample, VirusTotal, and VirusShare dataset. Feature extraction options include: *1) Imports or Sections only, 2) Import and Sections only 3) Both Imports, Sections, and some basic file information.*

Nonetheless, we only focus on implementing option (2) because the model only works with Imports and Sections, so the defect or redundancy of some features also affects the ability of running. The information extracted from the above options will be used as input to train the black box detector with algorithms such as Random

[2]PEsidious: https://github.com/CyberForce/Pesidious

Table 4: The action space of RL module

Action	Based on GAN	Definition
`rename_section`	✓	Modify the name of section
`add_section`	✓	Add new (unused) sections
`add_import`	✓	Add DLLs or functions to DLLs in Import Table
`pe_stripper`		Remove redundant features
`edit_tls`		Change Thread Local Storage
`load_config_dir`		Load config directory
`append_overlay`		Append bytes to the overlay
`remove_debug`		Remove debug info
`break_optional _header_checksum`		Modify checksum value in optional header

Forest, Gradient Boosting, and Decision Tree. In addition, we utilize the Ember dataset [1] in training MalConv model.

4.2.2 MalGAN module. This module focus on training GANs to be able to generate new information for actions like *add_import* or *add_sections* or *rename_sections*. And to enhance the performance of model in generating mock samples, 2 main parts of GANs are Generator (G) and Discriminator (D) which will be trained together. The dimension of the latent vector or noise vector to feed to Generator input is set to 10. Meanwhile, the feature vector size for GAN (both Generator and Discriminator) is 5428 (by mapping all imports and sections from our dataset), instead of 160 dimensions as MalGAN. The 5428-dimensional vectors includes 5141 imports and 287 sections resulting True Positive Rate (TPR) of blackbox detectors to 0%. Furthermore, G and D have the same multi-layer feed-forward neural network, in which the hidden layer size of Generator and Discriminator was set to 256. The activation function for the Generator and Discriminator is LeakyReLU. The MalGAN module is trained with the number of epochs 100, and batch size of 32.

4.2.3 Reinforcement Learning module. In this module, we rebuild DQEAF's environment which suitable for agent to learn. The environment defined contains observation space (reward) and action space.

- `Action Space:` Agent utilizes 9 actions as listed in Table 4.
- `Reward:` The reward for every action is determined based on the black-box detector's prediction score. And, the reward is calculated as **Eq. (1)**.

In addition, agent RL is trained with a basic CNN module containing 2 hidden convolutional layers with 256 nodes and 64 nodes, respectively. The activate function for both layers is ReLU. And the 518-dimensions observation of each step is normalized to $[0.5, -0.5]$ by batch normalization for each layer of the network.

The episodes for training RL agent is set to 1000, and discount factor γ is 0.99. Additionally, the number of maximum mutations allowed on each PE file ($Max_{mutations}$) is set to 80 by default. For modifying PE files in RL-based mutation phase, we use Lief library [24] to parse and modify PE file, PE Bliss [4] for rebuilding PE file to a new file.

4.3 Evaluation metrics

To analyze the evasion of mutated files against the black-box ML-based detector, we use the average prediction score (PS_{ml}) on each mutant, ranging of $[0, 1]$, as in **Eq. (5)**.

$$PS_{ml} = \frac{1}{N}\sum_{i=1}^{N} S_i \tag{5}$$

, where S_i is the confident score of ML-based detectors on malware mutants; N is the total number of malware mutants putting to a specific black-box ML-based detector.

Also, this work uses DS_{ml}, the ratio of detected malware files of ML-based detector when inspecting mutated files, with d is the number of detected files in total mutated files N. It is defined as in **Eq. (6)**.

$$DS_{ml} = \frac{d}{N} \tag{6}$$

Regarding the evaluation metrics for evasive samples against VirusTotal engine, **Eq. (7)** are utilized to compute the Average Detection Score of VirusTotal Engine (DS_{vt}).

$$DS_{vt} = \frac{1}{N}\sum_{i=1}^{N} PEG_i \tag{7}$$

, where PEG_i is the ratio of VirusTotal engines returning malware result for the mutated sample i to the all used engines; N is the total number of malware mutants putting to VirusTotal engine.

4.4 Experimental Results

4.4.1 Performance of Baseline Detectors against GAN-based samples. According to the testing data in Table 2, we implement GAN to generate adversarial malware examples from original malicious ones. Both types of samples (original & adversarial) are fed into different black-box ML-based models for classification. For conducting the following experiments, the malware prediction threshold of black-box models is set to 0.6. It means that, if the prediction score surpasses 0.6, the input sample is considered as malware.

As a result, Table 5 gives the detection performance of ML-based detectors before and after being attacked with adversarial samples by GAN. When it comes to original samples, the overall performance of such malware detectors witnesses a high level in the metrics of AUC and ODR (original detection rate), which are respectively greater than 0.8 and 80% in all cases. However, those ML-based models misclassify adversarial GAN-based malware samples when the adversarial detection rate (ADR) is about 18.24% in the best case (MalConv). Note that, ADR is calculated by the classification rate of black-box detectors against only the adversarial GAN-based samples that are mutated from original malware ones.

The aforementioned result has indicated that our MalGAN module is working well in generating plausible adversarial malware vectors against black-box classifiers.

4.4.2 The evasive performance of mutants with RL and RL-GAN methods. In this part, we conduct two kinds of attacks against ML-based detectors to prove the attacking effectiveness of our proposed RL-GAN system compared to RL approach [12]. More specifically, in the context of RL-GAN method, the action space of RL module

Table 5: The original malware and mutated malware detection rate of black-box ML-based detectors

Black-box detectors	AUC	ODR	ADR
MalConv (Ember)	0.9964	96.49%	18.24%
Random Forest (VirusTotal)	0.8740	87.82%	9.11%
Gradient Boosting (VirusTotal)	0.9031	89.33%	14.29%
Decision Tree (VirusTotal)	0.8131	80.13%	5.977%
Random Forest (VirusShare)	0.8813	86.97%	10.97%
Gradient Boosting (VirusShare)	0.9007	88.61%	15.21%
DecisionTree (VirusShare)	0.8371	82.11%	7.8%

Table 6: The performance of Windows malware mutant against black-box detectors when using RL-only and RL-GAN method

Black-box detectors	Original (PS_{ml})	Mutated (PS_{ml}) RL	RL+GAN
MalConv (Ember)	0.9649	0.6533	**0.5510**
Random Forest (Virus Total)	0.8782	0.5123	**0.4012**
Decision Tree (Virus Total)	0.8013	0.4655	**0.3120**
Gradient Boosting (Virus Total)	0.8933	0.5811	**0.5000**
Random Forest (Virus Share)	0.9110	0.5561	**0.4197**
Decision Tree (Virus Share)	0.8199	0.5100	**0.4122**
Gradient Boosting (Virus Share)	0.9200	0.6108	**0.5597**

consists of only 3 GAN-provided actions (Table 4) while the 6 others are utilized in case of using only RL. Also, we use 120 MICS Real-world files (Table 3) as the original malware data, which is then used for crafting malware variants in this experiment. The black-box models are pre-trained in accordance with the training dataset in Table 2.

Table 6 has shown a downtrend by more than 30% in the average prediction score PS_{ml} of all black-box malware detectors when dealing with PE mutants files compared to original ones. Furthermore, the attacking performance of RL-GAN method is better than RL-only approach in every single case. Almost all of PS_{ml} in the context of RL-GAN exceeds approximately 10% compared to RL-only method. For instance, in the best case, Decision Tree (Virus Total) predicts RL-only mutated files with PS_{ml} of 0.4655 while it solely has PS_{ml} of 0.3120 when classifying RL-GAN malware mutants. This experiment partially proves the effectiveness of our RL-GAN framework in evading black-box malware detectors.

Table 7: Test average prediction score (PS_{ml}) provided by Black-box ML-based detector

Black-box detectors	4 actions	9 actions
MalConv (Ember)	**0.557**	0.5933
Random Forest (VirusTotal)	**0.4136**	0.4513
Gradient Boosting (VirusTotal)	**0.5477**	0.6011
Decision Tree (VirusTotal)	**0.3512**	0.4011
Random Forest (VirusShare)	**0.4019**	0.4503
Gradient Boosting (VirusShare)	**0.56**	0.6413
DecisionTree (VirusShare)	**0.3313**	0.415

4.4.3 *The evasive performance of the mutated PE malware files with different Action space settings.*

Regarding RL, action space plays a vital role in training an effective model, especially in crafting malware samples. The better action space is, the more evasive the malware is. From the above point of view, we divide into 2 cases of action space to evaluate the performance of our framework:

- Case 1: using 4 actions (*rename_section, add_section, add_import, append_overlay*).
- Case 2: using all 9 actions listing in Table 4.

And these experiments are also evaluated based on PS_{ml} metric, as in **Eq. (5)**. As the result shown in Table 7, in all 7 black box detectors, RL method with 4-actions has proven itself effective in increasing the evasion of mutant malware. The highest and lowest PS_{ml} values of these black-box detectors in the face of Windows malware due to RL method with 4-actions mutations are 0.3313 and 0.56, respectively. Moreover, the PS_{ml} values of RL method with 4-actions is significantly reduced by more than 12% with RL method with 9-actions.

4.4.4 *The evasive performance of the mutated PE files against black-box detectors with different malware mutation budget.*

Additionally, we conduct an experiment based on our malware mutants with 4 black-box ML-based detectors at 3 different levels of the maximum allowable number of mutations ($Max_{mutations}$) to show that ($Max_{mutations}$) could affect the performance of our RL-GAN model. In this experiment, we set the black-box models' malware threshold at 0.8, which indicates that any samples are only considered malware if their prediction score exceeds 0.8. Also, PS_{ml} and DS_{vt} metrics are respectively used as evaluation results.

According to the results in Table 8, in all 4 types of black box detectors, we easily realize that the more $Max_{mutations}$ is, the less PS_{ml} and DS_{vt} are. In terms of the MalConv detector, the values of PS_{ml} and DS_{vt} with $Max_{mutations}$ = 80 and 160 decrease to

Table 8: Test average prediction score and Average detection score of VirusTotal Engine provided by Black-box ML-based detector as Malware with RL trained using only four actions

Black-box detectors	$Max_{mutations}$	PS_{ml}	DS_{vt}
Random Forest (VirusTotal source)	80	0.5971	61/75
	120	0.5012	55/73
	160	0.471	45/72
Decision Tree (VirusTotal source)	80	0.5334	63/77
	120	0.4287	58/76
	160	0.3617	50/73
Gradient Boosting (VirusTotal source)	80	0.6813	53/75
	120	0.604	46/73
	160	0.5477	41/72
MalConv (Ember dataset)	80	0.7991	54/75
	120	0.6511	46/73
	160	0.5812	39/72

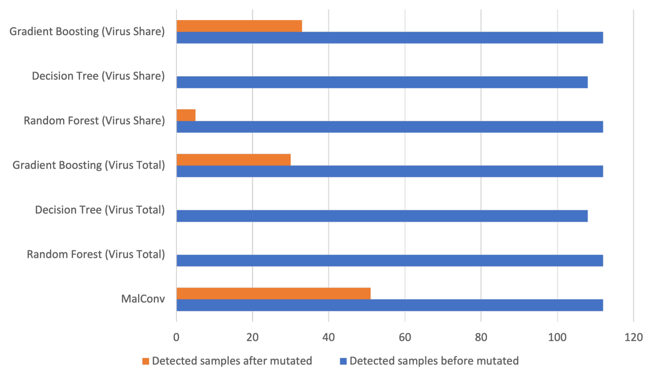

Figure 2: Graph showing the total number of data samples in Real-world data detected as malware with RL-GAN trained using only four actions.

about 27%, 24% respectively. Similar to the MalConv detector, the remaining detectors have also reduced 2 values of PS_{ml} and DS_{vt} over 20%.

4.4.5 The evasive performance of the mutated malware files against black-box detectors on Real-world dataset. In this scenario, we use real-world dataset with 120 malware files, aforementioned in Table 3, to benchmark the evasion performance of crafted malware by only 4 actions (Case 1 in section 4.4.3).

Fig. 2 gives the results of undetectable mutant rate of ML-based detectors on real-world dataset. To be more specific, more than a half of 120 malware mutants can avoid the detection of MalConv model. Even, Decision Tree and Random Forest algorithms, which are trained on both VirusTotal and VirusShare dataset, nearly could not recognize any mutants as malware. Besides, we also send these 120 PE mutated files to VirusTotal engines for classification. As a consequence, the number of VirusTotal engines, which can detect our mutants, drops from 65/77 to 49/77. The result has pointed out

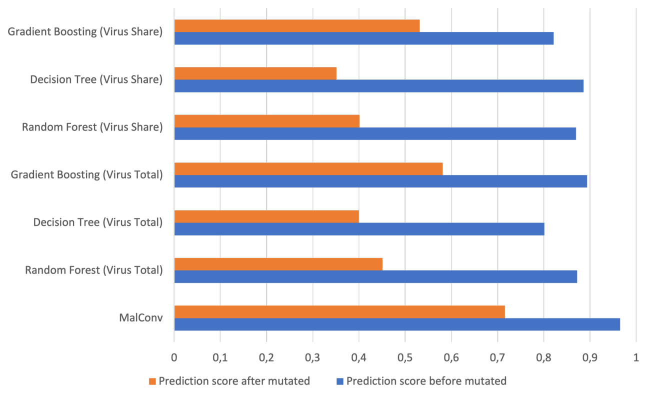

Figure 3: Average prediction score of ML-based detectors on 120 original malware files and its mutants on Real-world dataset.

that our method is able to craft mutants of malware files, bypassing ML-based anti-malware and real-world malware detection engines.

Fig. 3 describes the average prediction score of ML-based detectors, PS_{ml}, on mutated malware files. Clearly, after encountering the mutants from original malware, PS_{ml} is significantly reduced to nearly below the threshold of malware detection task over all detectors.

5 CONCLUSION

This paper explores the ability of Reinforcement Learning (RL) and GANs in mutating Windows malware samples. Therein, adversarial features generated by GANs are used to fed to train RL agents for crafting new metamorphic malware mutants. The combination of RL and GANs can help malware enhance the evasive capability against ML-based detectors. We have already evaluated the robustness of VirusTotal engine and various ML-based detectors, especially MalConv - one of state-of-the-art ML-based detectors, on spotting mutants of Windows malware. The experimental results on real-world datasets proves that the combination method of RL and GAN has promising approach to understand the robustness of ML-based detectors, also the evolution of evasive malware on computers.

In the future, we intend to investigate the robustness of ML-based malware detectors that use the dynamic analysis approach instead of analyzing the PE file features. More specifically, API calls are considered actions in RL agent training to be put into the PE file to create chaos in the sequence of behaviors to elude the recognition of antivirus.

ACKNOWLEDGMENTS

This research is funded by University of Information Technology - Vietnam National University HoChiMinh City under grant number of D1-2022-51.

REFERENCES

[1] Hyrum S. Anderson and Phil Roth. 2018. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. https://doi.org/10.48550/ARXIV.1804.04637

[2] Ömer Aslan Aslan and Refik Samet. 2020. A Comprehensive Review on Malware Detection Approaches. *IEEE Access* 8 (2020), 6249–6271. https://doi.org/10.1109/ACCESS.2019.2963724

[3] Ben Athiwaratkun and Jack W. Stokes. 2017. Malware classification with LSTM and GRU language models and a character-level CNN. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2482–2486. https://doi.org/10.1109/ICASSP.2017.7952603

[4] BackupGGCode. 2015. PE Bliss - Cross-Platform Portable Executable C++ Library. https://github.com/BackupGGCode/portable-executable-library.

[5] Ho Bae, Younghan Lee, Yohan Kim, Uiwon Hwang, Sungroh Yoon, and Yunheung Paek. 2021. Learn2Evade: Learning-Based Generative Model for Evading PDF Malware Classifiers. *IEEE Transactions on Artificial Intelligence* 2, 4 (2021), 299–313. https://doi.org/10.1109/TAI.2021.3103139

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. https://doi.org/10.48550/ARXIV.1606.01540

[7] Lingwei Chen, Yanfang Ye, and Thirimachos Bourlai. 2017. Adversarial Machine Learning in Malware Detection: Arms Race between Evasion Attack and Defense. In *2017 European Intelligence and Security Informatics Conference (EISIC)*. 99–106. https://doi.org/10.1109/EISIC.2017.21

[8] Fred Cohen. 1987. Computer viruses: Theory and experiments. *Computers & Security* 6, 1 (1987), 22–35. https://doi.org/10.1016/0167-4048(87)90122-2

[9] CyberForce. 2015. PEsidious - Malware Mutation Using Reinforcement Learning and Generative Adversarial Networks. https://github.com/CyberForce/Pesidious.

[10] George E. Dahl, Jack W. Stokes, Li Deng, and Dong Yu. 2013. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 3422–3426. https://doi.org/10.1109/ICASSP.2013.6638293

[11] Luca Demetrio, Scott E. Coull, Battista Biggio, Giovanni Lagorio, Alessandro Armando, and Fabio Roli. 2021. Adversarial EXEmples: A Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection. *ACM Trans. Priv. Secur.* 24, 4, Article 27 (9 2021), 31 pages. https://doi.org/10.1145/3473039

[12] Zhiyang Fang, Junfeng Wang, Boya Li, Siqi Wu, Yingjie Zhou, and Haiying Huang. 2019. Evading Anti-Malware Engines With Deep Reinforcement Learning. *IEEE Access* 7 (2019), 48867–48879. https://doi.org/10.1109/ACCESS.2019.2908033

[13] Zhiyang Fang, Junfeng Wang, Boya Li, Siqi Wu, Yingjie Zhou, and Haiying Huang. 2019. Evading Anti-Malware Engines With Deep Reinforcement Learning. *IEEE Access* 7 (2019), 48867–48879. https://doi.org/10.1109/ACCESS.2019.2908033

[14] Daniel Gibert, Carles Mateu, and Jordi Planes. 2020. The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges. *J. Netw. Comput. Appl.* 153, C (3 2020), 22 pages. https://doi.org/10.1016/j.jnca.2019.102526

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[16] Weiwei Hu and Ying Tan. 2017. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. *CoRR* abs/1702.05983 (2017). arXiv:1702.05983 http://arxiv.org/abs/1702.05983

[17] Raphael Labaca-Castro, Sebastian Franz, and Gabi Dreo Rodosek. 2021. AIMED-RL: Exploring Adversarial Malware Examples with Reinforcement Learning. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part IV* (Bilbao, Spain). Springer-Verlag, Berlin, Heidelberg, 37–52. https://doi.org/10.1007/978-3-030-86514-6_3

[18] Deqiang Li, Qianmu Li, Yanfang (Fanny) Ye, and Shouhuai Xu. 2021. Arms Race in Adversarial Malware Detection: A Survey. *ACM Comput. Surv.* 55, 1, Article 15 (11 2021), 35 pages. https://doi.org/10.1145/3484491

[19] Davide Maiorca, Battista Biggio, and Giorgio Giacinto. 2019. Towards Adversarial Malware Detection: Lessons Learned from PDF-Based Attacks. *ACM Comput. Surv.* 52, 4, Article 78 (8 2019), 36 pages. https://doi.org/10.1145/3332184

[20] Nuno Martins, José Magalhães Cruz, Tiago Cruz, and Pedro Henriques Abreu. 2020. Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: A Systematic Review. *IEEE Access* 8 (2020), 35403–35419. https://doi.org/10.1109/ACCESS.2020.2974752

[21] Athiq Mohammed, G. Viswanath, K. babu, and T. Anuradha. 2020. *Malware Detection in Executable Files Using Machine Learning*. 277–284. https://doi.org/10.1007/978-3-030-24322-7_36

[22] Ori Or-Meir, Nir Nissim, Yuval Elovici, and Lior Rokach. 2019. Dynamic Malware Analysis in the Modern Era—A State of the Art Survey. *ACM Comput. Surv.* 52, 5, Article 88 (9 2019), 48 pages. https://doi.org/10.1145/3329786

[23] Razvan Pascanu, Jack W. Stokes, Hermineh Sanossian, Mady Marinescu, and Anil Thomas. 2015. Malware classification with recurrent networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1916–1920. https://doi.org/10.1109/ICASSP.2015.7178304

[24] Quarkslab. 2017. LIEF - Library to instrument executable formats. https://github.com/lief-project.

[25] Umara Urooj, Bander Ali Saleh Al-rimy, Anazida Zainal, Fuad A. Ghaleb, and Murad A. Rassam. 2022. Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions. *Applied Sciences* 12, 1 (2022). https://doi.org/10.3390/app12010172