A Method for Generating Malware Countermeasure Samples Based on Pixel Attention Mechanism

Xiangyu Ma[1], Yuntao Zhao[1*], Yongxin Feng[2], and Yutao Hu[1]
[1] School of information science and engineering, Shenyang Ligong University
6 Nanping Middle Road, Hunnan District, Shenyang, Liaoning Province, 110159, China
[e-mail: maxiangyu0729@126.com, zhaoyuntao_2014@163.com, huyutao_2023@163.com]
[2] Graduate School, Shenyang Ligong University
6 Nanping Middle Road, Hunnan District, Shenyang, Liaoning Province, 110159, China
[e-mail:fengyongxin@263.net]
*Corresponding author : Yuntao Zhao

Received June 21, 2023; revised December 10, 2023; revised December 21, 2023; accepted February 7, 2024; published February 29, 2024

Abstract

With information technology's rapid development, the Internet faces serious security problems. Studies have shown that malware has become a primary means of attacking the Internet. Therefore, adversarial samples have become a vital breakthrough point for studying malware. By studying adversarial samples, we can gain insights into the behavior and characteristics of malware, evaluate the performance of existing detectors in the face of deceptive samples, and help to discover vulnerabilities and improve detection methods for better performance. However, existing adversarial sample generation methods still need help regarding escape effectiveness and mobility. For instance, researchers have attempted to incorporate perturbation methods like Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), and others into adversarial samples to obfuscate detectors. However, these methods are only effective in specific environments and yield limited evasion effectiveness. To solve the above problems, this paper proposes a malware adversarial sample generation method (PixGAN) based on the pixel attention mechanism, which aims to improve adversarial samples' escape effect and mobility. The method transforms malware into grey-scale images and introduces the pixel attention mechanism in the Deep Convolution Generative Adversarial Networks (DCGAN) model to weigh the critical pixels in the grey-scale map, which improves the modeling ability of the generator and discriminator, thus enhancing the escape effect and mobility of the adversarial samples. The escape rate (ASR) is used as an evaluation index of the quality of the adversarial samples. The experimental results show that the adversarial samples generated by PixGAN achieve escape rates of 97%, 94%, 35%, 39%, and 43% on the Random Forest (RF), Support Vector Machine (SVM), Convolutional Neural Network (CNN), Convolutional Neural Network and Recurrent Neural Network (CNN_RNN), and Convolutional Neural Network and Long Short Term Memory (CNN_LSTM) algorithmic detectors, respectively.

Keywords: Malware, Generative Adversarial Networks, Deep Learning, Pixel Attention Mechanism, Adversarial Samples

1. Introduction

The Internet plays a crucial role in today's politics, economics, and education. With the rapid development of intelligent terminal devices and Internet technologies, the Internet has become indispensable in people's work and daily lives. However, the frequency of network attack incidents has been increasing yearly, particularly in the increasingly complex network environment, where this trend has become more evident. Network attacks threaten individuals' security and jeopardize the interests and security of society and nations. Therefore, it is of utmost importance to continuously enhance network security measures to mitigate the damages that these attacks [1] may bring. Network security has always been a hot research topic in the academic field. Only by continually raising awareness of network security [2] and improving network security measures can we better enjoy the convenience of the Internet.

According to relevant reports, the number of infected terminals with Trojan or zombie network malware [3] in China exceeded 1.19 million in 2021. At the same time, the number of incidents involving tampering, backdoor implantation, and counterfeit websites exceeded 22,000. The National Vulnerability Database Platform (CNVD) has compiled information on 1,660 information system security vulnerabilities, including 570 high-risk vulnerabilities. Additionally, in 2021, Huawei Technologies recorded 20,203 vulnerabilities, of which over 2,591 were classified as critical vulnerabilities, and there were as many as 8,451 high-risk vulnerabilities. Verizon, the American telecommunications company, published the "2022 Data Breach Investigations Report," which indicated an increasing number of cyberattack cases in the Asia-Pacific region, including phishing attacks and telephone eavesdropping. Furthermore, in March 2022, Toyota Motor Corporation experienced system paralysis due to a ransomware attack on one of its suppliers. In May of the same year, a subsidiary of the Nikkei Group in Singapore was also targeted by a ransomware attack. That month, SpiceJet, an Indian airline, was also hit by a ransomware attack, resulting in hundreds of passengers being stranded at the country's airports. Therefore, research on adversarial samples of malicious software has become increasingly important, as it aids in discovering similar types of malicious software and their variants, quickly identifying the employed disguise techniques, threat levels, infection strategies, and other relevant information. This, in turn, enhances the robustness of malicious software detection algorithms.

The objective of traditional methods for generating adversarial samples [4] against malicious software is to mutate and obfuscate the code and behavior of the malware to evade conventional detection and defense mechanisms. These methods include polymorphic mutation, code encryption, self-modifying code, sandbox evasion, junk code injection, and dynamic link library hijacking. However, these methods still have several drawbacks. Firstly, although they can make it difficult for malicious software to be recognized by traditional detection techniques, they are not wholly undetectable. Security researchers and software vendors continuously improve detection technologies and algorithms to counter malware's mutation and obfuscation strategies. Secondly, generating adversarial samples may require significant time and computational resources. Operations such as mutation, encryption, or code injection can make the malware larger and more complex, thus impacting its operational efficiency. Additionally, specific adversarial sample generation methods may cause damage to the malware itself, leading to erroneous or unstable behavior. Some methods rely on specific runtime environments or target systems, limiting their applicability and effectiveness. Most importantly, as security technology advances, the generation and detection of adversarial samples against malicious software are also evolving. The effectiveness of traditional methods may gradually diminish, resulting in an ongoing cycle of negative interaction between

malware and security measures. Therefore, adversarial sample generation methods must be constantly updated and improved to maintain effective countermeasures against malicious software. In further research, emphasis should be placed on overcoming these limitations and proposing more reliable and effective methods for generating adversarial samples to enhance the detection [5] and defense capabilities against malware.

To address the issues mentioned earlier, this paper proposes a malicious software adversarial sample generation method based on a pixel attention mechanism to produce more reliable adversarial samples and construct a more stable adversarial sample generation model. The main contributions are as follows:

- A sandbox environment was employed to construct a dataset of malicious software API call sequences, and the sequences were transformed into word vectors using the FastText model, resulting in the generation of a matrix of word vectors.
- The malware application programming interface (API) call sequence dataset is constructed using a sandbox environment and transformed into a word vector matrix using the FastText model.
- The semantic relationships within the malicious software API call sequence are combined with computer vision by converting the generated word vector matrix into grayscale images. This transformation converts the malicious software detection problem into an image classification task.
- The generation of adversarial samples is based on the DCGAN model, incorporating a pixel attention mechanism. This mechanism exhibits higher sensitivity to pixel values, enabling accurate capture of crucial features within the images. It enhances the modeling capability of the DCGAN model, thereby significantly improving both the stability of the model and the quality of the adversarial samples.
- Detailed evaluation and comparison were conducted on other similar adversarial sample generation models. Experimental results demonstrate that the adversarial samples generated by PixGAN achieved evasion rates of 97%, 94%, 35%, 39%, and 43% on Random Forest (RF), Support Vector Machine (SVM), Convolutional Neural Network (CNN), Convolutional Neural Network and Recurrent Neural Network (CNN_RNN), and Convolutional Neural Network and Long Short Term Memory (CNN_LSTM) algorithm detectors, respectively.

The remaining sections of this paper are organized as follows: Section 2 introduces the principles of attention mechanism and generative adversarial network models. Section 3 presents the relevant contributions in adversarial sample generation and detection. Section 4 provides a detailed description of our proposed method for adversarial sample generation, including the visualization representation method for malicious software and improvements to the DCGAN network model. Section 5 showcases the experimental process and result analysis. Finally, Section 6 presents the conclusions and outlines future research directions.

2. Background

In this section, we provide a brief overview of the principles of the attention mechanism, generative adversarial network models, and the FastText model.

Firstly, the attention mechanism [6] is an approach used in artificial neural networks that allows the model to focus on critical information and the most relevant parts when processing sequential data. This method was initially proposed to address the challenges of handling long sequences in natural language processing tasks. In the attention mechanism, each input is

assigned a weight representing its importance to the output. A learned model calculates these weights, often normalized using the softmax function to ensure that the sum of all input weights is equal to 1. By assigning different weights, the model can concentrate on input positions relevant to the current output. Therefore, the attention mechanism has been widely applied in various fields, such as natural language processing, speech recognition, image classification, and machine translation, achieving excellent performance in many tasks. In our subsequent work, it provides a more flexible and efficient approach for processing grayscale images of malicious software.

Generative adversarial networks [7] are a robust machine learning framework to generate realistic and persuasive synthetic data samples. They consist of two main components: the generator and the discriminator. The generator aims to produce synthetic data samples that resemble actual data samples. In contrast, the discriminator aims to differentiate between actual and synthetic data samples generated by the generator. This concept can be summarized as follows.

$$min_G max_D V(G,D) = E_{x\sim P_{data}(x)} log D(x) + E_{Z\sim P_z(z)} \log\left(1 - D(G(Z))\right) \quad (1)$$

The min-max Equation, also known as Equation 1, the value function $V(G,D)$ is defined. When GAN is employed for data generation, we consider the presence of real data x (classified as 1) and generated data z (classified as 0). The optimal objective for D is to maximize the classification of x as one and minimize the classification of z as 1. This is expressed as $D(x) \approx 1$ and $D(G(z)) \approx 0$, leading to a maximum value of 0. However, if x is misclassified, meaning $D(x) \approx 0$ or $D(G(Z)) \approx 1$, the logarithmic terms $\log\big(D(x)\big) \approx -\infty$ or $\log\Big(1 - D\big(G(Z)\big)\Big) \approx -\infty$ come into play. Consequently, the value function $V(G,D)$ equals -& in these cases, and the learning process of D focuses on continuously enhancing $V(G,D)$. Conversely, the primary goal for G is to maximize the classification of z as one and minimize the classification of x as 0, as described by Equation 2.

$$min_G V(G,D) = max_G (E_{Z\sim} P_Z(Z) \log\left(D(G(Z))\right) \quad (2)$$

Through adversarial training between G and D, the performance of the generator and discriminator gradually improves. The generator can generate more realistic synthetic samples, while the discriminator becomes more accurate in distinguishing between natural and synthetic samples. Ultimately, the generator can produce synthetic samples that resemble actual data, achieving a convincing effect. The architecture of Generative Adversarial Networks (GANs) is prevalent in the literature on adversarial sample generation for malicious software, as observed in [8][9][10][11][12][13]. These studies demonstrate the effectiveness of GANs in adversarial sample generation for malicious software.

The FastText model is essentially an improvement of the bag-of-words model (CBOW) in word2vec, combined with a pre-trained linear classifier such as logistic regression or support vector machines (SVM). This model consists of three components: the input, hidden, and output layers. The sample text is initially transformed into corresponding n-gram feature vectors through the input layer. Subsequently, the input vectors are subjected to average pooling in the hidden layer. Finally, the output layer uses the softmax function to predict the results. In this study, the FastText model is employed to vectorize the API call sequences, essentially accomplished within the input layer of FastText. The primary structure of the input layer is illustrated in **Fig. 1**.

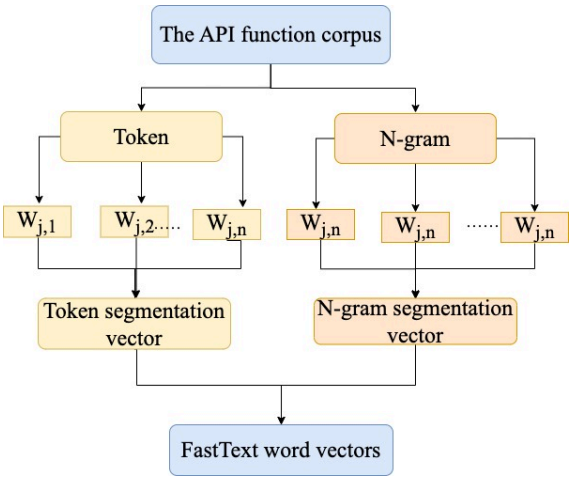

Fig. 1. Input Layer Architecture of the FastText Model

According to **Fig. 1**, the input layer of the FastText model consists of two types of vectors: embedding vectors for each API function in the token dictionary and embedding vectors obtained through n-gram feature extraction. These two vectors are added together to obtain the required word vectors for the model. From the above process, it can be observed that the FastText model incorporates n-gram features in processing API sequences. The objective of the FastText model is to transform maximum likelihood into log-likelihood and minimize this objective function. The computation of the objective function is as follows:

$$LL = -\frac{1}{k}\sum_{k=1}^{k} Y_k \log\left(f(BX_k)\right) \tag{3}$$

In Equation 3, Y_k represents the label value of the k-th malicious sample, f denotes the predicted class using the softmax function, B is a weight matrix, and X_k represents the feature vector of the sample, as expressed by the equation.

$$X_k = A[func(v_1, v_2, \cdots v_{N-1}, v_{N-2})] \tag{4}$$

In Equation 4, *func* represents the averaging function used to compute the feature vector of the input API function sequence. A denotes the weight matrix, V represents the n-gram word vectors in the input sample, and N is the window size for word selection.

The n-gram features consider the influence of both the preceding and succeeding context in the text, effectively capturing the semantic information of the surrounding words during the sliding process. Therefore, n-gram features provide an adequate representation of the text.

3. Related Works

Adversarial sample research refers to the field of spoofing machine learning models and leading to misclassification by making small but intentional perturbations to the input data. In recent years, adversarial sample research has become one of the hotspots in machine learning and deep learning. Researchers have continuously explored the generation methods, defense techniques, and adversarial training of adversarial samples and have made significant progress in computer vision, natural language processing, and other fields. The research on adversarial samples not only helps to reveal the vulnerability of deep learning models but also provides new ideas and challenges for improving the robustness of models. With the deepening of the

research on adversarial samples, people have become more concerned about machine learning models' security and reliability issues.

In this section, we mainly introduce the related research progress in adversarial samples from the authors, method names, essential techniques, and their respective advantages and disadvantages, as shown in **Table 1**.

Table 1. Advances in confrontation sample research

Author	Critical Technologies	Benefit	Shortcoming
Rahul Yumlembam et al	Classifier based on Graph Neural Networks (GNN) [14]	It achieved an accuracy of 98.33% on the CICMaldroid dataset.	It requires a large amount of training data.
	The algorithm is based on the Generative Adversarial Networks (GANs).	It reduced the detection rate of the GNN malicious software classifier.	The robustness of the model limits the generation of adversarial samples.
Xiangjun Li et al	The adversarial sample generation algorithm is based on feature space [15] distribution and feature filtering.	Adopting a multi-feature set detection algorithm improved the robustness of adversarial sample classification detection.	The multi-feature set detection algorithm requires building multiple training sets, which increases the computational burden.
Jianjie Zhang et al	The adversarial malware generation method is based on the concept of N-gram [16].	Inspired by the concept of N-gram in natural language processing, the feature resources have been expanded.	The features are functionally independent of each other, which may affect the original executability of the malicious program.
Fahad Mazaed Alotaibi et al	Conditional Generative Adversarial Networks combined with deep learning feature processing grayscale images [17] and API sequences.	By processing grayscale images and API sequences on a per-pixel basis, a robust representation of the Android Package Kit (APK) files can be obtained.	It consumes computational resources and time, as well as demanding a large amount of data for training and optimization, leading to the phenomenon of overfitting.
Pan Wang et al	The semi-supervised learning method for encrypted traffic classification based on Generative Adversarial Networks [18].	It can achieve fine-grained classification of network traffic, improving network resource utilization.	There may be common issues in GAN models such as overfitting, requiring significant computational resources and time.
Kehong Li et al	Dynamic chaotic crossover optimized bidirectional residual gated recurrent units [19] for feature extraction.	The Generative Feature Disentanglement for Adversarial Attack (GFDA) strategy is proposed to optimize the Wasserstein Generative	There are numerous hyperparameters to adjust, leading to high computational complexity and long training time.

		Adversarial Network (WGAN) and generate pseudo-samples for unseen classes.	
Dong-Ok et al	The malware training framework based on Generative Adversarial Networks (GAN) [20].	It can generate high-quality and diverse images resembling zero-day malware.	The detection capability for completely new and previously unseen malware samples is limited.
Yuanzhang Li, Yaxiao Wang	The Feature Vector-based Generative Adversarial Network (fvGAN) attacks machine learning-based malware classifiers [21].	Adversarial feature vectors were generated in the feature space and transformed into adversarial malware examples, resulting in a high evasion rate of adversarial samples.	Attacks against specific malware classifiers may have limited generalization capabilities.
Shymalagowri Selvaganapathy, Sudha Sadasivam	The feedforward deep neural network model [22] is used to construct feature engineering for malware.	The impact of evasion attacks on Android malware applications is explored in the Drebin dataset to gain insight into the behavioral characteristics of Android malware.	The feedforward deep neural network model is unsuitable for processing large-scale malware datasets.
Tertsegha J et al	Two GAN architectures with data transformation were proposed to train and generate discrete and continuous network traffic features simultaneously.	A good match exists between the logarithm of the mean and standard deviation of fake data [23] and the corresponding quantities in real data.	If there is bias or noise in the training dataset, it may affect the quality of the generated fake data.
Y. Ding et al	An adversarial malware sample generation method based on feature byte sequences [24].	Feature byte sequences can be shared to generate many adversarial samples.	The adversarial sample generation method may lead to the failure of generated malware samples under different environments or defense mechanisms.

Based on the contributions of the researchers above, we have identified issues regarding the poor stability and limited transferability of adversarial samples in malware generation models. The proposed method in this paper effectively captures important semantic information within malicious software API sequences, laying a solid foundation for subsequent adversarial sample generation. Additionally, by incorporating a pixel attention mechanism into the DCGAN network model, we enhance the modeling capability and stability of the model by addressing crucial pixels. As a result, high-quality adversarial samples of malware are successfully generated. In the next section, we will provide a detailed description of the methodology.

4. Methodology

This section introduces a proposed method for generating adversarial malware samples based on the pixel attention mechanism. This method incorporates the pixel attention mechanism into the DCGAN network model, which weights different pixels by anchoring key pixels in grayscale images [25][26][27]. This mechanism enhances the DCGAN model's focus on essential pixel regions, generating high-quality adversarial samples. The design scheme of this method is illustrated in **Fig. 2**.

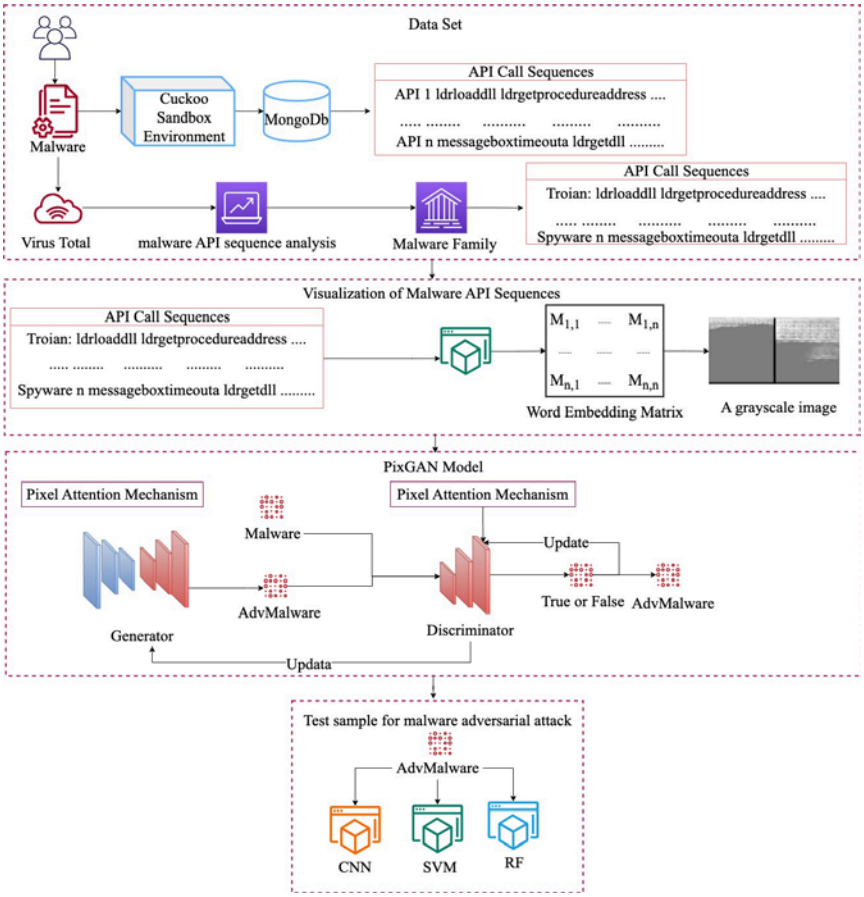

Fig. 2. Flowchart of the Adversarial Malware Sample Generation Method

Based on **Fig. 2**, the proposed method consists of four steps: dataset construction, visualization representation of malicious software, establishing adversarial network models, and quality assessment of adversarial samples. This method effectively generates adversarial samples for malicious software, enhancing the robustness of malicious software detectors.

4.1 Dataset

In this section, we established our dataset consisting of ordered sequences of malicious software API calls [28][29], which were analyzed in a sandbox environment. The detailed process for constructing the malicious software dataset is depicted in **Fig. 3**.

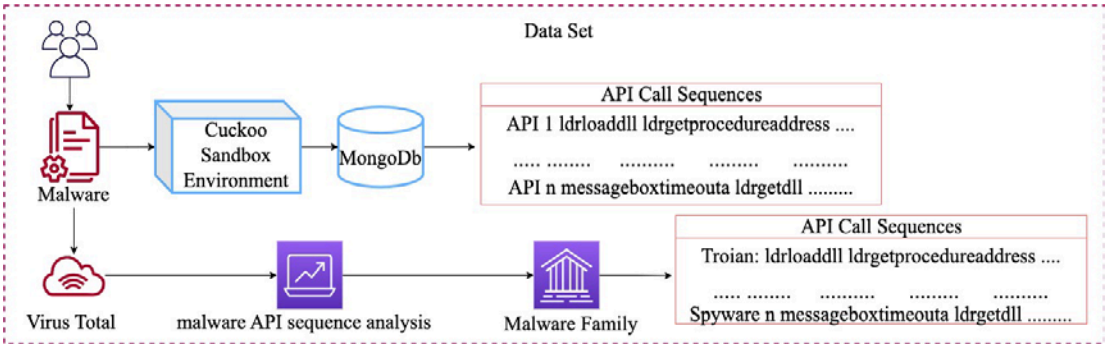

Fig. 3. Process Flowchart for Malicious Software Dataset Construction

As shown in **Fig. 3**, our process for building the dataset consists of four steps. Firstly, a sandbox environment is set up by installing the Ubuntu operating system and Cuckoo sandbox application on a malware analysis machine to prevent any interference or limitations during the malicious software operation. Secondly, the malicious software is sequentially run in the Cuckoo sandbox, which writes the analysis information of each malware into a MongoDB database. By analyzing this information, the behavior dataset of the malicious software on the analysis machine can be obtained, which includes all Windows malicious software API call sequences. The Windows malicious software API call sequences are labeled and filtered to obtain the required sequences. Furthermore, the Virus Total website's API analysis service is utilized to scan each malware using anti-virus applications, providing more complete results for the analysis. Finally, based on the analysis results from the Virus Total website, the malware family name associated with each Windows malicious software API call sequence is determined. The Windows malicious software dataset thus generated includes both the malicious software API call sequence and the associated malware family name.

The established dataset through the steps mentioned earlier consists of ten different sequences of malicious software API calls, with a total of 7,107 samples belonging to eight categories of malware families, namely Spyware, Virus, Backdoor, Downloader, Trojan, Adware, Dropper, and Worms. The dataset partitioning is shown in **Table 2**.

Table 2. Malware Dataset, Quantity, and Labels

Name	Number	Function	Label
Spyware	832	Lurking inside computers, stealing user information.	1
Virus	1001	Self-replicating and infecting normal programs and system files to spread.	2
Backdoor	1001	Exploiting vulnerabilities to gain computer privileges.	3
Downloader	1001	Downloading malicious software from remote servers and disguising it as legitimate software.	4
Trojan	1001	Stealing, deleting, or modifying data.	5
Adware	379	Displaying various advertisements on the computer.	6
Worms	1001	Causing problems such as depletion of computer resources, network congestion, and data loss.	7
Dropper	891	Self-extracting and releasing other malicious programs.	8
Total	7107		

4.2 Visualization of Malware API Sequences

In this chapter, we have detailed a visualization method that combines the semantic relationships of malicious software API call sequences with grayscale images to provide more

comprehensive information for the analysis and detection of malware. **Fig. 4** outlines the overall steps of the API sequence visualization method proposed in this study.

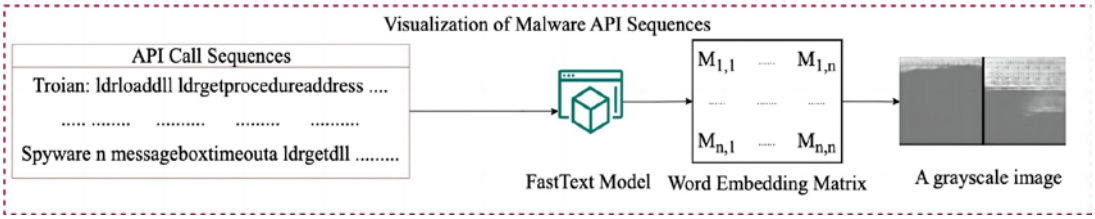

Fig. 4. Framework Diagram of the Method for Converting API Sequences into Grayscale Images

As shown in **Fig. 4**, we need to convert the malicious software into grayscale images of the same size (64*64) for inputting into a generative adversarial network. The specific steps are as follows:

Firstly, we use the FastText model to convert each API call sequence into a word vector matrix. FastText is a text representation method based on word level, which represents each API call as a word and generates the corresponding word vector. In this way, each API call can be represented as a word corresponding to a word vector, and each API call sequence can be represented as a matrix. For the dataset we established, the word vectors generated by the FastText model are shown in **Table 3**, and the generated word vector matrices are shown in **Table 4**.

Table 3. Word Vector

API	**Word Vector**				
Ldrgetprocedureaddress	[-0.3627447	0.00520169	……	-0.06013719	1.2452438]
Findfirstfileexw	[0.5446774	0.62287396	……	-1.125845	0.30359298]
GetModuleHandleA	[-0.3651721	-0.7364854	……	-1.1804125	1.9169759]
TerminateProcess	[0.6624189	1.4515684	……	-1.0373702	-0.6090182]
LoadLibraryA	[0.8548435	-0.03293405	……	-0.493055	0.91679096]
UnhandledExceptionFilter	[1.051844	2.0425346	……	-1.1316224	-0.85219264]
GetLastError	[-0.3451861	-0.04660342	……	0.11660674	0.5597831]
……	……	……	……	……	……

Table 4. Word Vector

Word Vector Matrix				
[-0.24487647	-0.9550668	……	-3.85715389	1.3867563]
[-0.049142	-0.3848626	……	-1.17403603	0.55121046]
[-0.22239758	0.16714095	……	0.10780774	0.11150108]
……	……	……	……	……
[0.67640847	0.6420632	……	-0.21321912	-0.46966267]

Next, we will normalize the generated word vectors to ensure that the values of each dimension are within the range of 0 to 255. Then, we will reshape the normalized word vectors to a size of 64*64 to generate corresponding images. The reshaped word vectors will serve as the pixel intensity values of the image, resulting in a grayscale image. Each pixel's intensity value corresponds to the word vector's value at the respective position. The final grayscale image is shown in **Fig. 5**.

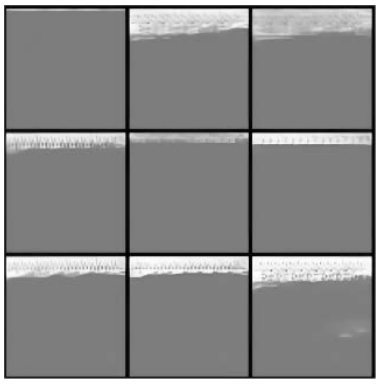

Fig. 5. Grayscale Texture Images of Malicious Software Converted from Different API Sequences

4.3 PixGAN Model

In this section, we present the proposed Pixel-Attention Mechanism-based Generative Adversarial Network model for generating malicious software, which we refer to as PixGAN in this paper. Based on the architecture of the DCGAN network model, we have introduced improvements by incorporating a pixel attention mechanism that is highly sensitive to contrasts. Specifically, the pixel attention mechanism calculates weights. It applies them to feature maps, enabling the generative adversarial network model to capture task-related features better and improve the quality of generated adversarial samples. **Fig. 6** illustrates the overall network architecture of the PixGAN model.

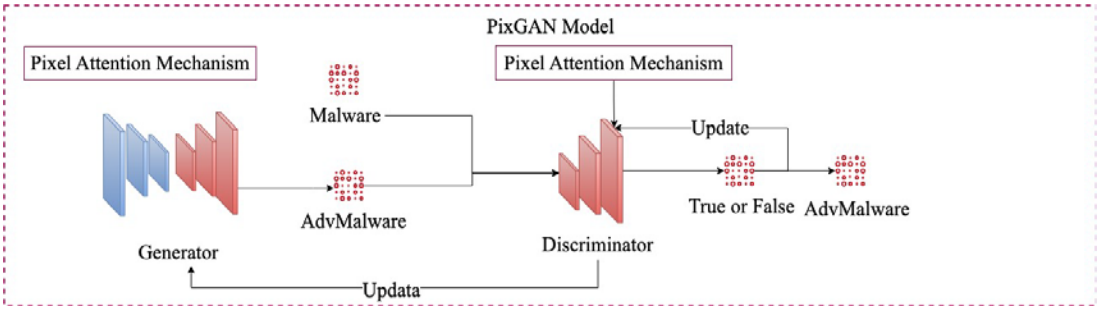

Fig. 6. The Architecture Diagram of the PixGAN Model

(1) Constructing Pixel Attention Mechanism

The pixel attention mechanism consists of convolutional layers, fully connected layers, activation parameter layers, and a Multiply layer, as shown in **Fig. 7**.

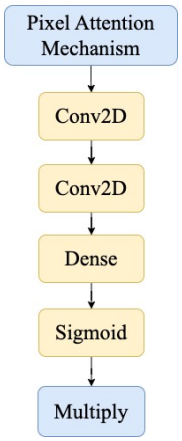

Fig. 7. Structure Diagram of the Pixel Attention Mechanism

As shown in **Fig. 7**, the input layer of this attention mechanism receives grayscale images of size 64*64*1 as input data. The following two convolutional layers are used to extract features from the images. The subsequent fully connected and activation function layers are used to learn attention weights. Finally, the element-wise multiplication layer multiplies the original image with the attention weights element-wise to achieve weighted image blending.

(2) Constructing PixGAN Generator

The PixGAN generator comprises a pixel attention mechanism, deconvolutional layers, batch normalization, and activation function layers. The specific network structure parameters of the generator are shown in **Fig. 8**.

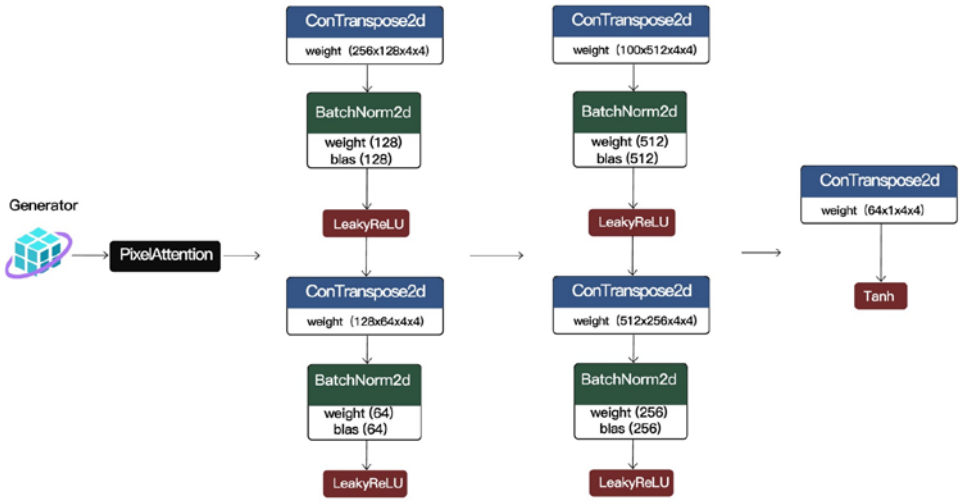

Fig. 8. Generator Network Structure Parameter Diagram

As shown in **Fig. 8**, the PixGAN model's generator accepts grayscale images and random noise as inputs. First, the grayscale image is weighted using the pixel attention mechanism. Then, transposed convolution operation with a kernel size of 4x4 and stride of 2 is used for up sampling the input to generate higher-resolution images. Batch Normalization layers are used for batch normalization operations to accelerate training speed and improve model stability.

The output layer of the generator uses a Conv2DTranspose layer, where the number of output channels is 1, the kernel size is 4x4, the stride is 2x2, and the padding mode is 'same.' The activation function of the output layer is Tanh, which limits the pixel values to [-1, 1] to generate adversarial samples.

(3) Constructing PixGAN Discriminator

The PixGAN discriminator comprises a pixel attention mechanism layer, convolutional layers, activation functions, and a Flatten layer. The specific network structure parameters of the discriminator are shown in **Fig. 9**.

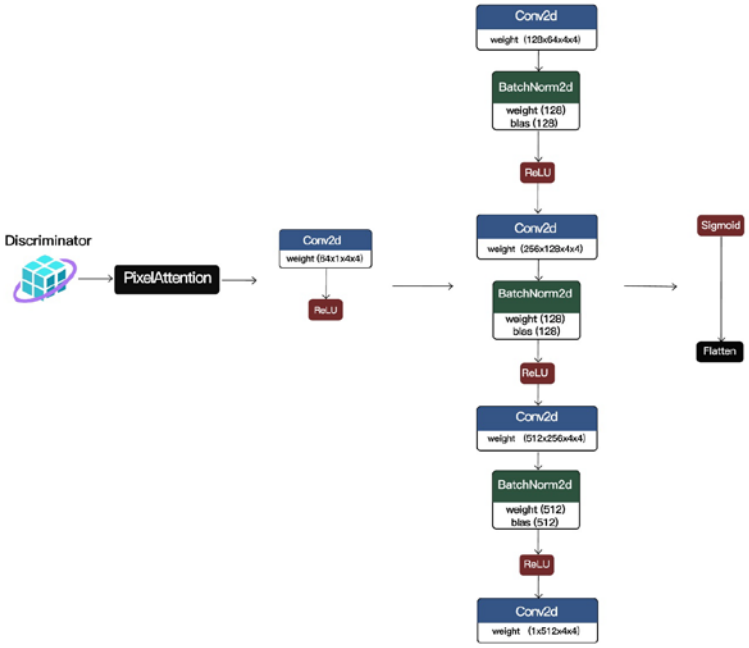

Fig. 9. Discriminator Network Parameter Structure Diagram

As shown in **Fig. 9**, the input to the discriminator is the pixel values of the grayscale image. The pixel attention mechanism module is used for feature processing, and each feature map obtains an attention weight. The attention weights are then used to weight the feature maps, giving more weight to those that are more important for classification, thus improving the performance of the discriminator. The weighted feature maps are then passed through five convolutional layers with a kernel size of 4x4, stride of 2x2, and padding mode of 'same.' Finally, the output passes through activation function layers and a Flatten layer. The activation function used is sigmoid, which limits the output values to the range [0, 1], representing the probability that the sample is accurate. The Flatten layer is used to flatten the multidimensional input into a one-dimensional vector, converting the feature maps output from the convolutional layers into vector form.

4.4 Basic Algorithm Flow

Below is the pseudocode for generating adversarial samples of malicious software using the pixel attention mechanism. As shown in **Table 5**.

Table 5. PixGAN algorithm process

Malware adversarial sample generation model—PixGAN
Input: Original malware sample M, number of iterations epoch, maximum number of iterations Max, learning rate α, noise dimension Z. Output: Adversarial sample M_{adv}. 1. While (epoch < Max) do: 2. Initialize the generator G and the discriminator D with random weights. 3. Define the loss function L_G for the generator and L_D for the discriminator. 4. The generator incorporating the attention mechanism maps the noise Z to a perturbation map. 5. The original malware sample M is combined with the perturbation mask to generate the adversarial sample M_{adv} 6. Feed both the original malware sample M and the adversarial sample M_{adv} into the discriminator. 7. Obtain the cross-entropy loss L_G for the generator and the cross-entropy lossL_D for the discriminator, respectively. 8. Update the weights of the generator O_g by descending along the gradient $\nabla_{O_g} L_G$. 9. Update the weights of the discriminator O_d by descending along the gradient $\nabla_{O_d} L_D$. 10. Return the adversarial sample M_{adv}. 11. End while

5. Experiments

5.1 Experimental Environment

The environment configuration used in this experiment is shown in **Table 6**.

Table 6. Experimental settings

Parameter	Value
OS	Windows 10
CPU	Intel Core i9
GPU	NVIDIA GeForce RTX 3090
RAM	32G
Python	Anaconda/Python3.8.3
Deep Learning Framework	Pytorch

5.2 Training Malware Detector Model

The original dataset is usually divided into training and testing sets when training a neural network model. The training set fits the model by setting the classifier's parameters and training the classifier model. The testing set is used for model prediction and performance evaluation, measuring the model's classification ability.

In this experiment, a scheme was designed using random sampling to randomly partition sample data as the training and testing sets for training PixGAN and malware detector. Based on this, 80% of the data was considered the training set, and 20% was considered the testing set. The epoch of the malware detector is set to 3500 iterations. Through training on malware samples, the stability of the malware detector model is demonstrated in Fig. 10, while the accuracy of the malware detector model is depicted in **Fig. 11**.

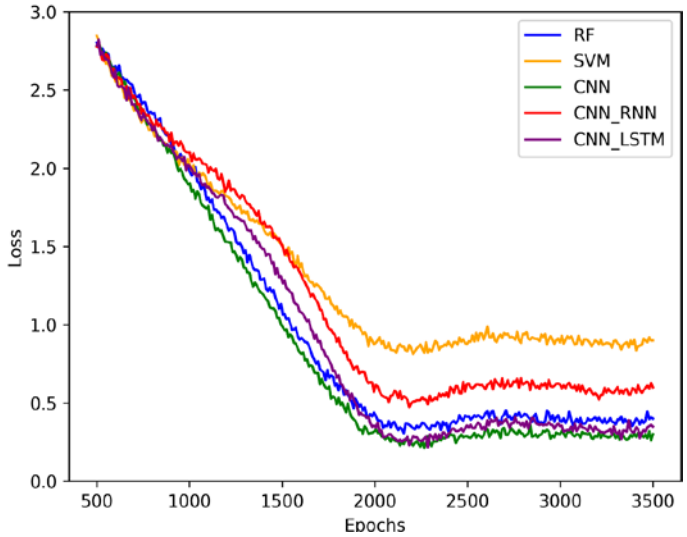

Fig. 10. Illustration of the loss results for the malware detector

According to the analysis from **Fig. 6**, the minimum loss function values for RF, SVM, CNN, CNN_RNN, and CNN_LSTM are 0.4, 0.9, 0.3, 0.6, and 0.35, respectively. After training the malware detection model, it is observed that the loss function exhibits a decreasing trend, and the loss function converges after 2500 iterations. At this point, the performance of the malware detection model reaches its optimum.

According to the analysis of **Fig. 11**, after training the malware detection models based on machine learning and deep learning, the recognition rate of original malware has reached over 90%. The detection model achieved excellent classification results on the training and testing sets of original samples, providing a basis for verifying the quality of adversarial samples in subsequent experiments.

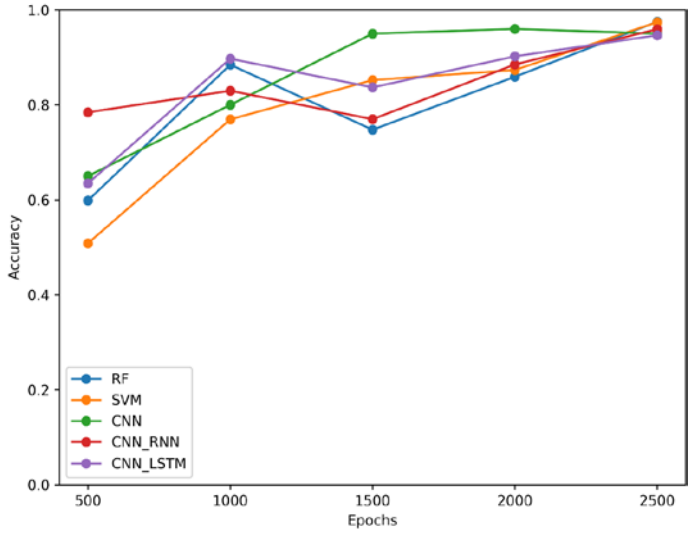

Fig. 10. Illustration of the accuracy results for the malware detector

5.3 Validating Model Performance

This paper used the PixGAN model to generate adversarial malware samples. In the early stages of the experiment, attempts were made to improve the GAN model due to the instability of the GAN algorithm during training. A stable generative adversarial network that generates high-quality samples were constructed using the PixGAN model based on the DCGAN network architecture. This was achieved by introducing a pixel attention mechanism to anchor key texture feature pixels in grayscale images, weight different pixels to enhance focus on important pixel regions and improve the modeling ability of the generator and discriminator.

During the training of the PixGAN model generator and discriminator, the learning rate of the generator and discriminator was set to 0.01, 0.001, and 0.0002 during the experiment. When the learning rate was set to 0.01 and 0.001, it caused instability in the PixGAN model during the training process. The generator parameters were updated too quickly, resulting in divergence of the loss function, and the generator could not generate high-quality images effectively. The discriminator exhibited overfitting, being too sensitive to the subtle differences in the training data while ignoring the proper data feature distribution, resulting in decreased generalization ability.

Through repeated experiments and comparisons, we found that, ultimately, the noise dimension of the generator was set to 100, and the generator's and discriminator's learning rate was set to 0.0002. The number of iterations was set to 12000, and the batch size was set to 64. The generator's activation functions were selected as ReLU and TanH, while the discriminator's activation functions were chosen as Sigmoid and LeakyReLU.

This experiment evaluated the model's performance by examining the relationship between the minimum value of the loss function and the number of iterations. To verify the stability of different generative adversarial models under the same dataset, GAN, DCGAN, MalGAN, and PixGAN models were compared, due to the loss function's non-smoothness, adversarial networks' training process often involves fluctuations. The horizontal axis of the loss function trend chart represents the number of iterations, while the vertical axis represents the value of the loss function.

Fig. 11 presents a comparative illustration of the generator loss function between PixGAN and other models.

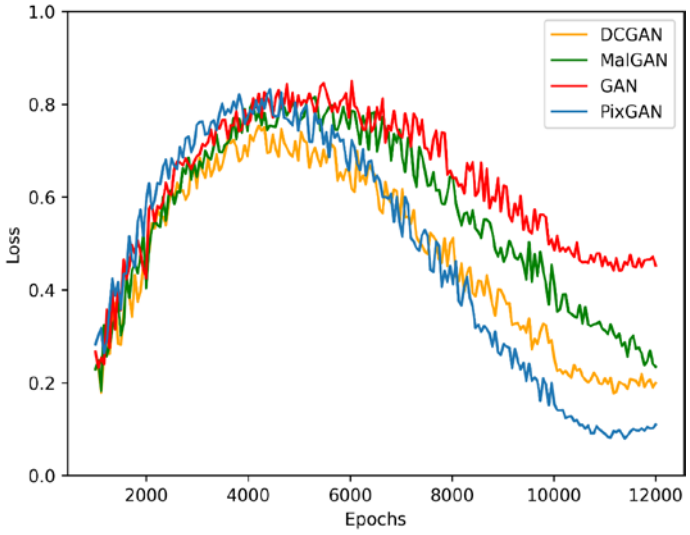

Fig. 11. Trend Chart of the Generator Loss Function

Observing the change in the generator's loss function trend: At the beginning of model training, there was a significant difference between the adversarial and natural samples. The discriminator had a strong performance at the beginning of training, making it difficult for the adversarial samples to confuse the discriminator. As a result, the value of the generator's loss function gradually increased. However, after 5000 iterations, the generator's performance gradually improved, and the value of the loss function decreased. Finally, after iterating 10000 times, the generator model tended to be stable.

Fig. 12 presents a comparative illustration of the discriminator loss function between PixGAN and other models.

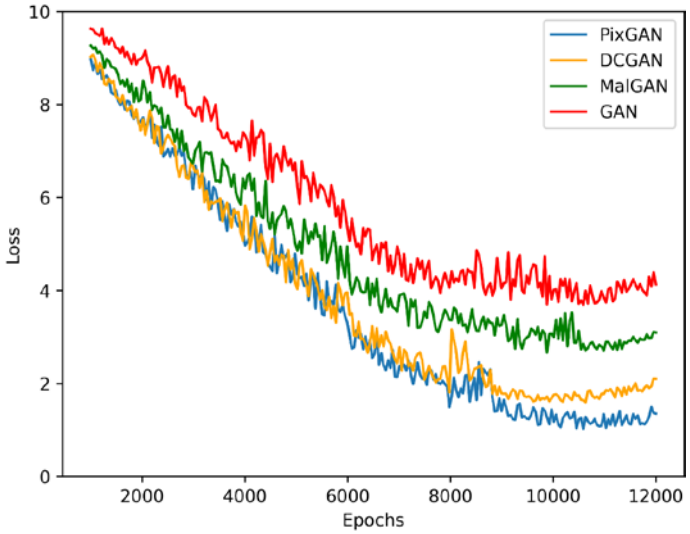

Fig. 12. Trend Chart of the discriminator Loss Function

Observing the change in the discriminator's loss function trend: At the beginning of model training, the discriminator had a strong discrimination ability. With the increase in the number of iterations and the generation of high-quality samples, the value of the discriminator's loss function gradually decreased. Finally, after iterating around 9000 times, the discriminator model tended to be stable.

Once the model reaches stability, the comparative results of the loss function values are presented in **Table 7**.

Table 7. Comparative results for loss function values

Model Name	Loss Function (Generator)	Loss Function (Discriminator)
GAN	0.45	4.2
DCGAN	0.25	2.1
MalGAN	0.35	3.6
PixGAN	**0.15**	**1.8**

According to **Table 7's** results, the PixGAN model achieves a minimum generator loss function value of approximately 0.15 and a minimum discriminator loss function value of approximately 1.8 in its stable state compared to other generative models. It indicates that PixGAN outperforms other generative models in terms of performance.

5.4 Evaluating the Quality of Adversarial Samples

To analyze the quality of the adversarial samples, this study used *ASR* as an evaluation metric. *ASR* is the ratio of adversarial samples that can evade detection to the number of malicious software samples. *NoS* represents the total number of undetected malware samples, and *ToS* represents the total number of malware samples. The expression for *ASR* is:

$$ASR = \frac{NoS}{ToS} \tag{5}$$

A total of 1280 adversarial samples generated by PixGAN were selected for quality validation in this experiment. These adversarial samples were tested on a CNN+LSTM detector, and the evasion rate was used as an indicator to measure the quality of the adversarial samples. A higher evasion rate indicates a higher quality of the adversarial samples. **Table 8** shows the number of evasions for different types of malwares.

Table 8. The number of Escaped Adversarial Samples in the CNN+LSTM Model

Name	Number	Name	Number
Spyware	175/575	Trojan	85/112
Virus	155/255	Adware	0/10
Backdoor	112/212	Worms	0/12
Downloader	24/84	Dropper	0/20
In Total	**551/1280**		

According to the analysis in **Table 8**, Trojans have the highest proportion among the evasive adversarial examples, while DownLoader has a minor proportion. When tested on a CNN+LSTM model detector, the evasion rate reached 43%, indicating that the generated adversarial examples have a good effect on the CNN+LSTM detector.

To verify the transferability of the adversarial examples, the malicious software adversarial examples were tested on four different types of malware detectors based on CNN+RNN, CNN, RF, and SVM, respectively. As shown in **Fig. 13**, the experimental results demonstrated good performance across all detectors.

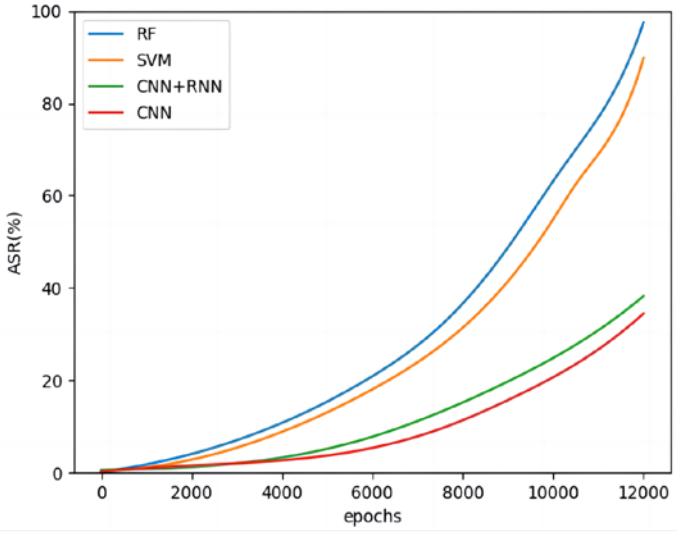

Fig. 13. Illustration of Evasion Effect on Four Types of Malware Detectors

By observing the results, it can be inferred that as the number of iterations increases, the evasion rate of the adversarial examples gradually increases. Specifically, on the RF, SVM, CNN+RNN, and CNN detectors, the evasion rates reached 97%, 94%, 39%, and 35%, respectively.

5.5 Evaluating the Quality of Adversarial Samples

Through experiments, it was discovered that the generative adversarial networks utilized in this study can effectively generate adversarial samples for malware. However, the evasion effectiveness of these generated adversarial samples varies among the models, as illustrated in **Table 9**.

Table 9. Comparison of evasion rates for adversarial samples

Model Name	RF	SVM	CNN	CNN_RNN	CNN_LSTM
GAN	92%	90%	20%	31%	30%
DCGAN	90%	93%	32%	33%	35%
MalGAN	95%	91%	34%	30%	27%
PixGAN	**97%**	**94%**	**35%**	**39%**	**43%**

Through comparison, it was found that the evasion rates of adversarial samples generated by PixGAN for malicious software are superior to other models across five malicious software detectors. It demonstrates that the adversarial samples generated by PixGAN exhibit higher evasion rates and better transferability.

6. Conclusion

The paper proposes a method for generating adversarial samples for malware based on a pixel attention mechanism. This method integrates the idea of pixel attention into the DCGAN model to enhance the modeling ability of its generator and discriminator. Specifically, by comparing the gradient information between the input grayscale image and the generated adversarial samples, we anchor the crucial pixels for the key texture features in the generated adversarial samples. It allows for the weighting of pixels in the grayscale image, thereby generating high-quality adversarial samples. Although our experimental results have shown satisfactory performance, we must acknowledge that the dataset is relatively limited. Therefore, it cannot fully demonstrate the model's generalization ability across different datasets. Additionally, there is room for improvement in the texture effects of the proposed malicious software visualization method. Another challenge is reverting grayscale images representing malicious software back to executable and malicious code samples. Our plans include diversifying the dataset and adjusting the model parameters to enhance its generalization ability. Simultaneously, we will focus on designing a reversible malicious code visualization method capable of generating visually appealing images with distinct texture effects while being able to self-revert to the original malicious code samples. Ultimately, we aim to advance the field of adversarial samples and malware detection through our research.

References

[1] M. Trevisan, F. Scoro, “Attacking DoH and ECH: Does Server Name Encryption Protect Users’ Privacy?,” *ACM Transactions on Internet Technology*, vol. 23, no. 1, pp. 1-22, May. 2023.
Article (CrossRef Link)

[2] A. Benmoussa, N. Lagraa, et al., "Interest Flooding Attacks in Named Data Networking: Survey of Existing Solutions, Open Issues, Requirements, and Future Directions," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1-37, December. 2022. Article (CrossRef Link)
[3] M. Hammad, N. Hewahi, et al., "T-SNERF: A novel high accuracy machine learning approach for Intrusion Detection Systems," *IET Information Security*, vol. 15, no. 2, pp. 178-190, March. 2021. Article (CrossRef Link)
[4] Z. Li, Y. Liu, et. al., "Highly transferable adversarial attack against deep-reinforcement-learning-based frequency," *Energy Conversion and Economics*, vol. 4, no. 3, pp. 202-212, June. 2023. Article (CrossRef Link)
[5] A. Pektaş, T. Acarman, "Malware classification based on API calls and behaviour analysis," *IET Information Security*, vol. 12, no. 2, pp. 107-117, March. 2018. Article (CrossRef Link)
[6] S. Chaudhari, V. Mithal, et al., "An Attentive Survey of Attention Models," *ACM Transactions on Intelligent Systems and Technology*, vol. 12, no. 5, pp. 1-32, October. 2021. Article (CrossRef Link)
[7] Z. Tang, J. Wang, B. Yuan, et al., "Markov-GAN: Markkov image enhancement method for malicious encrypted traffic classification," *IET Information Security*, vol. 16, no. 6, pp. 442-458, November. 2022. Article (CrossRef Link)
[8] J. Yuan, S. zhou, et al., "Black-Box Adversarial Attacks Against Deep Learning Based Malware Binaries Detection with GAN," *Frontiers in Artificial Intelligence and Applications*, vol. 325, pp. 2536-2542, 2020. Article (CrossRef Link)
[9] M. Kazi, S. Woodhead, et al., "An Investigation to Detect Banking Malware Network Communication Traffic Using Machine Learning Techniques," *Journal of Cybersecurity and Privacy*, vol. 3, no. 1, pp. 1-23, 2023. Article (CrossRef Link)
[10] P. Faruki, A. Bharmal, V. Laxmi, "Android Security: A Survey of Issues, Malware Penetration, and Defenses," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 998-1022, 2015. Article (CrossRef Link)
[11] L. Demetrio, B. Biggio, et al., "Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware," *IEEE Transactions on Information Forensics and Security*, vol. 16, no. 2, pp. 3469-3478, May. 2021. Article (CrossRef Link)
[12] D. Li, Q. Li, "Adversarial Deep Ensemble: Evasion Attacks and Defenses for Malware Detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3886-3900, June. 2020. Article (CrossRef Link)
[13] N. Martins, J. Cruz, et al., "Adversarial Machine Learning Applied to Intrusion and Malware Scenarios: A Systematic Review," *IEEE Access*, vol. 8, pp. 35403-35419, February. 2020. Article (CrossRef Link)
[14] R. Yumlembam, B. Issac, et al., "IoT-Based Android Malware Detection Using Graph Neural Network with Adversarial Defense," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8432-8444, May. 2023. Article (CrossRef Link)
[15] X. Li, K. Kong, "Feature selection-based android malware adversarial sample generation and detection method," *IET Iformation Security*, vol. 15, no. 6, pp. 401-416, Nov. 2021. Article (CrossRef Link)
[16] E. Zhu, J. Zhang, et al., "N-gram MalGAN: Evading machine learning detection via feature n-gram," *Digital Communications and Networks*, vol. 8, no. 4, pp. 485-491, Aug. 2022. Article (CrossRef Link)
[17] F. Alotaibi, Fawad, "A Multifaceted Deep Generative Adversarial Networks Model for Mobile Malware Detection," *Applied Sciences*, vol. 12, no. 19, Sep. 2022. Article (CrossRef Link)
[18] P. Wang, Z. Wang, et al., "ByteSGAN: A semi-supervised Generative Adversarial Network for encrypted traffic classification in SDN Edge Gateway," *Computer Work*, vol. 200, no. 9, Dec. 2021. Article (CrossRef Link)
[19] K. Li, W. Ma, et al., "Unbalanced network attack traffic detection based on feature extraction and GFDA-WGAN," *Applied Soft Computing*, vol. 216, Oct. 2022. Article (CrossRef Link)
[20] D. Won; Y. Jang, et al., "PlausMal-GAN: Plausible Malware Training Based on Generative Adversarial Networks for Analogous Zero-Day Malware Detection," *IEEE Transactions on Emerging Topics in Computing*, vol. 11, no. 1, pp. 82-94, Mar. 2023. Article (CrossRef Link)

[21] Y. Li, Y. Wang, "A feature-vector generative adversarial network for evading PDF malware classifiers," *Information Sciences*, vol. 253, pp. 38-48, June. 2020. Article (CrossRef Link)
[22] K. Selvaganapathy, S. Sadasivam, "Anti-malware engines under adversarial attacks," *International Journal of Computers and Applications*, vol. 44, no. 8, pp. 791-804, 2022. Article (CrossRef Link)
[23] J. Anande, S. Leeson, "Generative adversarial networks for network traffic feature generation," *International Journal of Computers and Applications*, vol. 45, no. 4, pp. 297-305, Jul. 2023. Article (CrossRef Link)
[24] Y. Ding, "An Efficient Method for Generating Adversarial Malware Samples," *electronics*, vol. 11, no. 1, pp. 154, 2022. Article (CrossRef Link)
[25] X. Wang, X. Wang, "A Novel Grayscale Image Steganography Scheme Based on Chaos Encryption and Generative Adversarial Networks," *IEEE Access*, vol. 8, pp. 168166-168176, Sep. 2020. Article (CrossRef Link)
[26] C. Bijitha, V. Nath, "On the Effectiveness of Image Processing Based Malware Detection Techniques," *Cybernetics and Systems*, vol. 53, no. 7, pp. 615-640, Jan. 2022. Article (CrossRef Link)
[27] H. Naeem, B. Guo, "A Cross-Platform Malware Variant Classification based on Image Representation," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 7, pp. 3756-3777, Jul. 2019. Article (CrossRef Link)
[28] J. Xu, Y. Li, R. Deng, "SDAC: A Slow-Aging Solution for Android Malware Detection Using Semantic Distance Based API Clustering," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 1149-1163, Apr. 2022. Article (CrossRef Link)
[29] D. Zou, Y. Wu, "IntDroid: Android Malware Detection Based on API Intimacy Analysis," *ACM Transactions on Software Engineering and Methodology*, vol. 30, no. 3, pp. 1-32, May. 2021. Article (CrossRef Link)
[30] M. Tang, Q. Qian, "Dynamic API call sequence visualisation for malware classification," *ACM Transactions on Software Engineering and Methodology*, vol. 13, no. 4, pp. 367-377, Jul. 2019. Article (CrossRef Link)

Xiangyu Ma was born in Dongying, Shandong, China, in 1997.He received a B.S. degree in network engineering from Weifang University in 2020. He obtained the CCIE/RS certification in 2020, participated in multiple scientific research competitions during his university years, and achieved good results. He is currently pursuing an M.S. degree in computer technology at Shenyang Ligong University, Shenyang, China. While pursuing a master's degree, he participated in a network security innovation competition and won third prize at the national level. His research interests include network security, network engineering, deep learning, and malicious code generation techniques.

Yuntao Zhao completed his Ph.D. in Navigation, Guidance, and Control from the Nanjing University of Science and Technology. He is currently a professor and doctoral supervisor at the School of Information Science and Engineering at Shenyang Ligong University. His main research areas are Cyberspace Security, Machine Learning, and Deep Learning Algorithms.

Yongxin Feng received an M.S. in computer science from Northeastern University in 2000 and a Ph.D. in computer science and technology from the School of Information Science and Engineering, Northeastern University, in 2003. She is currently a Professor at Shenyang Ligong University. She has authored over 60 papers in related international conferences and journals. Her research interests include network management, wireless sensor networks, and communication and information systems. She received the ICINIS 2011 Best Paper Awards and 15 Science and Technology Awards, including the National Science and Technology Progress Award and the Youth Science and Technology Awards from the China Ordnance Society.

Yutao Hu received his BS in Computer Science and Technology from the Shenyang University of Technology, China, in 2021. He is working toward a master’s in computer technology at the School of information science and Engineering, Shenyang Ligong University, China. His research interests include network security and artificial intelligence.