A Model for Detecting Malware Adversarial Samples Based on Anomaly Detection Technology

Yubin Ma, Yuxin Ding, Wen Qian

Harbin Institute of Technology (Shenzhen), Shenzhen, China

Abstract

Deep-learning-based malware detection methods have been widely used. Although these models have stronglearning ability and can automatically learn malware features, most of these models are vulnerable to adversarial samples.In this paper, we propose a malware adversarial samples detection model to solve this issue. The model uses the anomalydetection techniques to detect malware adversarial samples. Tobetter represent the features of PE files, we represent an PEfile as an RGB image and a one-dimensional byte sequence respectively. We design a generation model to extract datafeatures and reconstruct the original sample. The generation model includes two different encoders, one encoder extracts the one-dimensional feature of the PE file, and the other encoder extracts the two-dimensional features of the PE file. The extracted one-dimensional and two-dimensional features are fused as the input of the decoder. The decoder is responsiblefor reconstructing the input. In the training phase, we only provide benign PE files as the training data, which makes the encoder only well fit benign samples. Therefore, malware adversarial samples have larger reconstruction loss than benign PE files. In this way, adversarial samples can be detected. We conduct adversarial attacks against the existing malware classifier MalConv, and construct four types of adversarialsample datasets. The proposed model gets high accuracy for detecting adversarial samples.

Keywords

Anomaly detection, Adversarial samples, Mal-ware adversarial samples detection

1. Introduction

While the Internet brings great convenience to information transmission and sharing, it also intensifies the widespread spread of malware. Currently malware has seriously threat- ened the security of Internet. Malware has many variants, and updates quickly, which makes malware detection tech-nology face serious challenges. With the rapid development of deep learning technologies, deep learning-based malware detection models have been proposed and achieved high de- tection accuracy. However adversarial samples can evade thedetection of deep learning models, which poses a potential threat to the security of deep learning models.

To recognize adversarial samples, two categories of ad- versarial sample defense methods are proposed. The first category is the robust defense method, which improves the robustness of the classifier to defend against adversarialsamples. The second category is the detection method, whichuses the detection algorithm to detect adversarial examples mixed with normal samples.

Most of the adversarial sample defense methods in the malware detection field are robust defense methods, suchas adversarial training, model distillation, random feature failure, and integrated classifier. Adversarial training is to add adversarial samples generated by the adversarial sample generation algorithm into the training dataset and retrain the classifier and thus improve the robustness of the classifier. Model distillation defends against adversarial samples by improving the generalization performance of small networks.Random feature failure randomly masks some features ofthe input to defend against some adversarial sample attack algorithms. Integrated classifier uses multiple classifiers

AHPCAI2022@2nd International Conference on Algorithms, High Performance Computing and Artificial Intelligence
EMAIL: Corresponding author, Email: yxding@hit.edu.cn (YuXin Ding)

to learn malware features and then integrates the decisions of multiple classifiers to identify malware.

The difficulty for detecting malware adversarial samplesis that attackers can design different attacking methods to generate adversarial samples, it is impossible to know all of them, therefore it is very hard to train a machine learning model that can detect all kinds of adversarial samples. The similar problem also exists in the robust defense methods, only the known adversarial samples can be added into the training set to retrain a classifier. The retrained classifier stillcannot detect the unknown adversarial samples.

To solve this issue, an abnormal detection model is pro- posed to detect adversarial samples. the anomaly detection model consists of two parts, one is an asymmetric generationmode, which includes two encoders and one decoder. The data set for training the generation model only includesbenign samples. The second part is the detection model. Thismodel evaluates the similarity between the generated sampleand the original sample. If the generated sample has a big difference from the original sample. The original sample is recognized as an adversarial sample. We conduct adversarialattacks against the deep learning detection model MalConv [1], and construct four types of adversarial samples. Ex- periments show that the proposed model can achieve high detection accuracy for detecting adversarial samples. The contributions we have made are as follows.

- To the best of our knowledge, we are the first to apply anomaly detection to recognize malware adversarialexamples.
- Our model is one class classification model, which onlytrained using benign files, therefore, compared withother machine learning based method, our model has better generalization ability to recognize different typesof adversarial samples, including unseen samples.
- To evaluate the generalization ability of our method,we create an evaluation dataset. We adopt different methods to generate byte perturbations and try differentpositions to insert perturbated bytes. This dataset can beused as benchmark dataset to evaluate the performance of adversarial sample detecting methods.

2. Related Work

Our study mainly involves two research fields. One is malware adversarial attack methods and the other is malwareadversarial defense methods. In this section we introduce theresearch advances in these two areas, respectively.

2.1 Malware Adversarial Attack Methods

The adversarial attack algorithms in the malware domain are different from these in the computer vision domain. Eachbyte in a malware sample has a specified meaning, there- fore the generated sample should have the same functions and semantics as the original sample after being modifiedby the adversarial attack algorithms. Most of the existing adversarial attack algorithms in the malware domain aregradient-based algorithms, where perturbation is obtainedby optimizing a distance metric between the original andthe perturbed sample. To generate adversarial samples for MalConv model [1](a deep learning-based malware detec- tion model), Kolosnjaji [2] et al. firstly added random bytes to the tail of a malware sample and then iteratively update these bytes using a gradient algorithm, and only one byteis modified in each iteration. The experiments show more than 60% adversarial samples can evade the classifiers. Suciu [3] et al. proposed an enhanced attack on MalConv [1] using iterative FGM, which generates perturbations in the embedding space, and then finds the nearest neighbor bytes to the modified embedding representation by traversing the bytes in the computed embedding matrix, then modifies the current byte to be the nearest neighbor byte. In additionto the gradient-based attack model, Chen [4] et al. applied the feature visualization method Grad cam [5] to extract features of benign files important for MalConv [1] classifier,then added the extracted features to the tail of the malware samples to generate the adversarial samples. They also combined the FGSM algorithm to enhanced benign feature attack (BFA) to increase the success rate of attack. We also use the above adversarial attack method to build our test dataset.

2.2 Malware Adversarial Defense Methods

There are the malware adversarial attack algorithms, and accordingly, there are malware adversarial defense meth- ods. For DNN-based malware detectors, Wang [6] et al.used the random feature failure method to defend against attack algorithms. Random feature failure defense againstthe attack by randomly deleting or masking the featuresof the input, and the disadvantage of this method is thatthe accuracy of malware detection is low. Grosse [7] etal. proposed two defenses, namely defensive distillation and adversarial training, to enhance the robustness of the DNN- based malware detectors. Modifying the structure of classifier can also defenses against the adversarial attack, e.g., using integrated classifiers or using model distillation. Smutz [8] et al. used the integrated classifier containingmultiple basic classifiers to defense against the attack. The integrated classifier votes on the results returned by basic classifiers to make a decision. Also similar to integrated classification, Biggio [9] et al. proposed a one-and-a-half class classifier, specifically, the authors firstly combined a two-class classifier with a one-class classifier and then com-bined them using another one-class classifier. In additional, other researchers also used random subspaces and bagging techniques to enhance SVM-based malware detectors, whichare called as Multi- Classifier System SVM (MCS-SVM).

For windows malware, Dujaili [10] et al. proposed the maximum minimization adversarial training, which is used to enhance DNN-based detectors. In the defense method,the inner layer is optimized to generate hostile files by maximizing the loss function of the classifier, and the outer layer optimizes the parameters of the DNN to minimize the loss of the classifier for hostile classification. Li [11] et al. used variational self-encoder and multilayer perceptron to detect malware and combined their detection results to detectmalware and defend against the adversarial attacks.

3. Proposed Model

The proposed detection model is an unsupervised one- class classification model based on anomaly detection tech- nology. The input data of this model are benign PE files.By learning the features of benign PE files, it has lower re- construction error for reconstructing benign samples. When reconstructing adversarial samples, a higher reconstruction error will be generated. Therefore, by evaluating the simi- larity between the original sample and the generated (recon- structed) sample, adversarial samples can be detected. Here, we describe how the model detects malware adversarial samples. Figure 1 shows the overview architecture of the abnormal detection, which consists of three stages.

- Stage1: Data processing. All PE files are represented intwo forms, one dimensional sequential data (1D) and two dimensional RGB image data (2D).
- Stage2: Data Reconstruction. In this stage, we train twoencoders and one decoder, the Enc_1 extracts features from the 1D byte sequences, and the Enc_2 extracts the features from the 2D RGB data. We fuse the extracted 1D features and 2D features as the input of decoder. Then the Dec_1 decodes the fused input to get the reconstructed output data.
- Stage3: Adversarial Sample Detection. A testing sampleis input to the encoders, and the decoder generates the reconstructed sample. By evaluating the reconstructed loss, we can decide if the testing sample is a malware adversarial sample.

3.1 Data Processing

a) Convert PE file to Two-dimensional image: PE filesare portable and executable files in Windows OS, a PEfile mainly includes DOS header, NT header, section table and specific sections. PE files have different size, and their

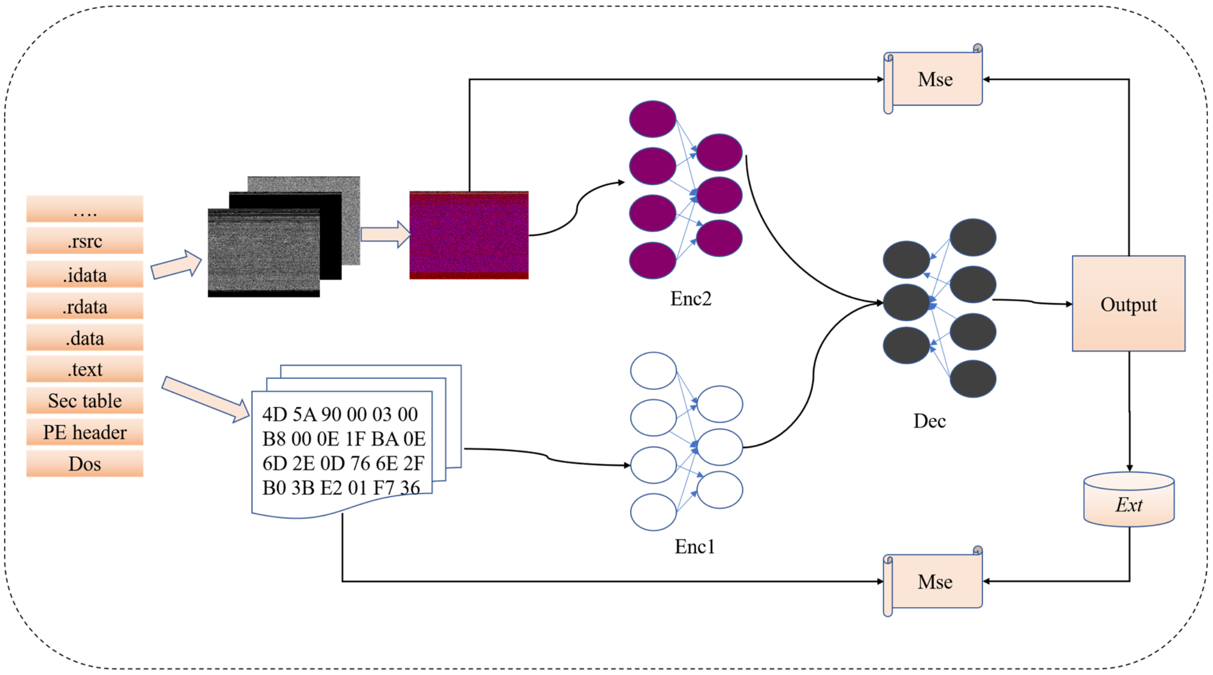

Fig. 1. Overview of the abnormal detection model.

distribution is not uniform. It is impossible to use the entire PE file as the input of the model. Therefore, we need to process PE files to better learn the features of PE files. In order to learn the features of benign PE files well, we extractthe bytes in each section in a PE file. Kancherla [14] et al. represented PE files into gray-scale images, but the size of PE files is large, it is unable to extract all bytes in a PE fileto construct an image, some sections in a PE file have tobe ignored, such as the. rsrc segment, which is at the endof the PE files, its information is often discarded. To fully represent a PE file, we represent PE files as RGB images. We extract bytes from each section as the data of channelsof an RGB image. In details, the data of R channel is thefirst K bytes of the code section .txt, the data of G channelis the first K bytes of the data sections, including .rdata,.idata, .edata, .data, and the data of B channel is the first K bytes of the other parts of a PE file. If there are not enough bytes, padding 0 byte at the end of each channel. The bytes in each channel are expanded into a two-dimensional image and then fuse into an RGB image.

b) Convert a PE file to a one-dimensional byte sequence: A PE file can be seen as a binary stream. We merge every 8 bits into one byte, and the value of each byte isfrom 0 to 255. we connect these bytes one by one to get the one-dimensional data representation of a PE file. Usually, the size of PE files is large, we cannot analysis the whole file. In our work, we extract the channels of the above RGB image, and connect each channel one by one to obtain the one-dimensional byte sequence used to describe a PE file.

3.2 Data Reconstruction

We construct a generation model to construct the input data. The generation model is an asymmetric autoencoder, which includes two encoders and one decoder. The firstencoder Enc_1 encodes the one-dimensional byte sequence to get the 1D feature vector of a PE file and the second encoderencodes the 2D image to get the 2D feature vector of a PE file. We make the dimension of 1D feature vector encoded by Enc_1 the same as that of the 2D feature vector encodedby Enc_2. Then, we connect these two feature vectors asthe input of the decoder *Dec*. Then we use the decoder to reconstruct the original input. The size of the reconstructed output has the same dimension as the RGB image. So wecan calculate the mean squared error (Mse) between theoriginal 2D image and the reconstructed output to evaluate the similarity between them. We also extract the RGB channels form the reconstructed image, and get the one- dimensional byte sequence which has the same dimensionas the original one-dimensional byte sequence. In the same way we can calculate the mean squared error between the original 1D sequence and the reconstructed sequence. The total loss function is shown as Eq(1).

$$l_{MSE} = ||x_{d_2} - [Dec(Enc_1(x_{d_1}) + Enc_2(x_{d_2}))]|| + ||x_{d2} - Ext(Dec(Enc_1(x_{d1}) + Enc_2(x_{d2})))|| \quad (1)$$

In Eq(1), X denotes the set of original input samples. Ext means to extend two-dimentional image to one-dimentional sequence. X_{d1} denotes the set of one-dimensional byte sequence for PE files, X_{d2} denotes the set of two-dimensional RGB images for PE files, Enc_1 denotes the encoder function that encodes the 1D sequence into a feature vector in the latent space, and Enc_2 denotes the encoder function encodes the 2D image into a feature vector in the latent space. Dec denotes the decoder function that converts the feature vectors in the latent space into the original input data. In ourwork, the structure of Enc_1 contains seven one dimensional convolution layers. The active function in each layer is the Leakly relu function. The structure of Enc_2 contains six two dimentional convolution layers and we also use Leakly relu function as the activation function. In Dec, we use six two dimensional deconvolution layers and the active functionin each layer is the Leakly relu function. We calculate the total loss using Eq(1), and then use the gradient descent algorithm to train the encoders and decoder. The training process is shown in Algorithm 1.

Algorithm 1 Training the generation model

Require: Training set of benign PE files X, number of iterations N, length of extracted segments K.

Ensure: Models: Enc_1 for extracting one-dimensional fea- tures, Enc_2 for extracting two-dimensional features, Dec for decoder.

```
1:  function TRAINING($x$,$K$,$N$)
2:      for $i = 1$   $N$ do
3:          for $x$ in $X$ do
4:              $x_{d_1}$ ← PRE_PRO ONE($x$,$K$)
5:              $x_{d_2}$ ← PRE_PRO TWO($x$,$K$)
6:              $encres_1$ ← $Enc_1(x_{d_1})$
7:              $encres_2$ ← $Enc_2(x_{d_2})$
8:              $decr$ ← $Dec(encres_1, encres_2)$
9:              $loss_{encdec}$   $Msel(decr, x_{d_1})$
10:             + $Msel(decr, x_{d_2})$
11:             $Backpropogate_{loss_{encdec}}$ to change $Enc_1$,
12:             $Enc_2$,$Dec$
13:         end for
14:     end for
15:     return $Enc_1$,$Enc_2$,$Dec$
16: end function
```

3.3 Abnormal Detection

We only use the benign file to train the abnormal detectionmodel. Therefore, if the testing sample is a benign sample, the mean square error between the reconstructed sample andthe benign sample is lower, otherwise the mean squarederror is high. According this, we can detect the adversarial sample. In the detection phase, a testing sample is inputto the generation model. The encoder outputs a generated sample. Then we calculate the mean squared error between the generated sample and the testing sample. If the mean squared error is greater than a threshold value. The testing sample is classified as an adversarial sample.

4. Experiment

In this section, we mainly make three experiments. The first experiment is to decide the input length of the genera- tion model. The second experiment we make is to compare the performance of different malware adversarial sample detection models. The third experiment is the ablation ex- periment, which prove fusing different features can improvethe performance of the abnormal detection model. Before conducting the experiments, we constructed four different types of datasets based on different malware adversarialsample generation algorithms.

4.1 Selecting Perturbation Locations for Adversarial samples

- When a PE file is loaded from disk into memory, it takes up more virtual address space than it does on the hard disk. This is because the sections in eachPE file are contiguous on disk, while in memory theyare aligned by page, so there are some gaps between sections after being loaded. Adding random scrambled bytes in these gaps will not affect the functions of thePE file. The parameter *PointerToRawData* in the section table of each PE file specifies the offset of thecurrent section on disk, *V irtualSize* is the total size loaded in memory, *SizeOf RawData* is the size ofthe section on disk, and *V irtualAddress* is the offset address in memory. The actual size occupied when loaded into memory is smaller than the size occupiedon disk, so we can get this gap interval and find the lo-cation where we can add scrambled bytes by indexing. Adding a scrambling between the start and end locationis not going to affect the malicious functionality of the malware. Figure 2 shows the mapping of PE files on disk to memory.

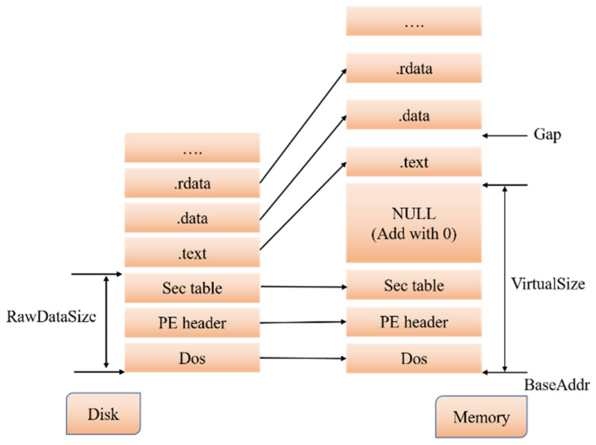

Fig. 2. PE file on disk corresponding to the memory

- Besides the gaps between sections, we can add new sections in a PE file. By modifying the parameter valuesof the section table in the table header, we can add arbitrarily named new sections to a PE file. Since the codes in other sections of the PE file does not callthe codes in the newly added sections, this inserting method also does not affect the functionality of the original PE files. According to the structure of the PE file, we can get the value of *NumberOf Sections*in the PE file header, which is the number of sec-tions. The value of *FileAlignment* in the PE optional header, which is the amount of alignment of the PE file on the disk. The value of *SectionAlignment*, which is the amount of alignment of the PE file in memory. And then we calculate the size of the real new section to be added by initializing the inser-tion size value and the amount of alignment in the disk. Meanwhile, we calculate the size of the last section in disk based on the *PointerToRawData*, *SizeOf RawData* values and *FileAlignment* of the last section. And we calculate the size of the last section in memory based on the *V irtualAddress* and *MiscV irtualSize* of the last section. We create a space of size *SIZEOFSECTIONHEADER* in the section table of the PE file and fill it with the data obtained above in the corresponding location of the new section table. Finally, we find the new section startoffset value and the size of the section to be filled, andset all the byte values of added section to 0x00. At this point, the new section is added to the end of the PE files. In the papers of Kolosnjaji [3] and Chen [4],their methods add bytes directly at the end of PE files,these methods have a slight defect, their methods just read the start and end position of each section in the header of PE file, and get the length of the whole PEfile. So we can avoid reading the scrambled bytes addeddirectly at the end. In this paper, we use the section gapof PE files and create a new section at the tail of PE files as the scrambled position.

4.2 Selection of Model Parameters

In the proposed model, we need to extract the first Kbytes from each type of sections in the PE file. In the experiments, we choose K as 2 x10^4, 10 x10^4, 15 x10^4,20 x10^4, 25 x10^4 respectively. The difference d between thereconstructed sample and the original sample is calculated byEq.(1), which is a floating-point number. Since the training set we use only contains benign samples, we can get a mean square loss value for each sample after encoding and decoding. We average the mean square loss values of all train set samples to get the threshold. If the output resultof the data in the test set through the model is greater than the threshold, we will determine it as an abnormalsample, otherwise it will be determined as a normal sample. During the training process, we use 15840 benign PE files as the training data and they vary in length from 3KB to 60MB. The experimental results are shown in Table 1. In Table 1, SinAD+Gap, SinAD+NS, IFGM+NS and BFA+NSrepresent four adversarial datasets. SinAD, IFGM and BFA represent the algorithms used to generate the adversarialsamples. SinAD is the single-byte modified adversarial sam-ples generation algorithm [2], IFGM is the iterative FGM algorithm [3], BFA is the benign feature based algorithm [4].Gap and NS represent the methods for inserting perturbated bytes. Gap means the perturbated bytes are inserted into the gaps between sections in a PE file, and NS means inserting perturbated bytes into newly created sections in a PE file. In the real scenario adversarial samples are far less than benign samples, so we set the ratio of adversarial samples to benignsamples in the test dataset to be 1:10. We prepare four testingdatasets, and each testing dataset includes one adversarial dataset, 150 adversarial samples and 1500 randomly selectedbenign samples.

From Table 1 we can see that the highest AUC values are obtained on four testing datasets, which means the overall performance of the detector for K=2x10^4 is better than others. So in the following experiments, K is set as 2x$10^4$for each abnormal detection model.

TABLE 1
EXPERIMENTAL RESULTS UNDER DIFFERENT K VALUES

K	Metric	SinAD+GAP	SinAD+NS	IFGM+NS	BFA+NS
	Acc	0.788	0.770	0.770	0.809
	Pre	0.901	0.888	0.887	0.938
2W	Recall	0.833	0.827	0.827	0.828
	F1	0.866	0.856	0.856	0.879
	Roc auc	**0.702**	**0.679**	**0.683**	**0.764**
	Acc	0.889	0.882	0.865	0.911
	Pre	0.889	0.883	0.903	0.913
10W	Recall	0.969	0.969	0.923	0.968
	F1	0.927	0.924	0.913	0.939
	Roc auc	**0.529**	**0.489**	**0.606**	**0.477**
	Acc	0.849	0.824	0.810	0.782
	Pre	0.892	0.888	0.899	0.912
15W	Recall	0.916	0.892	0.863	0.823
	F1	0.904	0.890	0.880	0.865
	Roc auc	**0.681**	**0.639**	**0.638**	**0.592**
	Acc	0.895	0.899	0.916	0.932
	Pre	0.900	0.911	0.885	0.954
20W	Recall	0.950	0.940	0.947	0.934
	F1	0.924	0.931	0.909	0.929
	Roc auc	**0.867**	**0.842**	**0.925**	**0.896**

	Acc	0.838	0.832	0.828	0.832
	Pre	0.883	0.879	0.900	0.910
25W	Recall	0.915	0.913	0.882	0.872
	F1	0.898	0.895	0.891	0.895
	Roc auc	**0.665**	**0.632**	**0.586**	**0.458**

4.3 Comparison With Other Anomaly Detection Algorithms

In this section we compare the proposed model with two classical anomaly detection algorithms, LOF [12] and DeepSVDD [13]. As there is no anomaly detection algorithmto be used for detecting adversarial samples, we reproduce the two algorithm and apply them to detect adversarial samples. LOF is an anomaly detection algorithm based on domain density, and is widely used in the field of computer vision and the DeepSVDD is a deep learning-based anomalydetection algorithm. The results of the comparison experi- ments are shown in Table 2.

From Table 2, it can be seen that the LOF algorithmhas the lowest AUC value. The reason is that LOF is less effective for high-dimensional data classification. Our method is significantly better than the other two models. Compared with DeepSVDD, the structure of our model is flexible, in our model the decoder and encoder are separated,so we can easily increase new encoders to learn more useful data features.

4.4 Ablation Study

There are three modules in our model. To evaluate the influence of each module on the model performance, we

TABLE 2
COMPARISON EXPERIMENT WITH OTHER ANOMALY DETECTION ALGORITHM

Method	Metric	SinAD+GAP	SinAD+NS	IFGM+NS	BFA+NS
	Acc	0.559	0.569	0.487	0.519
	Pre	0.827	0.759	0.642	0.948
LOF	Recall	0.518	0.518	0.518	0.518
	F1	0.637	0.615	0.573	0.670
	Roc auc	**0.599**	**0.594**	**0.473**	**0.525**
	Acc	0.906	0.912	0.928	0.918
	Pre	0.957	0.964	0.981	0.974
DeepSVDD	Recall	0.939	0.939	0.939	0.939
	F1	0.948	0.951	0.959	0.956
	Roc auc	**0.759**	**0.794**	**0.879**	**0.764**
	Acc	0.895	0.899	0.916	0.932
	Pre	0.900	0.911	0.885	0.954
Ours	Recall	0.950	0.940	0.947	0.934
	F1	0.924	0.931	0.909	0.929
	Roc auc	**0.867**	**0.842**	**0.925**	**0.896**

conduct ablation experiments. We consider three scenarios. The first scenario is that we don't use the *Enc*$_1$ to extract1D features of PE files. The second scenario is that we don'tuse the *Enc*$_2$ to extract 2D features of PE files. And the last scenario is that we use all modules for training and testing. We also use these four test sets, and the values of AUC for the ablation experiments are shown in Table 3.

TABLE 3

ABLATION EXPERIMENT COMPARISON RESULT

Ablation	Metric	SinAD+GAP	SinAD+NS	IFGM+NS	BFA+NS
	Acc	0.880	0.800	0.918	0.107
	Pre	0.907	0.898	0.885	0.870
No Enc1	Recall	0.935	0.851	0.959	0.838
	F1	0.921	0.874	0.920	0.123
	Roc auc	**0.632**	**0.570**	**0.909**	**0.450**
	Acc	0.820	0.810	0.872	0.814
	Pre	0.892	0.882	0.863	0.913
No Enc2	Recall	0.881	0.882	0.887	0.836
	F1	0.886	0.882	0.875	0.873
	Roc auc	**0.707**	**0.679**	**0.872**	**0.751**
	Acc	0.895	0.899	0.916	0.932
	Pre	0.900	0.911	0.885	0.954
ALL	Recall	0.950	0.940	0.947	0.934
	F1	0.924	0.931	0.909	0.929
	Roc auc	**0.967**	**0.842**	**0.925**	**0.896**

From the results of the ablation experiments, deletingEnc_1 or Enc_2 all leads the decrease of the overall perforam- nce. The lack of Enc_1 has a greater impact on the BFA+NS dataset. However, regardless of removing any encoder, the overall detection performance on IFGM+NS dataset does not change much. It can be seen that on most datasets, the 1D feature has greater influence on the reconstructed data than the 2D feature. Overall, all three modules have a positive impact on the final classification performance, and none are indispensable.

5. Conclusion

We propose an anomaly detection model to detect mal- ware adversarial samples. The model is trained by learning the features of benign samples and treats all non-benignsamples as anomalous data. To better learn data features, we represent benign samples as binary files and 2D image files respectively, and design two encoders to learn both 1D and 2D features. In the testing phase, we detect adversarial sample according to the similarity between the reconstructedsample and the original sample. The experiments show that the proposed model can effectively detect malware adversarial samples mixed in benign samples.

6. Acknowledgement

This work was supported by the National Natural Sci- ence Foundation of China (Grant No. 61872107) and the Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (2022B1212010005).

7. References

[1] Raff E, Barker J, Sylvester J, et al. Malware detection by eating a whole exe[C]//Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[2] Kolosnjaji B, Demontis A, Biggio B, et al. Adversarial malware binaries: Evading deep learning for malware detection in executa- bles[C]//2018 26th European signal processing conference (EU-SIPCO). IEEE, 2018: 533-537.

[3] Suciu O, Coull S E, Johns J. Exploring adversarial samples in malwaredetection[C]//2019 IEEE Security and Privacy Workshops (SPW).IEEE, 2019: 8-14.

[4] Chen B, Ren Z, Yu C, et al. Adversarial samples for cnn-basedmalware detectors[J]. IEEE Access, 2019, 7: 54360-54371.
[5] Selvaraju R R, Cogswell M, Das A, et al. Grad-cam: Vi- sual explanations from deep networks via gradient-based localiza- tion[C]//Proceedings of the IEEE international conference on com- puter vision. 2017: 618-626.
[6] Wang Q, Guo W, Zhang K, et al. Adversary resistant deep neural networks with an application to malware detection[C]//Proceedings of the 23rd ACM sigkdd international conference on knowledge discovery and data mining. 2017: 1145-1153.
[7] Grosse K, Papernot N, Manoharan P, et al. Adversarial samples for malware detection[C]//European symposium on research in computer security. Springer, Cham, 2017: 62-79.
[8] Smutz C, Stavrou A. When a Tree Falls: Using Diversity in Ensemble Classifiers to Identify Evasion in Malware Detectors[C]//NDSS. 2016.
[9] Biggio B, Corona I, He Z M, et al. One-and-a-half-class multiple classifier systems for secure learning against evasion attacks at test time[C]//International Workshop on Multiple Classifier Systems. Springer, Cham, 2015: 168-180.
[10] Al-Dujaili A, Huang A, Hemberg E, et al. Adversarial deep learn-ing for robust detection of binary encoded malware[C]//2018 IEEE Security and Privacy Workshops (SPW). IEEE, 2018: 76-82.
[11] Li H, Zhou S, Yuan W, et al. Robust Android Malware Detection against Adversarial Example Attacks[C]//Proceedings of the Web Conference 2021. 2021: 3603-3612.
[12] Breunig M M, Kriegel H P, Ng R T, et al. LOF: identifying density- based local outliers[C]//Proceedings of the 2000 ACM SIGMOD international conference on Management of data. 2000: 93-104.
[13] Sandra K, Lee S H. BM3D and Deep Image Prior based Denoising for the Defense against Adversarial Attacks on Malware Detection Networks[J]. International journal of advanced smart convergence, 2021, 10(3): 163-171.
[14] Zhang Y, Li H, Zheng Y, et al. Enhanced DNNs for malware classifi- cation with GAN-based adversarial training[J]. Journal of Computer Virology and Hacking Techniques, 2021, 17(2): 153-163.