An Adversarial Machine Learning Method Based on OpCode N-grams Feature in Malware Detection

Xiang Li
National Key Laboratory of Science and Technology on Information System Security
Bejing, China

Kefan Qiu
Nankai University
Tianjin, China

Cheng Qian
National Key Laboratory of Science and Technology on Information System Security
Beijing,China

Gang Zhao§
National Key Laboratory of Science and Technology on Information System Security
Beijing,China

***Abstract*—Machine learning has become an important method in malware detection. However, due to the weakness of machine learning models, a large number of researches related to adversarial machine learning has emerged. At present, the researches about adversarial machine learning mainly focus on the image and speech recognition. In the field of malware detection, because the feature modification can easily damage the integrity and functionality of the code, it is usually through adding noise such as garbage instructions to achieve fooling the malware detection model. In this paper, we propose an adversarial machine learning method on malware detection model based on OpCode n-grams feature. We first collect a large number of malicious code and normal code data sets, and use TF-IDF to extract OpCode n-grams with different n values from the data set. Then we train three malware detection models based on OpCode n-grams. Under the premise of comprehensive consideration of efficiency, accuracy and interpretability, we select XGBoost as the adversarial feature extraction model and extract adversarial features. Finally, in order to verify the accuracy of the extracted features, we conduct an adversarial machine learning experiment. The experimental results show that the adversarial method proposed in this paper can completely fool the machine learning detection model.**

1. Introduction

With the vigorous development of the information industry, security incidents caused by malware are also in an inexhaustible variety. The "2018-2019 Annual Security Report"[1] issued by the world-renowned anti-virus software testing agency AV-TEST pointed out that nearly 400,000 new malware appear everyday, and computer protection software has to resist more than 3.9 malwares per second. Moreover, as the protection performance of anti-virus software continues to improve, the technology for hackers to write malicious code has become more advanced and faster. Therefore, coping with malware is getting more and more challenging, given their relentless growth in complexity and volume.

Recent years, with the continuous maturity of machine learning technology, machine learning has gradually been applied in the area of malicious code detection and got fairly good effect. Schultz et al. [1] innovatively proposed the use of machine learning algorithms to detect malicious code. Studies have shown that the accuracy of malicious code detection based on machine learning algorithms has reached 97.76%. Compared with the anti-virus engines based on signatures at that time, the accuracy rate has improved more than one times. Abou-Assaleh et al. [2] used the N-Gram features of binary character in malicious sample, and achieved 98% accuracy on a small sample set of more than 100 samples in the case of tri-fold cross-validation. Michael et al. [3] used the situation of system state changes to detect malicious code behaviors, and reached a detection rate of 91% on a data set of more than 4,000 samples, exceeding the detection accuracy of mainstream anti-virus engines. Especially with the development of cloud computing and big data technology, it provides guarantees for computing power and data volume for malicious code detection technology based on machine learning. The malicious code detection technology based on machine learning is to constitute vectors through the disassembly and feature extraction of malicious code, and then use various algorithms to build a classification model based on machine learning, and finally use a large number of samples to perform iterative learning for the purpose of predicting the malicious code. Currently, many anti-virus software have integrated artificial intelligence engines, such as 360's QVM engine.

Although the use of machine learning techniques has greatly promoted the development of malware detection technology, due to the weaknesses and problems of machine learning models, much researches about adversarial machine learning have been generated. At present, researchers mainly

§ *Gang Zhao is corresponding author.*

1. https://www.av-test.org

978-1-7281-9558-2/20/$31.00 ©2020 IEEE
DOI 10.1109/DSC50466.2020.00066

Authorized licensed use limited to: National Taiwan Univ of Science and Technology. Downloaded on February 11,2025 at 14:38:29 UTC from IEEE Xplore. Restrictions apply.

carry out research on adversarial machine learning in the fields of images recognition and speech recognition [4] [5] [6]. Through carefully constructing and modifying the dataset of images or speeches, the machine learning model cannot correctly recognize the image or voice. The malicious code is a carefully constructed executable program, so the modification of malicious code is different from the modification of images. Modifying of the key features of the malicious code can easily damage the integrity and functionality of the program, resulting in the unexecutable or the function changing of the malicious code. Therefore, the adversarial machine leaning in the area of malware detection is currently the main problem.

In this paper, we propose a method of adversarial machine learning in malicious code detection to find out the adversarial features of Portable Executables in Window environment. The OpCode sequence which are generated during the disassembly of a large number of malicious code samples is extracted, and TF-IDF is used to extract OpCode n-grams. Then, based on the OpCode n-grams, three representative models include SVM, CNN, and XGBoost are trained. Under the premise of comprehensive evaluation of efficiency, accuracy and interpretability, we select XGboost as the malicious code feature extraction model. Based on the feature importance of weight, cover, gain, and the SHAP value of each feature, we collect the important features which can affect the test results of the model. In order to verify the accuracy of the selected features, we design two experiments in real-world dataset. We use the traditional obfuscation method to achieve fine-grained modification of the selected features in the malicious code, on the premise of ensuring code integrity and functionality. Then we use the previously trained SVM, CNN, XGBoost models to detect the modified malicious code, the test result shows that the adversarial method proposed in this paper can completely fool the machine learning detection model.

2. Related Work

Plenty of previous works have focused on analyzing malware based on machine learning. Majorities of them can be considered as classification-type solutions. Anderson et al. propose a detection method which utilizes the instruction traces. They modify a malware analysis framework called Ether to collect the instruction traces, and then create the similarity matrix according to the graph kernels combinations between instructions trace graphs. The matrix is finally fed into SVM to perform classification by using 2-gram based Markov chain to estimate the transition probability. Two different similarity evaluation method are applied to construct the matrix, namely Gaussian kernel and spectral kernel, which can measure the local similarity and global similarity between graphs [7]. Santos et al. combine dynamic and static features to propose a hybrid malware detector, which uses static analysis to model binary files into OpCode sequences for features extraction, and dynamic analysis is used to monitor the operations, system calls and exceptions meanwhile [8]. Saxe et al. take byte entropy histogram, PE import information and PE metadata as features, and use deep neural network and Bayesian calibration model to detect malwares [9]. Hardy et.al also apply the deep learning frame work, and combine SAE model to do the malware detection based on Windows API call features [10]. Raff et al. optimize the selected features for detecting malwares. Firstly, they only select the n-gram sequences that have high frequency (more than 1%), and a coarse-grain selection method is applied to reduce the data amount. The final features are determined after the logistic regression test in lasso and resilience models [11]. Given the argument that based on N-Gram have giant computation overhead while the effect is limited, Raff et al. propose that the source of features extraction can be limited to the binary file headers, and then the n-gram features can be directly obtained from the raw byte stream. They use fully-associative neural network and regression network as the classifier in this work [12]. To avoid the problem that feature extraction may impede learning process, Raff et al. present a work that feed the whole binary files into the convolution neural network, and the neural network do the feature extraction and classification directly. The CNN can convert the embedded byte stream into features so that more feature information can be obtained [13]. Xu et al. point that while the graph-matching based algorithm is widely used in similarity evaluation of multiple-platform binary file, it is quite time-consuming and not highly accurate. They propose a neural network based evaluation method called Gemini. Gemini can convert the graph into feature vector in its graph embedded network layer, and evaluate the difference between feature vectors [14]. Xu et al notice that one fixed feature of malware is that they are destined to change the control flow and data structure, therefore they propose a machine learning method based on virtual memory access pattern. A large amount of memory access information is processed to form histogram so that significant features can be preserved and distinguished. Several classification methods are application in this work, including logistic regression, SVM and random forest [15].

Current machine learning targeted malware technology focus on two aspects: attack during training phase and attack during prediction phase. The former mainly refers to poison attack, which tries to modify the statistical features of dataset, so that the machine learning model can be compromised. In most cases, the primary data is encrypted to prevent being modified easily, however in real-world scenario the dataset may vary as the environment changes, and correspondingly the machine learning model should be retrained, which leaves opportunity for attackers to operate the training data. Attack during prediction phase generally means take use of some weaknesses in machine learning model. Biggio et al. propose an optimized further-prioritized label flipping (FPFL) attack. This method modifies the train data and random hyperplane that is far from the decision boundary of SVM, which can incur lower accuracy compared with original FPFL attack [16]. Hu et al. states that although the current machine learning based antivirus software has a

Authorized licensed use limited to: National Taiwan Univ of Science and Technology. Downloaded on February 11,2025 at 14:38:29 UTC from IEEE Xplore. Restrictions apply.

black-box structure, the features that it checks can still be tested and confirmed. This work proposes MalGAN, which can generate adversarial sample to pass though the black-box check model [17]. Kreuk et al. present a GAN model for malware detector which use raw binary files as input. This model can generate one-key representation of adversarial discrete byte steam to reconstitute binary file. The reconstitute binary file can avoid being detected while keeping the original capability [18]. GoodfFellow et al. FGSM to efficiently produce adversarial samples, which attach noise to raw image in the direction of gradient descent [19]. Sarkar et al. propose two black-box attack: UPSET and ANGRI. For a machine learning model that classify samples into N sets, UPSET tries to produces K image-independent, universal disturbance. When attached with the disturbance, the image is in fact not belonged to the original category while the machine learning model still classify it to the same category. In the contrary, ANGRI produces image-dependent, specific disturbance for each unique image [6]. Carlini et al. propose C & W attack, which generate adversarial samples using L0-norm, Euclidean distance and Chebyshev distance. C & W attack has faster generation speed and great portability, which means the generated samples can also be used in black-box attack [20].

3. Approaches

3.1. Feature representation

In this paper, we extract OpCode features from PE samples and express the samples with OpCode expressions. The OpCode is the part of instructions that are specified in the machine instruction language to perform the operation. A complete machine language instruction generally contains an operation code and several operands. Depending on the CPU architecture, the operands for OpCode operations may include registers, values in memory, values stored in the stack, I/O ports, buses, and so on. The operation of operation code can include arithmetic, data operation, logic operation and program control. In the past decade, some malware detection research based on binary information of files has been started gradually. The paper [21] shows that through statistical analysis of OpCodes, it is found that there are obvious differences between malicious code and normal software in the distribution of OpCodes. Malicious code often uses some rare OpCodes, and a method of malicious code detection based on OpCodes is proposed. Based on the distribution of OpCodes, Shabtai et al. [22] employed detection method based on n-gram feature. N-gram is a string with all substrings of length n. A string is simply divided into fixed length n substrings. In 2015, Microsoft launched a malicious code classification competition on kaggle[2], and the champion team from the University of Pittsburgh also used the OpCode n-gram feature.

The problem of locating malicious code signature can be transformed into the problem of finding malicious OpCode n-gram sequence in samples, because from the perspective of compilation principle, binary machine code and assembly instruction's OpCode can be transformed into each other. From the perspective of malicious code classification, there are different families of malicious codes, and the malicious codes of the same family often have similar functions. Although the malicious code authors use various polymorphic deformation techniques to avoid killing, they do not modify the function of the program, mainly because the external view of the program changes, and they still have a high degree of similarity in the local sequence of operation codes, which are similar part of it can be seen as the "fingerprint" or "gene" of this malicious code family.

2. https://www.kaggle.com/

3.2. Feature Extraction

To extract OpCodes from PE samples, we need to disassemble the samples. Disassembly translates the machine instructions stored in the PE file into a language that is more easily readable by human beings, that is, assembly instructions. Finally, the sequence of OpCodes generated in the disassembly process is extracted, and its logical order is the same as that of the operation codes appearing in the executable file, without considering other information (such as memory location, register, etc.). In this paper, we use IDA to realize disassembly. We transform IDA Python script to disassemble PE Sample automatically. Generate ASM file to store assembly instructions and traverse ASM file to obtain OpCode sequence. Finally, the corresponding OpCode n-gram is generated according to different N values. The process is shown in Figure 1.

In this paper, we collected ASM files generated by disassembly from the open sources and divided the training set and test set according to the ratio of 7:3. And, the 2-gram, 3-gram and 4-gram sequences of operation codes are extracted from each ASM file. In the field of machine learning, a large number of features will not only increase the training time of the model, but also sometimes can not improve the accuracy of the model, or even reduce the accuracy. Therefore, we need to select features and reduce the number of features used in training while maintaining the accuracy of the model. We filter features according to document frequency (DF) of OpCode n-gram, and calculate term frequency inverse document frequency (TF-IDF) weight for each n-gram. Formula (1) gives the calculation method of word frequency. TF-IDF combines the word frequency (TF) of an entry in a document with the frequency (DF) of the entry in the document set, and its calculation method is shown in formula (2). Where n is the total number of documents in the whole document set, and DF is the number of documents containing the entry.

$$TF = \frac{TermFrequency}{max(TermFrequency \in document)} \quad (1)$$

$$TFIDF = TF * log(\frac{N}{DF}) \quad (2)$$

Authorized licensed use limited to: National Taiwan Univ of Science and Technology. Downloaded on February 11,2025 at 14:38:29 UTC from IEEE Xplore. Restrictions apply.

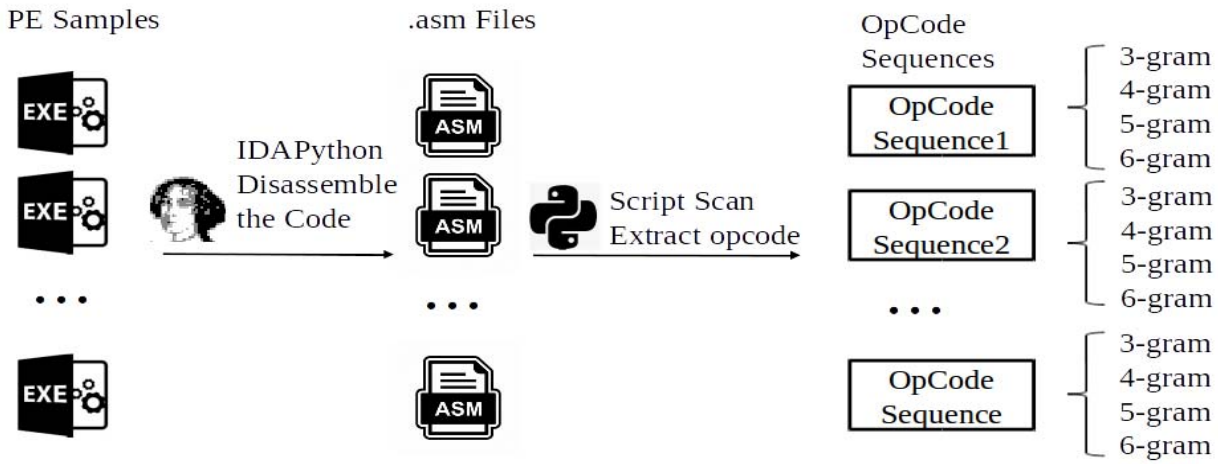

Figure 1. The Process of the OpCode generation

In order to reduce the number of OpCode n-grams, we first select the first 1000 OpCode n-grams according to the document frequency, and calculate their TF-IDF weights as features for model training.

3.3. Machine Learning Training and Feature Evaluation

3.3.1. Machine Learning Model Selection. Moskovitch et al. [23] deployed four classification algorithms based on OpCode n-gram: artificial neural network (ANN), decision tree (DT), naive Bayes (NB) and promotion decision tree (BDT). Through comparative experiments, it is proved that the BDT has the best performance in this task. In this paper, we evaluate Deep Neural Network(DNN), Support vector machine(SVM) and XGBoost in our dataset. The evaluation results are shown at Experiment section. The XGBoost model has better performance on classifying the malware. Note that, the XGBoost model's features have better interpretability. Therefore, we use XGBoost model to distinguish between malicious samples and benign samples.

3.3.2. Feature Evaluation. After obtaining XGBoost model, we need to explain the prediction results of the model and locate the important features that affect the decision-making of the model. Machine learning model can find the difference between malicious code and normal software. We analyze the features that lead to input samples being classified as malicious tags by the model, and use these OpCode n-gram black features to realize intelligent derivation of malicious code signatures. Feature importance is a traditional method to explain machine learning model decision. In this paper, we list three methods to measure the importance of different types of features:

1.Weight: The total number of times feature f splits in all XGBoost subtrees.

2.Cover: The average coverage of feature f to input samples when it splits in all XGBoost subtrees.

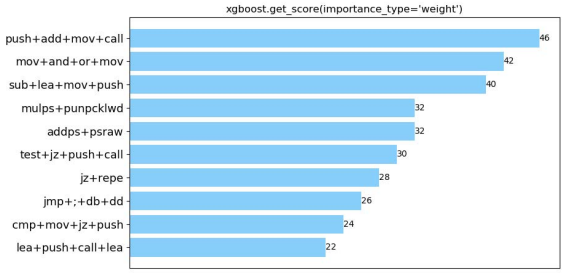

Figure 2. The importance of Weight

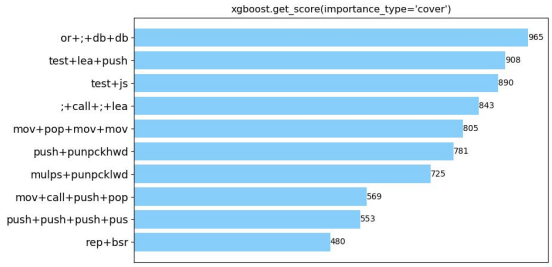

Figure 3. The importance of Cover

3.Gain: The average value of feature f improving the accuracy of the model at each split.

We sort the three types of feature importance, and output the ten most important features and their importance, as shown in figure 2-4.

As the figures show that by selecting different types of feature importance, we can get different results. When selecting weight, 4-gram feature "push + add + mov + call" is the most important feature, but when selecting cover and gain, 4-gram feature "or +; + db + db" and 2-gram feature "mulps + punpcklwd" seem to have the greatest impact on model decision-making. The results of these three types

Authorized licensed use limited to: National Taiwan Univ of Science and Technology. Downloaded on February 11,2025 at 14:38:29 UTC from IEEE Xplore. Restrictions apply.

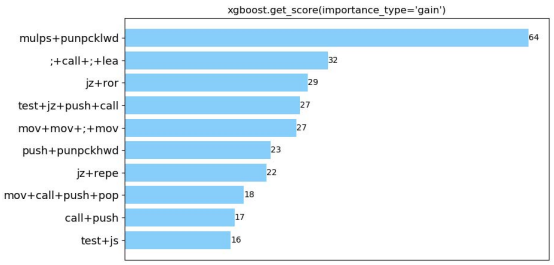

Figure 4. The importance of Gain

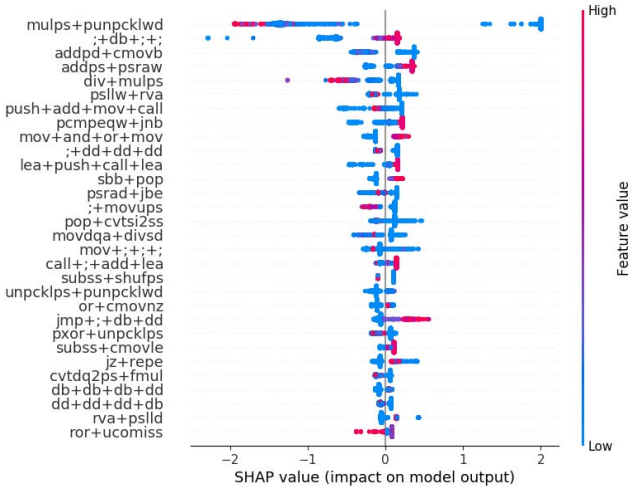

Figure 5. Distribution of some characteristic shap values

of feature importance calculation are inconsistent, which is not conducive to our accurate evaluation of the important features of the model. Therefore, we can not analyze the relationship between the features and the prediction results of XGBoost model according to the importance of features, nor can we interpret the positive and negative effects of different features on the prediction results of samples. In order to solve the inconsistency of feature importance in XGBoost, random forest and other tree set models and to explain the influence of feature on prediction results. We use the SHAP framework based on game theory. The SHAP framework can calculate the snap values for all features of the test samples, which can reflect the specific impact of each feature on the prediction results. As for the malicious code detection model, a test sample s has the feature f. if it is calculated that the SHAP value of the feature f in the sample s is positive, it means that the feature f classifies the sample s as malicious tag 1; if the SHAP value is negative, it means that the feature f classifies the sample s as benign tag 0. The results are shown in figure 5.

Figure 5 is a scatter diagram. Each row in the figure represents a feature. The x-axis abscissa is the corresponding snap value of the feature, and the y-axis ordinate is the feature name. Each point represents a sample in the training set, and the color of the point represents the value of the feature corresponding to the y-axis. The redder the color, the larger the value of the feature, and the bluer the value of the feature. Because the XGBoost model in this paper uses 1000 n-gram features, the Y-axis of figure 5 can't display all feature names. Figure 5 controls the number of features to 30, and visualizes the distribution of their shap values again.

According to the distribution of the shap values of the features, we can locate the important features intuitively and get the correlation between the features and the prediction results. As shown in Figure 5, the 4-gram "mov + and + or + mov" feature will significantly affect the prediction results. The red dots are basically concentrated in the area where the swap value is greater than zero, and the blue dots are basically concentrated in the area where the swap value is less than zero. It can be seen that the increase of the weight of this feature will increase the probability that the samples are predicted as malicious tags by the model. The "mov + and + or + mov" feature with large weight is a typical black feature.

4. Experiments

In this section, we organize two experiments in real world dataset. The experiment A is to compare three machine learning model on malware classification. The experiment B is to evaluate the effectiveness and availability of the adversarial features found by the methods.

4.1. Experiment A

4.1.1. Dataset. Dataset is a prerequisite for training models. In order to improve the authenticity and typicality of the malicious code, we select VirusShare[3] as the data source. VirusShare.com is a malicious code sample library. By continuously releasing the latest captured malicious code, it provides malicious code samples for security research, incident response, and judicial forensics. Currently, more than 34 million malicious code samples have been collected. Since 80% of the malicious code has been packed, which affects the accuracy of feature extraction, we selected the malicious code samples in VirusShare that have been unpacked by the CodexGigas team. So we can not only ensure the authenticity of the samples, but also eliminate the unpacking preprocessing step, and accelerate the extraction of features from the samples.

In addition, we collected windows software that has undergone 360 security tests, and PE files on Window7 and Window10 as benign dataset. The dataset uses for training and testing is shown in Table 1.

4.1.2. Results. In this experiment, we evaluate three classic machine learning models with the dataset mentioned before.

3. https://virusshare.com/

Authorized licensed use limited to: National Taiwan Univ of Science and Technology. Downloaded on February 11,2025 at 14:38:29 UTC from IEEE Xplore. Restrictions apply.

TABLE 1. DATASET OF MACHINE LEARNING COMPARISON

Feature	Train dataset	Test dataset	Total
Malware	6018	2950	8968
Goodware	1909	1120	3029
Total	7927	4070	11997

TABLE 2. PERFORMANCE OF THREE MALWARE DETECTION MODELS BASED ON SVM, DNN, AND XGBOOST ALGORITHMS

Performance	SVM	DNN	XGBoost
Precision@Malware	0.94	0.95	**0.99**
Recall@Malware	0.92	0.95	**0.99**
Precision@Goodware	0.95	**0.96**	**0.96**
Recall@Goodware	0.96	0.97	**0.99**

We use precision and recall to measure the performance of each model. The results are shown at table 2. As we can see, all the machine learning models perform well on classifying the malwares and the goodwares. The precision and recall rates with malware and goodware are higher than 92%. The DNN model perform better than SVM model. Specifically, the XGBoost model has the best performances on malware classification. That is the reason that we employ XGBoost model in this paper.

4.2. Experiment B

4.2.1. Sample construction. In the adversarial experiment, we randomly select 10 samples that were detected as malware by the machine learning model from the test samples. We use the XGBoost model trained in the previous to obtain the key features, and use instruction substitution to blur the key features.

Instruction substitution is using equivalent instruction sequences to replace original instruction sequences in a program. For example, the instruction "mov eax, ebx" can be replaced by "push ebx; pop eax". Correspondingly, the OpCode sequence "mov + and + or + mov" mentioned above can be replaced with "push + pop + and + or + push + pop". There are a variety of instructions in X86 or ARM instruction sets so that it provides sufficient conditions for the implementation of instruction substitution.

In addition, when replacing the instruction, it is also necessary to consider two situations caused by the different length of the replacement instructions and the original instructions: instruction contraction and instruction expansion. Instruction contraction means that the length of the new instructions after replacement is less than the original instructions. In view of this situation, we use the method of inserting "nop" instruction to fill the vacant part. Instruction expansion means that the length of the new instructions after replacement is greater than the original instructions. Because this situation is more complicated to modify, a little negligence will destroy the integrity of the program. Therefore, only short or equal length instructions are used in instruction substitution in this paper, thus avoiding the problem of instruction expansion.

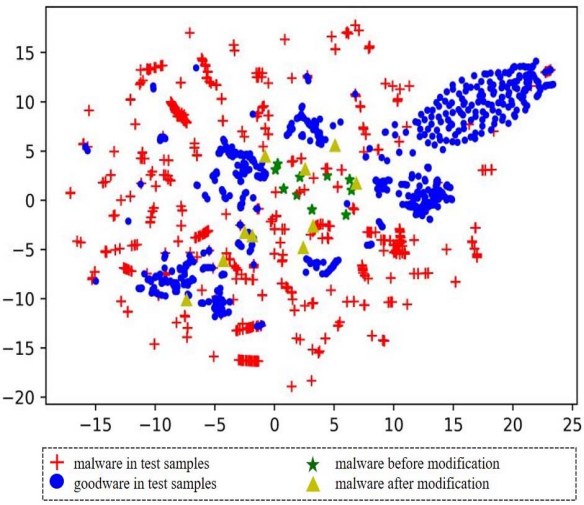

Figure 6. Comparison of SVM model results before and after adveriarial perturbation

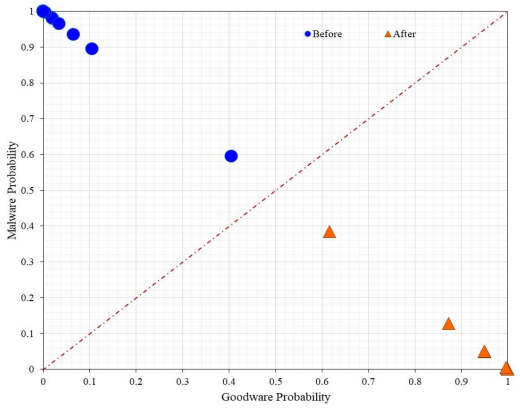

Figure 7. Comparison of DNN model results before and after adveriarial perturbation

4.2.2. Results and Discussion. After the key features are blurred, we use the SVM, DNN and XGBoost models trained in the previous article to predict the 10 new samples obtained. All these 10 samples are predicted as benign samples by the three models. The specific experimental results are shown in the figure below.

Figure 6 uses the tSNE dimensionality reduction algorithm for the SVM model, where • represents goodware in the test sample and represents malware in the test sample. The ⋆ represent 10 malicious samples randomly selected from the test samples. The average distance of these 10 malicious samples from the hyperplane in the SVM model is 1.24. The △ represents 10 new malicious samples obtained by modifying the characteristics of these 10 malicious samples. The 10 new malicious samples are all predicted to be benign by the SVM model, and the average distance from the hyperplane in the SVM model is -1.72.

Figure 7 shows the probability distribution of the ma-

Authorized licensed use limited to: National Taiwan Univ of Science and Technology. Downloaded on February 11,2025 at 14:38:29 UTC from IEEE Xplore. Restrictions apply.

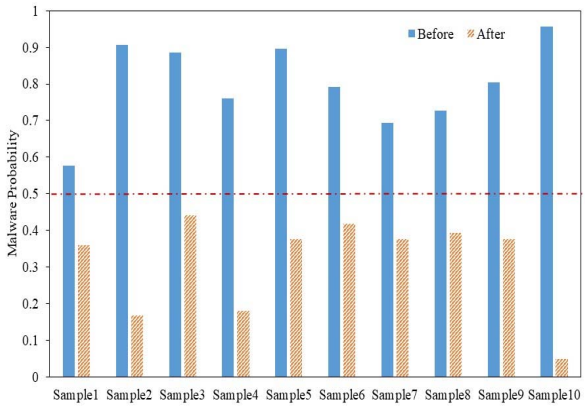

Figure 8. Comparison of XGBoost model results before and after adveriarial perturbation

licious code sample predicted by the DNN model before and after modification, where the abscissa represents the probability that the sample is predicted to be benign by the model, the ordinate represents the malignant probability. • represents the original malicious sample before modification, and △ represents the malicious sample after modification. It can be seen from the experimental results that the original samples are all distributed in the left half of the diagonal, indicating that these 10 original samples are all detected as malicious samples by the DNN model, and the modified samples are distributed in the right half of the diagonal, indicating that the modified sample can bypass the detection of the DNN model. Among them, the maximum increase rate of the benign probability is 99.78%, and the benign probability of the 10 samples is increased by 87.19% on average.

Figure 8 shows the probability of the malicious code sample being predicted by the XGBoost model before and after modification, where the ordinate represents the malicious probability, the blue histogram represents the malicious probability before modification, and the orange histogram represents the modified malicious probability. It can be seen that the malicious probability of the samples before modification is above the midline, indicating that these 10 original samples are all predicted as malicious samples by the XGBoost model. The malicious probability of the modified samples are all less than 50%, indicating that the samples after modification can completely avoid the detection of the XGBoost model. Among them, the maximum decline rate of malicious probability is 90.79%, and the average decline rate is 48.61%.

It can be seen from the above experimental results that the adversarial feature extraction method based on OpCode can not only effectively fool the XGBoost model for feature extraction, but also fool other malicious code detection models such as SVM and DNN. In addition, the above experimental results also reflect from the side that the features used by malicious code detection model based on SVM and DNN may be similar to XGBoost in predicting malicious code.

5. Conclusion and Future Work

This paper proposes and implements an adversarial machine learning method based on OpCode n-gram for malware detection models. We first collect 11997 samples, including 7927 malicious code samples and 4070 normal samples, and train three machine learning detection models including SVM, DNN, and XGBoost based on OpCode n-gram features. Then after comprehensively evaluating the three models, we adopt XGBoost model as the adversarial feature extraction model, and find the adversarial features based on OpCode n-gram. We randomly select 10 malicious samples. After modifying the adversarial features, we again use three malware detection models to make predictions. Experimental results show that the modified malicious code samples can not only fool the XGBoost-based malware detection model, but also fool the SVM and DNN malware detection models. On the one hand, it verifies the accuracy of the adversarial features. On the other hand, it also shows that the adversarial features we find have certain transitivity between different models.

At present, our adversarial features are mainly focused on malicious features. In the future, we will try to find benign features that can perturb machine learning models, and conduct experiments on more models and larger data sets.

References

[1] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *IEEE Symposium on Security Privacy*, 2001.

[2] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "N-gram-based detection of new malicious code," in *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.*, vol. 2, 2004, pp. 41–42 vol.2.

[3] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2007, pp. 178–197.

[4] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. Mcdaniel, "Ensemble adversarial training: Attacks and defenses," 2017.

[5] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," pp. 1765–1773, 2017.

[6] S. Sarkar, A. Bansal, U. Mahbub, and R. Chellappa, "Upset and angri : Breaking high performance image classifiers," 07 2017.

[7] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," *Journal in computer Virology*, vol. 7, no. 4, pp. 247–258, 2011.

[8] I. Santos, J. Devesa, F. Brezo, J. Nieves, and P. G. Bringas, "Opem: A static-dynamic approach for machine-learning-based malware detection," in *International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions*. Springer, 2013, pp. 271–280.

[9] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2015, pp. 11–20.

Authorized licensed use limited to: National Taiwan Univ of Science and Technology. Downloaded on February 11,2025 at 14:38:29 UTC from IEEE Xplore. Restrictions apply.

[10] W. Hardy, L. Chen, S. Hou, Y. Ye, and X. Li, "Dl4md: A deep learning framework for intelligent malware detection," in *Proceedings of the International Conference on Data Mining (DMIN)*. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2016, p. 61.

[11] E. Raff, R. Zak, R. Cox, J. Sylvester, P. Yacci, R. Ward, A. Tracy, M. McLean, and C. Nicholas, "An investigation of byte n-gram features for malware classification," *Journal of Computer Virology and Hacking Techniques*, vol. 14, no. 1, pp. 1–20, 2018.

[12] E. Raff, J. Sylvester, and C. Nicholas, "Learning the pe header, malware detection with minimal domain knowledge," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 121–132.

[13] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, "Malware detection by eating a whole exe," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] X. Xu, C. Liu, Q. Feng, H. Yin, L. Song, and D. Song, "Neural network-based graph embedding for cross-platform binary code similarity detection," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 363–376.

[15] Z. Xu, S. Ray, P. Subramanyan, and S. Malik, "Malware detection using machine learning based analysis of virtual memory access patterns," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 169–174.

[16] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Asian conference on machine learning*, 2011, pp. 97–112.

[17] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on gan," *arXiv preprint arXiv:1702.05983*, 2017.

[18] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, and J. Keshet, "Adversarial examples on discrete sequences for beating whole-binary malware detection," *arXiv preprint arXiv:1802.04528*, 2018.

[19] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[20] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[21] Bilar and Daniel, "Opcodes as predictor for malware," *International Journal of Electronic Security Digital Forensics*, vol. 1, no. 2, p. 156, 2007.

[22] R. K. Shahzad and N. Lavesson, "Detecting scareware by mining variable length instruction sequences," in *Scandinavian Conference on Artificial Intelligence*, 2011.

[23] R. Moskovitch, C. Feher, N. Tzachar, E. Berger, M. Gitelman, S. Dolev, and Y. Elovici, "Unknown malcode detection using opcode representation," 2008.

Authorized licensed use limited to: National Taiwan Univ of Science and Technology. Downloaded on February 11,2025 at 14:38:29 UTC from IEEE Xplore. Restrictions apply.