Combination of General-Purpose and Attack-Specific Detectors against Adversarial Malware

Yoshiki Namura*, Tomotaka Kimura*, and Jun Cheng*
*Graduate School of Science and Engineering, Doshisha University, Kyoto, Japan
Email: tomkimur@mail.doshisha.ac.jp; jcheng@ieee.org

***Abstract*—In this paper, we propose a two-stage detection method for detecting malware and adversarial malware. In the first stage, we use a general-purpose detector that targets several types of adversarial malware to determine whether the input data are malware. In the second stage, we use attack-specific detectors to detect malware that was determined to be benign in the first stage. The second-stage detector is composed of multiple detectors. We train each detector using data generated by a specific adversarial malware generation method. Through experiments on the Android malware dataset, we demonstrated that our proposed method improved accuracy compared with existing methods.**

***Index Terms*—Deep learning, adversarial malware, evasion attack, malware detection**

I. Introduction

In recent years, the number of malware incidents has increased rapidly. There have been cases of confidential information being stolen using malware. To combat these malware incidents, machine learning-based malware detection has been actively studied [1]. In machine learning-based malware detection, the characteristics of the software are converted into feature vectors that are evaluated to determine whether the software is benign or malicious. Although machine learning-based malware detection systems are a promising solution, they are exposed to the threat of evasion attacks, which is an inherent threat to machine learning [2]. Evasion attacks cause misclassification and false positives by applying small perturbations to the inputs. When an evasion attack is carried out against the malware detection system, malware is overlooked. Overlooking malware can lead to serious incidents. Thus, countermeasures against evasion attacks are an important issue.

As a countermeasure against evasion attacks, the ADE (Adversarial Deep Ensemble) has been considered [3]. ADE performs adversarial learning for a single detector using data generated by multiple evasion attack methods. Using data from multiple evasion attacks, ADE achieves robustness against learned and similar evasion attacks. Although ADE is robust against various evasion attacks, because the characteristics of each attack are different, training them all together in a single detector results in lower generalization ability in detection accuracy for each attack. Therefore, it is difficult to distinguish between various attacks using the single general-purpose detector and it is necessary to take different countermeasures for each type of evasion attack.

In this paper, to defend multiple evasion attacks, we propose a two-stage detection method. In the first stage, we used a general-purpose detector that classifies software as benign or malicious without much consideration of the robustness against evasion attacks. Because the benign software identified by this detector may include adversarial malware generated by evasion attacks, we re-evaluate the software identified as benign using a second-stage detector. Second-stage detectors consist of multiple evasion attack-specific detectors trained on data generated by different evasion attacks that were judged to be benign by the first-stage detectors. Thus, the second stage is dedicated to discriminating adversarial malware included in the data determined to be benign in the first stage. For adversarial malware generated by each evasion attack, the accuracy of discrimination can be improved by learning the data generated by the evasion attack individually in each of the second-stage detectors. Through experiments using the Android malware dataset, we show that accuracy can be improved using two-stage detection in the proposed method.

This paper is organized as follows. Section 2 presents the proposed two-stage detection method and the detection flow. Section 3 evaluates the effectiveness against some kind of attacks. Section 4 concludes the paper and shows future research.

II. Proposed Method

Figure 1 shows a flow chart of the proposed two-stage detection method. In the first stage, to detect malware, the data $\boldsymbol{x}$ of the software for which the decision is to be made is input into the general-purpose detector f. The first-stage detector f is trained using a dataset $\mathcal{D}$ that is labeled as benign or malware. Because the first-stage detector does not take into account adversarial malware, it is difficult to detect all adversarial malware. Therefore, the proposed method aims to detect adversarial malware using the second-stage detector.

In the second stage, multiple attack-specific detectors are used in sequential order to detect adversarial malware. Let N detectors be denoted by $g_1, g_2, \ldots, g_N$. These detectors are trained using data that has been added to the training dataset, $\mathcal{D}$, with the addition of adversarial malware generated by specific evasion attacks, and using data that has been judged

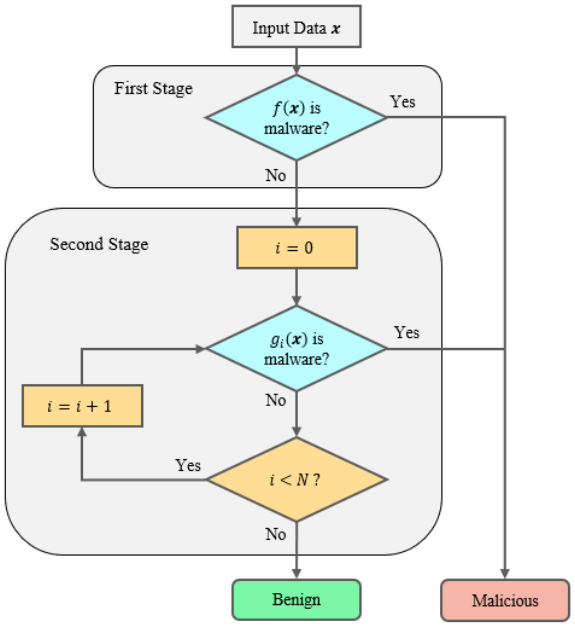

Fig. 1: Flowchart of the proposed method

to be benign by the first stage of detection. Many methods for generating adversarial malware have been devised to date, and the feature vectors of each method are diverse. Therefore, detection accuracy could be improved using detectors that have been individually trained rather than using a single detector.

The detailed decision procedure for second-stage detection is as follows: The feature of the software $\boldsymbol{x}$ is the input into the detectors $g_1, g_2, \ldots, g_N$. If any one of these detectors discriminates that software $\boldsymbol{x}$ is malware, then software $\boldsymbol{x}$ is determined to be malware. In contrast, if all the detectors identify that software $\boldsymbol{x}$ is benign, then software $\boldsymbol{x}$ is determined to be benign. Hence, the second stage uses detectors that have learned the specific features of each attack, thus reducing the number of missed scored in the first stage.

III. Performance Evaluation

To demonstrate the effectiveness of the proposed method against evasion attacks, we evaluated its detection accuracy using the AndroZoo dataset [4]. The AndroZoo dataset is a repository of Android applications provided by the University of Luxembourg. From the AndroZoo dataset, we used 51,709 benign and 14,275 malware Android Package Kit (APK) files. From the features contained within these APK files, we used the top 10,000 most frequent features in the training data as 10,000-dimensional binary feature vectors. From the data used for performance evaluation, we used 60% as training data, 20% as validation data, and the remaining 20% as test data.

The first-stage detectors f of the proposed method were the detectors Basic NN, where NN denotes neural network, ADE-MA, and AT-rFGSM.

- **Basic NN**: Basic NN consists of a four-layer structure with 160 neurons in each of the two middle layers. We set the batch size, epoch size, and learning rate to 128, 150, and 0.001, respectively. As the activation function and optimization algorithm, the rectified linear unit (ReLU) function and Adam algorithm are used, respectively.

- **ADE-MA**: ADE-MA is an ensemble detector that learns multiple methods of generating adversarial malware in a mixed manner. The ensemble is composed of five NNs. The average value of the logit of each NN is used as the output of the final ensemble detector. The NN has a four-layer structure. In this study, the number of neurons in the two middle layers was 160 for each layer, the batch size was 128, the number of epochs was 150, the learning rate was 0.001, the activation function was ReLU, and the learning algorithm was Adam. We used four types of adversarial malware generation methods to train ADE-MA: PGD-ℓ_1, PGD-ℓ_2, PGD-ℓ_∞, and FGSM (Fast Gradient Sign Method) attacks using Adam. During the training process, we selected an adversarial malware generation method that maximizes the loss function value of each NN for each mini-batch from the aforementioned four types of generation methods and performed adversarial learning that incorporated the selected adversarial malware generation method.

- **AT-rFGSM**: AT-rFGSM adversarially trains by using the rFGSM method [5] for generating adversarial malware. The same number of adversarial malware samples as training data for malware are generated using rFGSM, and they are trained together with benign software and malware. The detector is constructed using an NN that has the same settings as Basic NN.

In the second-stage detector, we generated adversarial malware and added it to the training data of the first-stage detector, and learned data that the first-stage detector judged to be benign. Like the first-stage detector f, the second-stage detector used a four-layer NN. We used four types of methods for generating adversarial malware: FGSM, JSMA (Jacobian-based Saliency Map Approach), Mimicry, and Pointwise. Therefore, we set the number of detectors to $N = 4$.

- **FGSM**: FGSM is a white-box attack that generates perturbations to increase the loss function value of the malware detector that is the target of the attack [6]. It generates adversarial malware that reduces detection accuracy by differentiating the loss function value for the input malware with the input value and then applying perturbations to the malware so that the gradient increases.

- **JSMA**: JSMA is a white-box attack that searches for efficient perturbations for each feature [7]. The Jacobian matrix in the loss function for the input malware is calculated. Attack performance is obtained with the smallest perturbation by selecting the maximum value in

TABLE I: Accuracy

Detector	Evasion attack				
	No attack	FGSM	JSMA	Mimicry	Pointwise
Proposal (Basic NN)	**99.44**	99.19	98.25	99.37	97.90
Proposal (ADE-MA)	**99.44**	**99.44**	**98.32**	99.65	**98.81**
Proposal (AT-rFGSM)	99.23	99.23	98.21	**99.75**	98.11
Basic NN-Only	99.05	2.84	24.55	0.00	0.00
ADE-MA-Only	99.37	99.37	47.36	38.04	36.60
AT-rFGSM-Only	99.05	99.05	42.57	0.49	0.49

TABLE II: Difference in the number of malware that can be detected in the first and second stages

(a) Proposal (Basic NN)

Attack	First stage				Second stage			
	N_B	$N_B^{(M)}$	N_M	$N_M^{(M)}$	N_B	$N_B^{(M)}$	N_M	$N_M^{(M)}$
FGSM	13,119	2,839	2,933	2,871	10,299	54	2,820	2,785
JSMA	12,499	2,219	3,553	3,491	10,326	81	2,173	2,138
Mimicry	13,200	2,920	2,852	2,790	10,294	49	2,906	2,871
Pointwise	13,200	2,920	2,852	2,790	10,336	91	2,864	2,829

(b) Proposal (ADE-MA)

Attack	First stage				Second stage			
	N_B	$N_B^{(M)}$	N_M	$N_M^{(M)}$	N_B	$N_B^{(M)}$	N_M	$N_M^{(M)}$
FGSM	9,892	51	6,160	5,659	9,858	33	34	18
JSMA	11,377	1,536	4,675	4,174	9,890	65	1,487	1,471
Mimicry	11,643	1,802	4,409	3,908	9,852	27	1,791	1,775
Pointwise	11,684	1,843	4,368	3,867	9,876	51	1,808	1,792

(c) Proposal (AT-rFGSM)

Attack	First stage				Second stage			
	N_B	$N_B^{(M)}$	N_M	$N_M^{(M)}$	N_B	$N_B^{(M)}$	N_M	$N_M^{(M)}$
FGSM	10,366	94	5,686	5,526	10,288	47	78	47
JSMA	11,979	1,707	4,073	3,913	10,317	76	1,662	1,631
Mimicry	13,180	2,908	2,872	2,802	10,273	32	2,907	2,876
Pointwise	13,180	2,908	2,872	2,802	10,320	79	2,860	2,829

the matrix.

- **Mimicry**: Mimicry is a black-box attack that disguises the characteristics of malware as those of benign software [8]. By replacing all characteristics except for the minimum necessary for functioning as malware with the characteristics of benign software, the malware acquires characteristics that are judged to be benign software by the detector.

- **Pointwise**: Pointwise is an attack that minimizes the magnitude of perturbations added to adversarial malware [3]. By randomly repeating the removal of perturbations at feature units, it generates adversarial malware that is difficult to detect.

For each generation method, we generated 14,275 adversarial malware from the APK files of malware. From the adversarial malware, we used 60% as training data, 20% as validation data, and the remaining 20% as test data.

As an evaluation metric, we used the accuracy of two-stage detection in an attack environment. Accuracy is as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{1}$$

where TP denotes true positive and refers to the total number of correctly identified positives, FP denotes false positive and refers to the total number of incorrectly identified positives, TN denotes true negative and refers to the total number of correctly identified negatives, and FN denotes false negative and refers to the total number of incorrectly identified negatives. In the experiments, we considered malware data as positive and benign data as negative.

TABLE III: Accuracy for Benign in the First-Stage and the Second-Stage Detection

Detector	–	Evasion attack			
		FGSM	JSMA	Mimicry	Pointwise
Basic DNN	First-stage	97.89	98.25	97.83	97.83
	Second-stage	99.03	98.87	99.07	98.80
ADE-MA	First-stage	**91.87**	**89.28**	**88.64**	**88.53**
	Second-stage	96.55	98.86	96.59	96.44
AT-rFGSM	First-stage	97.19	96.22	97.56	97.56
	Second-stage	99.05	98.87	99.15	98.85

A. Results

Table I shows the accuracy of two-stage detection, where Proposal (Basic NN), Proposal (ADE-MA), and Proposal (AT-rFGSM) represents the proposed method with Basic NN, ADE-MA and AT-rFGSM as first-stage detector, respectively. We also show the results of Basic NN-only, ADE-MA-only, and AT-rFGSM-only that indicate the existing methods of Basic NN, ADE-MA, and AT-rFGSM, respectively. Table I shows that accuracy improved when we used two-stage detection with any of the first-stage detectors. Furthermore, our proposed method not only counteracted evasion attacks that the first-stage detector could not counteract but also improved the detection accuracy of evasion attacks that had already taken countermeasures. Table I shows that the accuracy of the FGSM evasion attack against the first-stage detector AT-rFGSM improved from 99.05% to 99.23%; the improvement was very slight.

Tables II shows the difference in the number of malware that can be detected in the first and second stages. N_B and N_M denote the numbers of discriminated benign and malware, respectively. $N_B^{(M)}$ and $N_M^{(M)}$ also denote the true values of the malware contained in the determined benign N_B and malware N_M. The test data included benign software, malware, and adversarial malware generated by evasion attacks on each. As we can see from Tables II, when the number of malware contained in the data that was discriminated to be benign by the first-stage detector was small, that is, when it was possible to make a benign judgment with high accuracy in the first-stage detector, the accuracy of malware detection in the second-stage detector decreased. This is because the first-stage detector over-learned the benign data because the proportion of malware in the benign data was low. Therefore, to achieve further improvements in accuracy for adversarial malware that can be detected with high accuracy by the first-stage detector, it is necessary to alleviate the imbalance in the benign data.

The final detection accuracy of the two-stage detection is affected by the accuracy of the malware detection of the first-stage detector and the accuracy of the malware re-detection of the second-stage detector. Therefore, we evaluate the effects of the first-stage detectors. Table III shows the accuracy for benign in the first-stage and the second-stage detection. The test data included benign software, malware, and adversarial malware generated by evasion attacks. As we can see from Table III, the accuracy for the first-stage detector is lowest for ADE-MA and the detection rate for the two-stage detector is also low for ADE-MA. This result implies that the performance of the first stage detection affects that of the second stage detection.

IV. Conclusion

In this paper, we proposed a two-stage detection method that reinforces the accuracy of the first-stage detector as a detection system that counters increasingly diverse adversarial malware. In the proposed method, adversarial examples that the first-stage detector judged to be benign were included in the training data in the second-stage detector. Through experiments using the AndroZoo dataset, we showed that the accuracy was improved by performing two-stage detection rather than single-stage general-purpose detection. We used four types of evasion attack to train the two-stage detector, but dozens of evasion attacks have been considered to date. It is not realistic to learn all of them in advance and construct a detector for each one. Therefore, further studies are needed in the future.

Acknowledgment

This research was supported by JSPS KAKENHI (23K11077).

References

[1] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *Journal of Systems Architecture*, vol. 112, p. 101861, 2021.

[2] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. of Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD'13)*. Springer, 2013, pp. 387–402.

[3] D. Li and Q. Li, "Adversarial deep ensemble: Evasion attacks and defenses for malware detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3886–3900, 2020.

[4] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of android apps for the research community," in *Proc. of the 13th International Conference on Mining Software Repositories (MSR'16)*, 2016, pp. 468–471.

[5] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," in *Proc. of 2018 IEEE Security and Privacy Workshops (SPW'18)*. IEEE, 2018, pp. 76–82.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *In ICLR (Poster)*, 2015.

[7] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. of 2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

[8] N. Šrndić and P. Laskov, "Practical evasion of a learning-based classifier: A case study," in *Proc. of 2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 197–211.