Adversarial Training to Improve Accuracy and Robustness of a Windows PE Malware Detection Model

Luca Lobascio[1,3], Giuseppina Andresini[1,2], Annalisa Appice[1,2,*] and Donato Malerba[1,2]

[1]*University of Studies of Bari "Aldo Moro", Bari, Italy*
[2]*Consorzio Interuniversitario Nazionale per l'Informatica - CINI, Bari, Italy*
[3]*IMT School for Advanced Studies Lucca, Lucca, Italy*

Abstract

Windows PE malware is still considered a major threat in the cybersecurity landscape despite the amazing accuracy recently achieved in malware detection thanks to the use of Artificial Intelligence (AI). In fact, recent advances in Adversarial Learning have definitely shown that adversarial attacks can evade AI-powered decision models trained for Windows PE malware detection. These are malware created by leveraging AI vulnerabilities to fool AI-based malware detection systems. Adversarial Training is one of the fundamental strategies for defending an AI-based decision model against adversarial attacks. So, in this paper, we focus particularly on Adversarial Training as a method for both gaining accuracy and improving the robustness of a deep neural model trained for Windows PE malware detection. To this aim, we analyse the accuracy performance of a deep neural model trained with adversarial samples generated with Fast Gradient Sign Method (FGSM) against real Windows Portable Executable (PE) goodware and malware, as well as realistic Windows PE adversarial malware produced with state-of-the-art techniques.

Keywords

Windows PE Malware Detection, Deep Learning, Adversarial Learning, Adversarial Training

1. Introduction

Artificial Intelligence (AI) is transforming cybersecurity practices thanks to the amazing accuracy performance recently achieved with several AI-based malware detection systems. However, several recent studies have shown that AI-based decision models can be vulnerable to adversarial attacks. In malware detection scenarios, adversarial attacks are realistic manipulations of existing malware, which preserve the executable and malicious behaviour of the original malware but deceive the malware detection measures. In this study, we consider literature methods defined to generate ethical adversarial malware to help identify vulnerabilities of AI-based anti-malware systems and develop defensive strategies. Specifically, we focus on Windows PE malware since, still in 2023, Microsoft Windows remained the primary target for cyber-attacks, accounting for 88% of all malware-filled data detected daily.

Some studies [1, 2, 3, 4] show that Windows Portable Executable (PE) adversarial malware may be produced with catastrophic consequences for security systems of AI-powered malware detection models. In particular, the authors of [2] have recently performed a large-scale study of the evasion performance of five literature attack methods formulated to produce realistic adversarial Windows PE malware in both a white-box (i.e., FGSM padding+slack, Full DOS, Extend, Shift) and a black-box attack scenario (i.e., GAMMA). For each attack method, their study has produced a collection of realistic adversarial Windows PE malware that AI developers may use to test the robustness of AI models and commercial anti-malware. However, the research to identify the best practices to mitigate the adversarial vulnerabilities of AI systems for Windows PE malware detection is still under-explored.

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT
***Corresponding author.
luca.lobascio@imtlucca.it (L. Lobascio); giuseppina.andresini@uniba.it (G. Andresini); annalisa.appice@uniba.it (A. Appice); donato.malerba@uniba.it (D. Malerba)
0009-0000-3011-906X (L. Lobascio); 0000-0002-5272-644X (G. Andresini); 0000-0001-9840-844X (A. Appice); 0000-0001-8432-4608 (D. Malerba)

Adversarial Training [5] is one of the most promising defensive methods to improve the robustness of a decision model by mitigating the malicious effect caused by adversarial attacks. The Adversarial Training strategy has recently been used to mitigate overfitting and gain accuracy in cybersecurity problems. For example, the authors of [6] use samples generated with Fast Gradient Sign Method (FGSM) [7] [8] to mitigate overfitting problems and improve the generalization and diversity of ensemble models trained in problems of Android malware detection and network traffic intrusion detection. FGSM is a white-box, gradient-based, evasion method defined to generate adversarial samples of imagery and tabular data. In [6], FGSM and other general-purpose attack methods have been used on the feature-vector representation of the cyber objects, which was obtained by applying a static analyser of the files or a dynamic analyser of the software behaviour, or a network traffic analyser of the packet flows. However, the adversarial samples generated in this way cannot be mapped into realistic adversarial malware to be used to represent realistic vulnerabilities of AI-based decision models and evaluate models' robustness. Differently, in this study, we explore how an AI-based decision model that is able to generalize on adversarial samples generated with FGSM can improve the robustness against realistic adversarial attacks, gaining accuracy in correctly disentangling goodware from both real Windows PE malware and realistic adversarial Windows PE malware.

The paper is organized as follows. Section 2 describes the background of this work. Section 3 illustrates the Adversarial Training method deployed in this study. Section 4 illustrates the results of the evaluation study. Finally, Section 5 draws conclusions and sketches future research directions.

2. Background

Since its conception adversarial learning has widely investigated the "offensive" scope of AI by defining several evasion methods to identify vulnerabilities in AI-based decision models [9]. Recent studies have also started to devote attention to poisoning methods [10] to attack the learning stage of an AI system. The majority of the state-of-the-art evasion methods are formulated for imagery data and operate in a white-box setting, where the attacker has a complete knowledge of the decision model architecture and its parameters' values. FGSM [7] is one of the most popular white-box, gradient-based, evasion methods. It finds the loss (e.g., the cross-entropy) to apply to an input sample, to make decisions of a neural model less robust for a specific class. Projected Gradient Descent (PGD) [11] is the iterative variant of FGSM. LowProFool [8] uses the gradient data to guide the perturbation towards a target class in an iterative manner by penalizing the perturbation proportionally to the feature importance associated with the features. Deepfool [12] performs an iterative procedure to find the minimum adversarial perturbations on both an affine binary classifier and a general binary differentiable classifier. It integrates the one-versus-all strategy to be applied to multi-class problems. Notably, FGSM, as well PGD and DeepFool methods, are gradient-based methods formulated for the image domains. However, they are also used for tabular data. Although less numerous, a few evasion methods, such as Zeroth Order Optimization (ZOO) [13], are also formulated to operate in a black-box mode, where attackers have no access to information on the target model except for the final decisions.

These general-purpose evasion methods are commonly used in cybersecurity problems, although they can be applied to a feature vector representation of cyber objects. Both [14] and [15] have recently published an extensive survey to describe how general-purpose attack methods have been used in cybersecurity. This survey highlights that the majority of cybersecurity studies still use general-purpose attack methods applied to a tabular representation of cyber objects obtained after a feature engineering step (e.g., static analysis of codes, dynamic analysis of behaviours). In particular, several studies use the Adversarial Training strategy as a means of gaining generality in the decision model development, achieving higher accuracy on real malicious behaviours in the evaluation phase. However, these studies often leave aside the problem of producing realistic evasive objects in cybersecurity domains where, unlike images, there is no clear inverse mapping to the feature space [16]. For example, the authors of [17] use adversarial samples generated with FGSM to mitigate the overfitting phenomenon in a deep neural model trained with Adversarial Training using the guidance of eXplainable AI-based knowledge.

Similarly, the authors of [6] use adversarial samples generated with either FGSM, PGD, DeepFool and LowProFool to mitigate overfitting problems in deep neural model development and improve the generalization and diversity of ensemble systems. In both studies, the adversarial samples are generated in the feature space of malicious objects (i.e., Android malware or network traffic intrusions) and used only in the decision model development stage. As no inverse transform is formulated to transform a perturbed feature vector into a realistic cyber object, these adversarial samples are ignored for the evaluation of the decision model robustness.

On the other hand, the generation of realistic adversarial cyber objects in the problem space has recently attracted significant attention in the cybersecurity literature [1]. Focusing on Windows PE malware, the authors of [18] describe FGSM (padding + slack) that is the seminal white-box evasion method to generate realistic Windows PE adversarial malware to fool MalConv decisions [19]. MalConv is a literature Convolutional Neural Network learned from raw bytes of Windows PE files. As the MalConv model is trained without resorting to a surrogate feature vector representation of Windows PE files, it provides the ideal scenario for white-box attacking models developed in the problem space. Accordingly, the authors of [1] describe three further white-box evasion methods, named Full DOS, Extend and Shift, to evade MalConv. Extend injects noise bytes in the DOS header. Full DOS perturbs the bytes that are placed in the DOS header in the areas before the magic number and after the pointer to the PE header. Both Extend and Full DOS base on the fact that the DOS header is still kept in Windows PE files to make these files still compatible with the older operating systems. So, they change the DOS header, except for the magic number MZ and the four-byte-long integer at offset 0x3c, by keeping the functionality of the executable file Shift applies the shift operation to the first section to recover room to inject an adversarial byte payload. On the other hand, the authors of [20] describe the Adversarial Malware Generator (AMG) that is a black-box evasion method that uses a reinforcement learning agent to combine a set of functionality-preserving binary file manipulations and perturb Windows PE malware. Finally, the authors of [21] describe the Genetic Adversarial Machine learning Malware Attack (GAMMA), that is one of the most effective black-box evasion methods for Windows PE malware detection. It uses an evolutionary algorithm to inject an adversarial payload into a Windows PE malware. It solves an optimization problem that minimizes both the probability of evasion and the size of the benign injected content via a specific penalty term. The injected payload is extracted from goodware binary files, optimizing the selection and the size of benign content using selection, crossover, and mutation functions. The injection is performed with either “padding" or “section injection", which are two manipulations to preserve the maliciousness and executability of PE files, although the higher evasion ability is achieved with the section injection manipulations. Notably, in [1, 2], the authors show the higher evasion ability of GAMMA compared to the competitor white-box evasion methods FGSM (padding + slack), Full DOS, Extend, Shift. This is an intriguing achievement, considering that the black-box scenario is a more practical assumption for attacking a general anti-malware system.

Finally, the authors of [2] have recently explored the performance of a decision model trained through Adversarial Training with the realistic adversarial Windows PE malware produced with either FGSM (padding + slack), Full DOS, Extend, Shift and GAMMA. However, this evaluation study has shown that the decision model obtained with Adversarial Training loses accuracy in disentangling real Windows PE goodware from real Windows PE malware. At the same time, it slightly gains accuracy in recognizing realistic adversarial Windows PE malware when the adversarial objects are generated with either Extend or GAMMA only. Notably, the study of [2] uses the same category of adversarial objects in both the training and evaluation stages of each experiment. Differently, in this study, we use training samples generated with a general-purpose evasion method working in the feature space representation of Windows PE files while we test the robustness of the developed models on realistic adversarial Windows PE malware.

3. Adversarial Training method

In this Section, we describe the Adversarial Training method considered in this study to train a Windows PE malware detection model. This decision model is trained in the feature space extracted using the Library to Instrument Executable Formats (LIEF) [22] This is done according to a few recent studies [23, 2] showing that a decision model trained from LIEF features commonly achieves higher detection rate than a decision model trained (even with complex deep neural networks) from raw binary data. LIEF is used to parse the binary code a Windows PE file and extract 2381 raw static features grouped as follows:

1. ***Byte histogram***: 256 features to measure, for each distinct byte value, the ratio of the counts of each byte value within the file to the total number of bytes recorded in the file;
2. ***Byte entropy histogram***: 256 features to measure the scalar entropy on a sliding window paired with each byte occurrence within the window;
3. ***Strings***: 104 features to describe simple statistics information about printable strings;
4. ***General file***: 10 features to describe the file size and basic information extracted from the PE header;
5. ***Header***: 62 features to represent data extracted from both the COFF and the optional header;
6. ***Section***: 255 features to describe data recorded in the section header;
7. ***Import***: 1280 features to provide information on imported functions and associated libraries extracted from the import address table;
8. ***Export***: 128 features to list information on exported functions;
9. ***Data directory***: 30 features to describe the size and virtual size of the data directory entries recorded in the Windows PE file.

As a decision model, we consider a Deep Neural Network DNN model trained with LIEF features for Windows PE malware detection according to the description reported by [24]. The choice of this DNN model as the target model of the evaluation work is based on the evaluation results illustrated by [24]. In fact, their study shows that such DNN model allows us to achieve higher detection accuracy than several AI-based malware detection methods analysed. In our study, the DNN model is trained with the Adversarial Training strategy using a general-purpose evasion method to generate the adversarial samples for the model development.

As a general-purpose evasion method, we use the FGSM [7] method on the tabular data originating from the LIEF-extracted feature vector representation of Windows PE files. This uses the gradient information data to guide the perturbation towards the opposite class. The decision to use FGSM in this stage is based on the recent study of [6] that has compared the accuracy performance of FGSM, PGD, LowProFool and DeepFool by using them to train a deep neural ensemble model with Adversarial Training. Notably, the evaluation that has been done in several Android malware detection and network traffic intrusion detection problems has shown that the decision model trained with FGSM achieves, in general, higher accuracy. So, based on these results, we also evaluated the performance of FGSM in this study.

In short, the proposed method takes as input a training set $\mathcal{D} = \{(\mathbf{x_i}, y_i)\}_{i=1}^{n}$ of n Windows PE files, where each $\mathbf{x_i} \in \{0, 1, \ldots, 255\}^*$ is the binary code of a windows PE file and y_i is the real class (*goodware* or *malware*) associated with $\mathbf{x_i}$. The method, as schematized in Figure 1, learns the DNN-based Windows PE malware detection model in four steps:

1. The LIEF-extracted representation $\mathcal{D}^{lief}$ of $\mathcal{D}$ is obtained. To this aim, LIEF is used to generate a tabular example $\mathbf{x_i^{lief}} \in \mathbb{R}^{2381}$ for each Windows PE file $\mathbf{x_i}$ with $(\mathbf{x_i}, y_i) \in \mathcal{D}$. Formally, $\mathcal{D}^{lief} = \{(\mathbf{x_i^{lief}}, y_i) \in \mathbb{R}^{2381} \times \{goodware, malware\} | \forall (\mathbf{x_i}, y_i) \in \mathcal{D}\}$.
2. The initial DNN model $DNN_\theta \colon \mathbb{R}^{2381} \mapsto \{goodware, malware\}$ is trained with parameter θ estimated on $\mathcal{D}^{lief}$.
3. The adversarial set $\mathcal{A}^{lief}$ is produced by using FGSM with the data perturbation threshold ϵ. Specifically, the adversarial samples of $\mathcal{A}^{lief}$ are produced from the malicious samples of $\mathcal{D}^{lief}$ that are originally labelled as *malware* to evade DNN_θ.

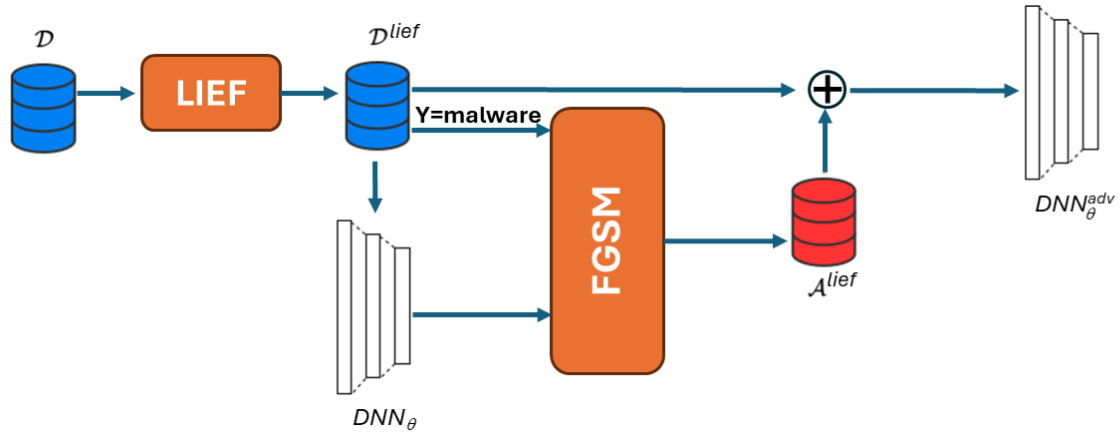


Figure 1: Schema of the Adversarial Training method

4. The final DNN model $DNN_\theta^{adv}: \mathbb{R}^{2381} \mapsto \{goodware, malware\}$ is trained with parameter θ estimated on $\mathcal{D}^{lief} \bigcup \mathcal{A}^{lief}$.

In the experimentation of this study, we evaluate the accuracy performance of both DNN_θ and DNN_θ^{adv} on the collection of real Windows PE files collected in [2], as well as the robustness of both deep neural models against the repository of realistic Windows PE adversarial malware created in [2].

4. Evaluation study and discussion

The performance of the Adversarial Training method was evaluated on a repository that collects both Windows PE files and adversarial Windows PE files. These experiments mainly aimed to explore the ability of Adversarial Training performed with FGSM of gaining accuracy in disentangling real Windows PE goodware from real Windows PE malware, as well as robustness in the presence of realistic adversarial Windows PE malware that were produced with state-of-the-art Windows PE attack methods such as FGSM (padding + slack), Full DOS, Extend, Shift and GAMMA. The Windows PE file repository and the adopted experimental set-up are presented in Sections 4.1 and 4.2, respectively. The implementation details of the proposed Adversarial Training method are reported in Section 4.3. The results are illustrated in Section 4.4.

4.1. Windows PE file repository

We considered two distinct Windows PE file repositories for the training and evaluation stages. In particular, the training stage was performed with the BODMAS repository [25]. This is a publicly available repository of Windows PE files with around one hundred and thirty-four thousand samples collected between 2019 and 2020 from a security company's internal database. Specifically, the BODMAS repository contains 57,293 Windows PE malware and 77,142 Windows PE goodware. The binary files are available in the repository for malware only. However, the pre-extracted features obtained through the static analyser LIEF are publicly available for all the collected samples recorded in the BODMAS repository.

The evaluation stage was performed with the Windows PE repository that has been recently created in [2]. This repository contains 27,035 real Windows PE files: 13,494 goodware and 13,541 malware. Goodware were selected from the public PE Malware Machine Learning (PEEML) repository, so that 980 goodware files were recorded in 2017, and 12514 goodware files were recorded in 2018. Notably, there is no overlap with Windows PE files recorded in the BODMAS repository. Malware files were selected from VirusShare so that 6459 malware were recorded in 2021, 83 in 2022 and 6999 in 2023. Selected malware were distributed as follows: Trojan (55%), Virus (15%), General (26%), Worms (0.9%) Ransomware

(0.7%), Adware (2.1%), Backdoor (0.3%). In addition, the repository contains: 4384 adversarial malware created using FGSM(padding+slach), 6115 adversarial malware created using Full DOS, 7951 adversarial malware created using Extend, 3423 adversarial malware created using Shift, and 6857 adversarial malware created using GAMMA. All the adversarial malware recorded in the repository are realistic, i.e., they preserve their malicious behaviours. In addition, they were produced by the authors of [2] manipulating each counterpart malware recorded in the same repository to evade the MalConv model.

4.2. Experimental set-up

For the experimentation, we used all Windows PE files recorded in the BODMAS repository (along their LIEF-extracted features) to train the initial DNN, create the adversarial set, and train the DNN with Adversarial Training. In the following, we denote the DNN trained with the sample created for the real Windows PE files only as DNN, while the DNN trained with the Adversarial Training method by accounting for the adversarial samples produced with FGSM as DNN^{fgsm}. According to [26], the FGSM method should be used with the perturbation value set as a small value (tentatively between 0 and 1). In this study, we set the perturbation threshold equal to 0.025, in order to scale the noise and ensure that perturbations are small enough to remain undetected to the human eye, but large enough to fool the attacked neural model.

For evaluating the accuracy and robustnesses of both DNN and DNN^{fgsm} we measured the Overall Accuracy (OA), Precision (P), Recall (R) and F1 score (F1) of both models on the version of the evaluation repository populated with the real Windows PE goodware and malware, as well as the version of the same evaluation repository augmented with the realistic adversarial Windows PE malware as described in [2]. In the following, we denote the original repository of real Windows PE files as REAL and the repositories augmented with the realistic adversarial Windows PE malware produced with FGSM (padding + slack) as REAL + FGSM (padding + slack), Full DOS as REAL + Full DOS, Extend as REAL + Extend, Shift as REAL + Shift and GAMMA as REAL + GAMMA, respectively.

4.3. Implementation details

The code[1] used to perform the experimental study was implemented in Python 3 using the high-level neural network API PyTorch. LIEF (version 0.9)[2] was used to extract the feature input space of the deep neural architecture. This architecture was implemented with three Fully Connected (FC) layers with 512, 128 and 8 neurons and one output layer with 2 neurons, respectively. The output probabilities were obtained using the softmax activation function in the last layer. The Tanh activation function was used in all the other layers. Two Batch Normalization layers were placed before the 1st FC layer and between the 2nd and 3rd FC layers. The DNN was trained with a batch size equal to 1024 and a maximum number of epochs equal to 150. An early-stopping approach was used to stop the model earlier than the maximum number of epochs allowed if it did not improve the validation loss. All the architecture parameters described above were set as described by [24], who showed that this architecture outperforms several AI-based decision models in Windows PE malware detection problems. The input space was transformed with Quantile Transformer as implemented in Scikit-learn library [3] to obtain features that follow a normal distribution and reduce the impact of outliers. Finally, the FGSM algorithm was used as implemented in the Adversarial Robustness Toolbox library[4].

4.4. Results

Table 1 reports the accuracy performance of both DNN and DNN^{fgsm} models as they were measured in the evaluation settings: REAL, REAL + FGSM (padding + slack), REAL + Full DOS, REAL + Extend, REAL + Shift and REAL + GAMMA, respectively. These results support the conclusions already reported in

[1]The code is available at https://github.com/ll2909/lobascioITASEC25
[2]https://lief.re/
[3]https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.QuantileTransformer.html
[4]https://adversarial-robustness-toolbox.readthedocs.io/

Table 1
Accuracy performance (Overall Accuracy – OA, Precision – P, Recall – R and F1 score – F1 of of both DNN and DNN^{fgsm} measured on REAL, REAL + FGSM (padding + slack), REAL + Full DOS, REAL + Extend, REAL + Shift and REAL + GAMMA. The best results are in bold

test set	DNN				DNN^{fgsm}			
	OA	P	R	F1	OA	P	R	F1
REAL	91.90	**99.60**	84.17	91.23	**93.77**	98.98	**88.49**	**93.44**
REAL + FGSM (padding + slack)	91.59	96.85	87.73	92.06	**93.76**	**97.56**	**91.17**	**94.26**
REAL + Full DOS	91.07	95.36	88.65	91.88	**93.68**	**97.06**	**91.90**	**94.41**
REAL + Extend	91.73	96.14	89.69	92.80	**93.70**	**96.79**	**92.57**	**94.63**
REAL + Shift	91.68	97.38	87.10	91.95	**93.66**	**97.60**	**90.67**	**94.01**
REAL + GAMMA	88.22	**89.91**	88.47	89.18	**89.64**	89.71	**91.60**	**90.65**

[1, 2] that the black-box evasion method GAMMA is more effective than the white-box evasion methods FGSM (padding + slack), Full DOS, Extend and Shift. In fact, both DNN and DNN^{fgsm} achieve the lowest OA and F1 in REAL + GAMMA. On the other hand, these results show that the Adversarial Training strategy completed with the tabular samples generated with FGSM improves the accuracy performance of the DNN model trained for Windows PE malware detection. In particular, the overall improvement is achieved with the Adversarial Training method in all testing scenarios, i.e. REAL that includes real Windows PE files only, as well as REAL + FGSM (padding + slack), REAL + Full DOS, REAL + Extend, REAL + Shift and REAL + GAMMA, which include real Windows PE files and realistic adversarial Windows PE malware generated with literature evasion methods. Notably, the performances achieved in the REAL setting provide an assessment of the gain in model accuracy, while the performances achieved in the REAL + FGSM (padding + slack), REAL + Full DOS, REAL + Extend, REAL + Shift and REAL + GAMMA settings provide an assessment of the gain achieved in model robustness. Finally, a few exceptions are observed with respect to the evaluation of Precision – P. In particular, values of P measured for DNN are higher than values of P measured for DNN^{fgsm} in both REAL and REAL + GAMMA settings. This means that training the deep neural model by accounting for samples produced with FGSM in the LIEF-representation of Windows PE malware can sometimes lead the decision model to see a higher number of malicious behaviours in the evaluation stage. Hence, the decision model evaluated in this setting detects a higher number of True Malicious Behaviours, but this is coupled with a higher number of False Malicious Behaviours. In any case, the overall performance measured with OA and F1 shows that the trade-off between increasing the number of True Malicious Behaviours at the cost of increasing the number of False Malicious Behaviours is still in favour of the use of the Adversarial Training method also based on the performances observed in the REAL and REAL + GAMMA settings.

To complete this evaluation study, we used SHAP [27] to explain how the Adversarial Training method changed the effect of LIEF-extracted input features on decisions produced with both DNN and DNN^{fgsm}, respectively. SHAP is a well-known, post-hoc, XAI technique that has been recently used in several cybersecurity domains to gain accuracy [17] or explain decision model behaviours [6, 17]. It is based on a theoretic game that measures the effect of each input feature on the decision produced from the model to predict the class of a sample. This effect is measured as the average marginal contribution of the feature value for all alternative decisions. Specifically, we used SHAP[5] to explain decisions produced on malicious Windows PE files. Figures 2 and 3 show the Shapley values measured for the decisions produced from both DNN and DNN^{fgsm}, respectively, for the batch of real Windows PE malware of the considered testing set, and the batches of realistic windows PE malware produced with FGSM (padding + slack), Full DOS, Extend, Shift and GAMMA, respectively. Both Figures show the top-10 features, ranked according to the average Shapley value, of both models in each malicious batch.

In particular, Figure 2 shows that the DNN model is mainly driven by the LIEF-extracted features *Data Directory 9* and *Imports 103*. Both are the top-two features seen to recognize real Windows PE

[5]Shapley values were obtained with shap.DeepExplainer https://shap.readthedocs.io/en/latest/generated/shap.DeepExplainer.html

malware, as well as adversarial Windows PE malware produced with FGSM (padding + slack), Full DOS and GAMMA. On the other hand, *Data Directory 9* falls to fourth, while *Header info 41* and *Header info 1* rise to first and second place, respectively, in the SHAP ranking of the adversarial Windows PE malware produced with both Extend and Shift. Instead, neither *Header info 41* nor *Header info 1* appear in the top-ten features seen to recognize real Windows PE malware and realistic adversarial Windows PE malware produced by FGSM (padding + slack), Full DOS and GAMMA. Focusing the attention on Extend and Shift, we note that there is a high intersection in the top-ten features seen by the DNN model for both categories of adversarial malware except for the LIEF-extracted feature *Headers info 9* that appears in the top-ten ranking of Extend and the LIEF-extracted feature *General File Info 7* that appears in the top-ten ranking of Shift. This suggests that the DNN was trained seeing high similarity between adversarial Windows PE malware generated by both these two evasion methods.

On the other hand, Figure 3 shows that the Adversarial Training methods actually changed the decision model by allowing the LIEF-extracted feature *Header Info 11* to gain importance in all decisions regarding both real Windows PE malware and all categories of adversarial Windows PE malware considered in this study. The only exception is observed with FGSM (padding + slack), where the top-ranked LIEF-extracted feature is *Imports 103*, but the LIEF-extracted feature *Header Info 11* is still the runner-up. On the other hand, the LIEF-extracted feature *Data Directory 9* is still important, but it falls to the second on third position. Another interesting consideration regards the SHAP feature rankings of DNN^{fgsm} for Extend and Shift. These two rankings have more differences between them than their counterparts shown in Figure 2 for DNN. This suggests that the Adversarial Training method allows DNN^{fgsm} to see better the differences between the adversarial Windows PE malware produced with these two evasion methods. Finally, we note that the Shapley values computed for decisions produced with DNN^{fgsm} range in smaller intervals with a lower number of Shapley value outliers than the Shapley values computed for the counterpart decisions produced with DNN. The relationship between this behaviour and a possible higher robustness of explanations produced for deep neural models trained with the Adversarial Training method deserve further investigation in future works.

5. Conclusion

Over the past decade, recent achievements in adversarial learning have shown that adversarial malware create an additional attack vector to the robustness of AI methods developed for anti-malware systems. In this paper, we illustrate the results of an evaluation study of the performance of a deep neural model trained with an Adversarial Training method against five state-of-the-art attack methods defined in literature to produce realistic adversarial Windows PE malware.

The study shows that the Adversarial Training method can take advantage of a general-purpose evasion method – FGSM – that is commonly used with imagery and tabular data to generate adversarial samples in a powerful feature-vector representation of Windows PE files. It uses these samples to train a deep neural model that gains accuracy and robustness in the Windows PE malware detection task. In particular, we show that the deep neural model trained with the Adversarial Training method outperforms the counterpart model trained without Adversarial Training, gaining accuracy on real Windows PE malware and robustness against realistic Windows PE adversarial malware. In addition, we use a post-hoc XAI technique – SHAP – to explain how Adversarial Training changed the ability of the deep neural models to recognize malicious behaviours.

As reported [28], the effectiveness of adversarial training is affected by multiple variables (e.g., classifiers, feature representations). For this reason, we plan to extend this study to explore the performance of further general-purpose evasion methods in the training stage, as well as to additional Windows PE adversarial evasion methods in the evaluation stage. Also, we plan to start the investigation of offensive and defensive strategies for poisoning in Windows PE malware detection, as well as the robustness of explanations produced with defensive strategies. Finally, considering the large number of active Android devices worldwide, which has led to a growing interest in developing solutions to identify malicious applications [29, 30, 31], we plan to extend our work in order to investigate the

impact of adversarial training in Android malware detection models.

Acknowledgments

Luca Lobascio and Giuseppina Andresini are supported by the project FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI (CUP H97G22000210007), under the NRRP MUR program funded by the NextGenerationEU. Annalisa Appice and Donato Malerba are partially supported by project SERICS (PE00000014) under the NRRP MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. The research objectives of this paper are in partial fulfilment of the project AI-CREED (CUP H93C23000880005).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] L. Demetrio, S. E. Coull, B. Biggio, G. Lagorio, A. Armando, F. Roli, Adversarial EXEmples: A survey and experimental evaluation of practical attacks on machine learning for Windows malware detection, ACM Trans. Privacy Secur. 24 (2021) 27:1–27:31. doi:10.1145/3473039.

[2] M. Imran, A. Appice, D. Malerba, Evaluating realistic adversarial attacks against machine learning models for Windows PE malware detection, Future Internet 16 (2024) 1–30. doi:10.3390/FI16050168.

[3] X. Ling, L. Wu, J. Zhang, et al., Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art, Comput. Secur 128 (2023) 1–24. doi:10.1016/j.cose.2023.103134.

[4] A. Ponte, D. Trizna, L. Demetrio, B. Biggio, I. T. Ogbu, F. Roli, Slifer: Investigating performance and robustness of malware detection pipelines, Computers & Security 150 (2025) 104264. doi:https://doi.org/10.1016/j.cose.2024.104264.

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: ICLR 2014, 2014, pp. 1–10. doi:https://doi.org/10.48550/arXiv.1312.6199.

[6] M. Al-Essa, G. Andresini, A. Appice, D. Malerba, PANACEA: a neural model ensemble for cyber-threat detection, Mach. Learn. 113 (2024) 5379–5422. doi:10.1007/S10994-023-06470-2.

[7] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: ICLR 2015, 2015, pp. 1–11. doi:https://doi.org/10.48550/arXiv.1412.6572.

[8] V. Ballet, X. Renard, J. Aigrain, T. Laugel, P. Frossard, M. Detyniecki, Imperceptible adversarial attacks on tabular data, 2019. doi:https://doi.org/10.48550/arXiv.1911.03274.

[9] S. Y. Khamaiseh, D. Bagagem, A. Al-Alaj, M. Mancino, H. W. Alomari, Adversarial deep learning: A survey on adversarial attacks and defense mechanisms on image classification, IEEE Access 10 (2022) 102266–102291. doi:10.1109/ACCESS.2022.3208131.

[10] T. Zhiyi, C. Lei, L. Jie, Y. Shui, A comprehensive survey on poisoning attacks and countermeasures in machine learning, ACM Computing Surveys 55 (2022) 1–35. doi:10.1145/3551636.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: ICLR 2018, 2018, pp. 1–10. doi:https://doi.org/10.48550/arXiv.1706.06083.

[12] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, DeepFool: A simple and accurate method to fool deep neural networks, in: CVPR 2016, IEEE, 2016, pp. 2574–2582. doi:10.1109/CVPR.2016.282.

[13] P. Chen, H. Zhang, Y. Sharma, J. Yi, C. Hsieh, Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models, in: AISec 2017, 2017, p. 15–26. doi:10.1145/3128572.3140448.

[14] N. Martins, J. M. Cruz, T. Cruz, P. Henriques Abreu, Adversarial machine learning applied to intrusion and malware scenarios: A systematic review, IEEE Access 8 (2020) 35403–35419. doi:10.1109/ACCESS.2020.2974752.

[15] M. Macas, C. Wu, W. Fuertes, Adversarial examples: A survey of attacks and defenses in deep learning-enabled cybersecurity systems, Expert Syst. Appl. 238 (2024) 1–33. doi:10.1016/j.eswa.2023.122223.

[16] F. Pierazzi, F. Pendlebury, J. Cortellazzi, L. Cavallaro, Intriguing properties of adversarial ML attacks in the problem space, in: SP 2020, IEEE, 2020, pp. 1332–1349. doi:10.1109/SP40000.2020.00073.

[17] M. Al-Essa, G. Andresini, A. Appice, D. Malerba, An XAI-based adversarial training approach for cyber-threat detection, in: CyberSciTech 2023, IEEE, 2022, pp. 1–8. doi:10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927842.

[18] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, J. Keshet, Adversarial examples on discrete sequences for beating whole-binary malware detection, CoRR (2018). doi:https://doi.org/10.48550/arXiv.1802.04528.

[19] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, C. K. Nicholas, Malware detection by eating a whole EXE, in: AAAI 2018 Workshops, 2018, pp. 1–8. URL: https://cdn.aaai.org/ocs/ws/ws0432/16422-75958-1-PB.pdf.

[20] M. Kozák, M. Jurecek, M. Stamp, F. D. Troia, Creating valid adversarial examples of malware, J Comput Virol Hack Tech (2024) 1–15. doi:10.1007/s11416-024-00516-2.

[21] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, A. Armando, Functionality-preserving black-box optimization of adversarial Windows malware, IEEE Trans. Inf. Forensics Secur. 16 (2021) 3469–3478. doi:10.1109/TIFS.2021.3082330.

[22] M. Şandor, R. M. Portase, A. Coleşa, Ember feature dataset analysis for malware detection, in: ICCP 2023, 2023, pp. 203–210. doi:10.1109/ICCP60212.2023.10398693.

[23] O. Barut, T. Zhang, Y. Luo, P. Li, A comprehensive study on efficient and accurate machine learning-based malicious PE detection, in: CCNC 2023, 2023, pp. 632–635. doi:10.1109/CCNC51644.2023.10060214.

[24] C. Connors, D. Sarkar, Machine learning for detecting malware in pe files, 2022. doi:10.48550/arXiv.2212.13988.

[25] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh, G. Wang, Bodmas: An open dataset for learning based temporal analysis of pe malware, in: 2021 IEEE Security and Privacy Workshops (SPW), IEEE, 2021, pp. 78–84. doi:10.1109/SPW53761.2021.00020.

[26] T. Bai, J. Luo, J. Zhao, B. Wen, Q. Wang, Recent advances in adversarial training for adversarial robustness, in: 30th International Joint Conference on Artificial Intelligence, IJCAI 2021, IJCAI.ORG, 2021, pp. 4312–4321. doi:10.24963/ijcai.2021/591.

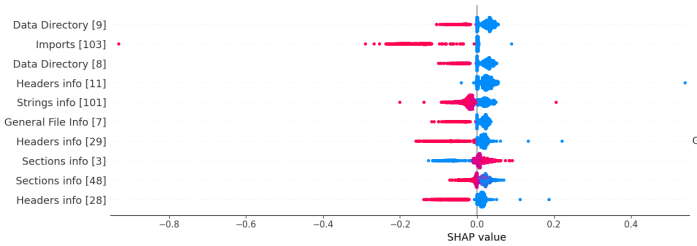
[27] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: 31st International Conference on Neural Information Processing Systems, NIPS 2017, NIPS, Curran Associates Inc., 2017, pp. 4768–4777. doi:10.5555/3295222.3295230.

[28] H. Bostani, J. Cortellazzi, D. Arp, F. Pierazzi, V. Moonsamy, L. Cavallaro, On the effectiveness of adversarial training on malware classifiers, 2024. doi:https://doi.org/10.48550/arXiv.2412.18218. arXiv:2412.18218.

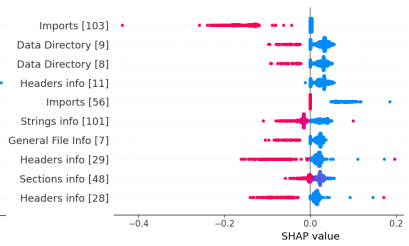
[29] A. Guerra-Manzanares, Machine learning for android malware detection: Mission accomplished? a comprehensive review of open challenges and future perspectives, Computers & Security 138 (2024) 103654. doi:https://doi.org/10.1016/j.cose.2023.103654.

[30] S. McFadden, Z. Kan, L. Cavallaro, F. Pierazzi, The impact of active learning on availability data poisoning for android malware classifiers, in: Proc. of the 2nd Workshop on Recent Advances in Resilient and Trustworthy MAchine learning-driveN systems (ARTMAN), 2024.

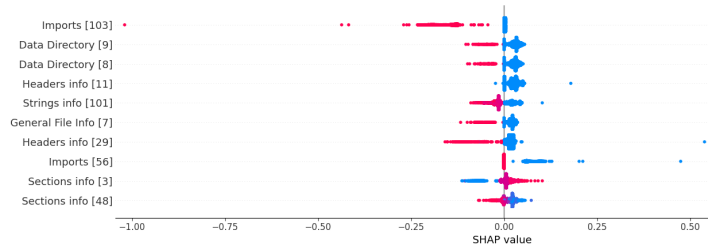
[31] G. Andresini, A. Appice, D. Malerba, Dealing with Class Imbalance in Android Malware Detection by Cascading Clustering and Classification, Springer International Publishing, Cham, 2020, pp. 173–187. doi:10.1007/978-3-030-36617-9_11.



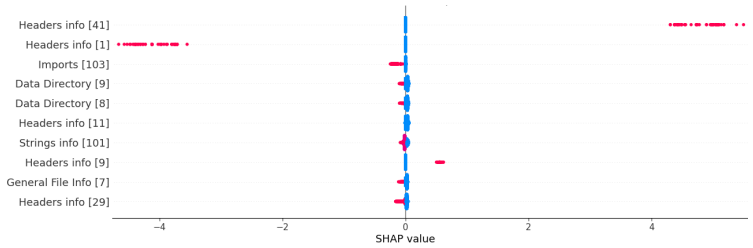
(a) DNN - malware



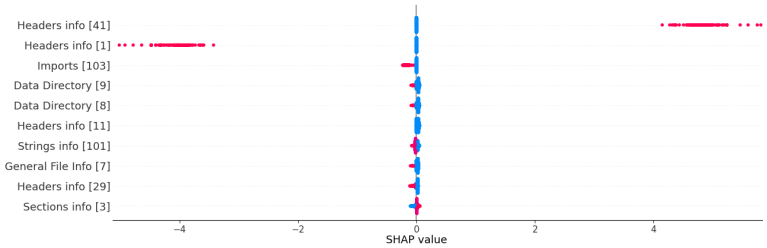
(b) DNN - FGSM (padding + slack)



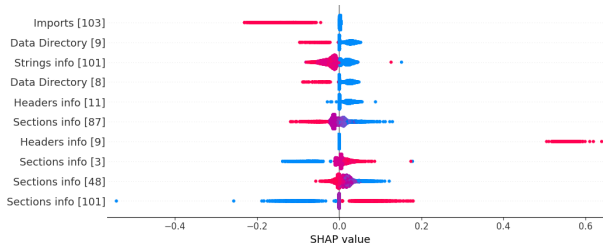
(c) DNN - Full DOS



(d) DNN - Extend

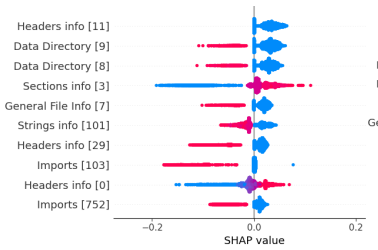


(e) DNN - Shift

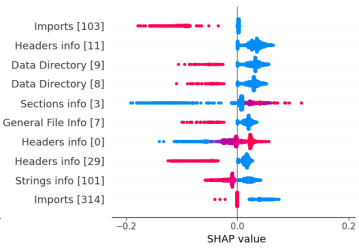


(f) DNN - GAMMA

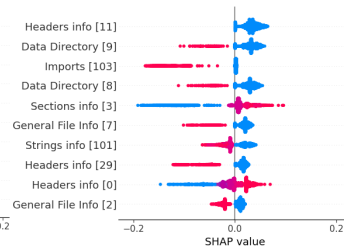
Figure 2: Top-10 LIEF-extracted features of DNN according to Shapley values measured for real and adversarial Windows PE malware



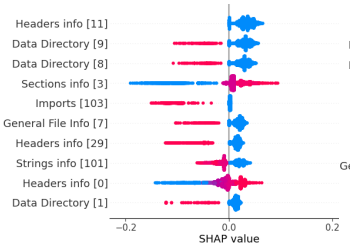
(a) DNN^{fgsm} - malware



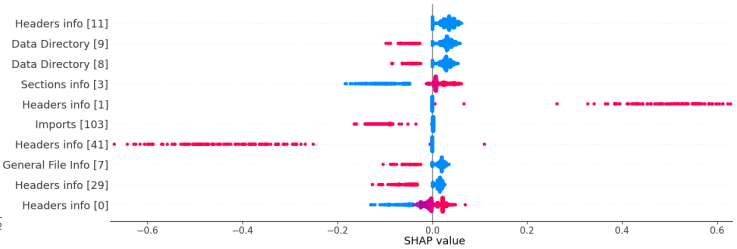
(b) DNN^{fgsm} - FGSM (padding + slack)



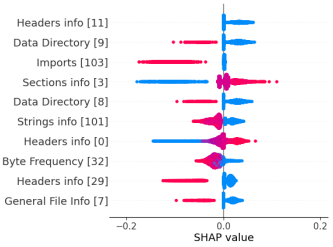
(c) DNN^{fgsm} - Full DOS



(d) DNN^{fgsm} - Extend



(e) DNN^{fgsm} - Shift



(f) DNN^{fgsm} - GAMMA

Figure 3: Top-10 LIEF-extracted features of DNN^{fgsm} according to Shapley values measured for real and adversarial Windows PE malware