future internet

MDPI

Article

Evaluating Realistic Adversarial Attacks against Machine Learning Models for Windows PE Malware Detection

Muhammad Imran [1,*], Annalisa Appice [1,2,*] and Donato Malerba [1,2]

[1] Department of Computer Science, University of Study of Bari Aldo Moro, Via Orabona, 4, 70125 Bari, Italy; donato.malerba@uniba.it
[2] Consorzio Interuniversitario Nazionale per l'Informatica—CINI, Via Orabona, 4, 70125 Bari, Italy
* Correspondence: muhammad.imran@uniba.it (M.I.); annalisa.appice@uniba.it (A.A.)

Abstract: During the last decade, the cybersecurity literature has conferred a high-level role to machine learning as a powerful security paradigm to recognise malicious software in modern anti-malware systems. However, a non-negligible limitation of machine learning methods used to train decision models is that adversarial attacks can easily fool them. Adversarial attacks are attack samples produced by carefully manipulating the samples at the test time to violate the model integrity by causing detection mistakes. In this paper, we analyse the performance of five realistic target-based adversarial attacks, namely Extend, Full DOS, Shift, FGSM padding + slack and GAMMA, against two machine learning models, namely MalConv and LGBM, learned to recognise Windows Portable Executable (PE) malware files. Specifically, MalConv is a Convolutional Neural Network (CNN) model learned from the raw bytes of Windows PE files. LGBM is a Gradient-Boosted Decision Tree model that is learned from features extracted through the static analysis of Windows PE files. Notably, the attack methods and machine learning models considered in this study are state-of-the-art methods broadly used in the machine learning literature for Windows PE malware detection tasks. In addition, we explore the effect of accounting for adversarial attacks on securing machine learning models through the adversarial training strategy. Therefore, the main contributions of this article are as follows: (1) We extend existing machine learning studies that commonly consider small datasets to explore the evasion ability of state-of-the-art Windows PE attack methods by increasing the size of the evaluation dataset. (2) To the best of our knowledge, we are the first to carry out an exploratory study to explain how the considered adversarial attack methods change Windows PE malware to fool an effective decision model. (3) We explore the performance of the adversarial training strategy as a means to secure effective decision models against adversarial Windows PE malware files generated with the considered attack methods. Hence, the study explains how GAMMA can actually be considered the most effective evasion method for the performed comparative analysis. On the other hand, the study shows that the adversarial training strategy can actually help in recognising adversarial PE malware generated with GAMMA by also explaining how it changes model decisions.

Keywords: Windows PE Malware; adversarial attacks; integrity violation; transferability; adversarial training; explainable artificial intelligence

Citation: Imran, M.; Appice, A.; Malerba, D. Evaluating Realistic Adversarial Attacks against Machine Learning Models for Windows PE Malware Detection. *Future Internet* **2024**, *16*, 168. https://doi.org/10.3390/fi16050168

Academic Editor: Francesco Buccafurri

Received: 14 March 2024
Revised: 7 May 2024
Accepted: 9 May 2024
Published: 12 May 2024

1. Introduction

The Portable Executable (PE) files belonging to the family of Microsoft Windows operating systems (i.e., Windows PE files) require ad hoc malware detection methods, as they adopt a file format specific to Windows operating systems. Windows PE malware is currently trending among prominent malware types. In fact, in 2023, Kaspersky's systems detected almost 125 million malicious files, with Windows as the primary target for cyber attacks. Specifically, Windows accounted for 88% of all malware-filled data detected daily (https://www.kaspersky.com/about/press-releases/2023_rising-threats-cybercriminals-unleash-411000-malicious-files-daily-in-2023, accessed on 30 April 2024).

These statistics explain the continuous improvement in the cybersecurity literature to keep security solutions up to date in the Windows PE domain. In particular, machine learning methods have dominated the literature on Windows PE malware detection during the last decade [1]. While traditional signature-based detection methods, which recognise Windows PE malware by comparing its signature with signatures of known malware samples, fail in detecting new malware samples, machine learning decision models may allow the detection of some new malware variants (see [2,3] for recent surveys on the topic). On the other hand, several recent studies have demonstrated that machine learning models are inherently vulnerable to adversarial attacks (i.e., manipulations to input data to deceive the model). Since the pioneering study on adversarial attacks [4], the literature has mainly focused on investigating adversarial attacks in computer vision. However, the topic started attracting attention in cybersecurity, particularly in Windows PE malware detection [5,6]. Notably, adversarial attack methods synthesised for imagery and tabular data [7] cannot be used for Windows PE files since realistic adversarial Windows PE malware files must preserve a Windows PE-compatible format (format-preserving property), obtain an executable file (executable-preserving property) and maintain the malicious nature of the executable file (maliciousness-preserving property). The format-preserving property implies that adversarial modifications must generate a binary file that complies with the standard format of Windows PE files. The executable-preserving property ensures that adversarial Windows PE malware loads all the necessary data and can be executed. The maliciousness-preserving property means that adversarial Windows PE malware maintains the same malicious behaviour (e.g., modifying registry items and deleting or encrypting files) as the original Windows PE malware. On the other hand, accounting for these challenges, various adversarial attack methods have been formulated recently in Windows PE malware detection research since 2017. The authors of [8] recently developed a taxonomy for the existing literature on adversarial Windows PE malware from different viewpoints, i.e., an adversary's knowledge, an adversary's space, target malware detection, and the attack strategy. However, while the authors of [8] provided an exhaustive review of the state-of-the-art research efforts on adversarial attacks against Windows PE malware detection, this study illustrates the results of an empirical evaluation of the evasion ability of some representative attack methods producing real adversarial Windows PE malware.

In particular, the evaluation study of this article was conducted using two well-known machine learning models, MalConv [9] and LGBM [10]. Specifically, MalConv is a byte-based Convolutional Neural Network (CNN) model. It takes the raw bytes from Windows PE files and uses an embedding layer, followed by several convolutional layers, to learn relevant features for the final classification that is performed with a sigmoid function. LGBM is a Light Gradient-Boosted Decision Tree Model trained on semantically rich features (engineered features) extracted through the static analysis of PE files. Notably, MalConv has recently been considered the target model to fool for several Windows PE malware attack methods [5,11,12] due to the good performance achieved by producing a binary code-based decision model. In fact, MalConv allows us to eliminate the use of any complex engineering step performed through the static analysis of the binary code or the dynamic analysis of the software behaviour. On the other hand, a recent study [13] compared the accuracy performance of several machine learning methods (i.e., Random Forest, XGBoost, LGBM and MalConv) used in Windows PE malware detection problems by showing that the decision model learned through LGBM is not only the most accurate model but also the fastest. In this study, as attack methods, we considered the following: Extend [5], Full DOS [5], Shift [5], FGSM padding + slack [14] and GAMMA [15]. The implementation of all of these attack methods is publicly available in the secm-malware library [16]. We note that the selected machine learning models and attack methods have been used recently in the evaluation study conducted in [5]. Extend, Full DOS, Shift, and FGSM padding+slack are white-box attacks, while GAMMA is a black-box attack. Similarly to [5], we used attack methods against MalConv and explored the transferability of adversarial malware from MalConv to LGBM. Our study confirms the conclusions drawn by [5]. The LGBM model remains quite robust to the adversarial malware generated by attacking the MalConv model.

The white-box attack Extend is more effective against MalConv, while the black-box attack GAMMA is more effective against LGBM, with Extend serving as the runner-up. Differently from [5], we used a larger and more recent dataset for our evaluation. In addition, we performed a distance-based analysis of adversarial Windows PE malware to explore possible relationships between the amount of changes introduced in the malware via the attack method and the evading ability of the produced adversarial malware. Finally, to the best of our knowledge, this is the first study that used a post hoc, Explainable Artificial Intelligence (XAI) technique, SHAP [17], to explain the effect of changes in the engineered features on the LGBM decisions, depending on the different types of attack methods considered. Specifically, XAI refers to the ability of a machine learning model to deliver human-understandable explanations for the decisions made via the model. Therefore, the objective of an XAI method is to make the decision-making process of a machine learning model interpretable and transparent. In particular, SHAP is a popular XAI method that computes Shapley values to explain the impact that each input feature has in generating every prediction delivered by the explained model.

As an additional contribution, we evaluated the performance of the adversarial training strategy performed when learning the LGBM model using the realistic adversarial samples produced with the attack methods considered in the study. The adversarial training strategy [4] is commonly considered one of the main defence techniques to resist adversarial attacks [18–21]. In this evaluation study, we applied the adversarial training strategy by extending the original training set with the adversarial malware files generated with Extend, Full DOS, Shift and FGSM padding + slack. We compared the performance of both the original LGBM model and the adversarial LGBM model on both the original test set and the extended test set (i.e., the original test set extended with the realistic adversarial Windows PE malware generated with the same attack method considered to apply the adversarial training strategy). Finally, we used SHAP to explain how the use of the adversarial training strategy changed the model decisions concerning malware. The results show that adversarial training tested with GAMMA can strengthen the robustness of the decision model, decreasing the false negative rate and increasing the false positive rate.

In short, the contributions and achievements of this article are as follows:

- The evaluation study of the evasion ability of Windows PE attack methods performed in [5] was extended by considering a larger dataset of more recent Windows PE files with respect to the one used in [5]. We made the dataset that was prepared to conduct our evaluation study publicly available for future research studies. This evaluation study confirms the conclusions drawn in [5] that LGBM outperforms MalConv, while GAMMA generates a higher number of realistic adversarial Windows PE malware files that are able to fool both MalConv and LGBM.
- A new interpretative analysis was performed to explain how the attack methods considered in this study can change the Windows PE malware files to fool the decisions of machine learning models. This analysis shows that the less effective attack methods of the performed evaluation, i.e., Full DOS and FGSM padding + slack, produce the adversarial malware files closest to the binary files of the original counterpart malware. On the other hand, the most effective attack method of this study, i.e., GAMMA, produces the adversarial malware files that are the furthest from the binary files of the original counterpart malware. In addition, this study explains how each attack method fools the decision model produced with LGBM, which is the most accurate machine learning method of this study. Specifically, it discloses which input features of the decision model change importance in the decisions produced for the study's adversarial malware files with respect to the decisions made using their original counterpart malware files.
- The adversarial training strategy was used as a defence adversarial learning approach to train a new LGBM model by incorporating the adversarial Windows PE malware files generated with the attack methods in both the training stage conducted to obtain the decision model and the evaluation stage conducted to measure the accuracy performance of the decision model. The evaluation results show that the use of the adversarial training strategy with GAMMA-produced samples can be an effective

strategy to strengthen the LGBM model against this attack type. In addition, the study explains how the adversarial training strategy changes the decisions of the LGBM model in this case. We note that exploring the explainability of both machine learning and adversarial learning behaviours is nowadays crucial in gaining the trust of cybersecurity stakeholders in such technologies.

The present study was boosted by the fact that recent growth in the Windows PE malware activity has been observed despite several recent machine learning models having achieved superior performance in detecting Windows PE malware. A factor that may have contributed to the recent boom in Windows PE malware is that machine learning models may be vulnerable to adversarial samples [22]. This provides the ethical foundation for adversarial learning studies in malware detection since deeper knowledge of adversarial malware can be mandatory to mitigate the consequences of attackers that use adversarial learning to write malicious code. In particular, thanks to the exploration of the performance of state-of-art methods for generating adversarial Windows PE malware on a larger scale and the explanation of how and why adversarial Windows PE malware generated with these methods can fool effective machine learning models, this study can be used as a springboard for facilitating defence actions to increase machine learning models' robustness with respect to adversarial sample vulnerabilities. In this regard, this paper makes the results of an evaluation study on the defensive capacity of the adversarial training strategy used with adversarial Windows PE available to cybersecurity practitioners who wish to understand how to improve the performance of the defence security of anti-malware systems. In addition, the attention of this study to the explainability analysis of both the performance of the literature's methods to generate adversarial Windows PE malware and the performance of the adversarial training strategy to improve the robustness of machine learning models constitutes a step forward in the called-for transparency of machine learning models used in multiple fields that comprise the cybersecurity field. Notably, this explainability demand has also been encoded by the European Union into law concerning the individual's right to an explanation when decisions made via automated systems significantly affect that individual (refer to the General Data Protection Regulation Article 22 and Recital 71 (https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679, accessed on 30 April 2024).

The paper is organised as follows. Section 2 introduces the layout of PE files and describes the main learning frameworks for PE malware detection. Section 3 describes both the materials and methods of this study (attack methods, machine learning models, SHAP and adversarial training). Section 4 presents the dataset and evaluation metrics used in the evaluation study. Section 5 illustrates the study's results and analysis. Finally, Section 6 draws conclusions and sketches future research directions. Specifically, future works include the following: the continuation of this study to explore the poisoning issue with the inputs of adversarial samples modified during training; the investigation of the performance of the adversarial training strategy as a defence mechanism against attacks produced using re-coding/armouring adversary tactics; and a study of the performance attack methods defined in the field of Android malware detection.

2. Background

In this section, we illustrate the background of this study. Specifically, Section 2.1 introduces some basic concepts and terminology that are used in this article, and it refers to both machine learning and adversarial learning. Section 2.2 describes basic concepts concerning Windows PE files. Section 2.3 overviews datasets and services available to conduct research studies on Windows PE malware detection problems. Finally, Section 2.4 introduces the background on attack and defence in adversarial machine learning.

2.1. Machine Learning Concepts and Terminologies

The term *machine learning* [23] denotes a sub-field of artificial intelligence, which studies and synthesises methods that can learn a decision model from data so that the

learned model allows us to generalise decisions to unseen data. *Classification* is one of the tasks most frequently performed using machine learning methods. The objective of a *classification method* is to learn a decision model, also called a *classification model*, to predict the correct categorical class of given input data. In this article, we consider *binary decision models* produced in a *numeric supervised setting*. Specifically, we were interested in learning about a decision model that is a function, $f : \mathbb{R}^m \mapsto \{0, 1\}$, produced from a set of labelled, numeric samples, $Tr \subseteq \mathbb{R}^m \times \{0, 1\}$, which is here referred to as a *training set*. In this study, each sample, $(\mathbf{x}, y) \in Tr$, represents a Windows PE file that is described using a vector, $\mathbf{x}$, of m real-valued input features and one-to-one associated with a binary class, y. The binary class assumes a value equal to 0 for goodware files and 1 for malware files. A decision model, f, that is learned from a training set, Tr, can be used to predict the unseen class of any new sample, given the observation of the vector, $\mathbf{x} \in \mathbb{R}^m$, of its input feature values. Let us denote $\hat{y} = f(\mathbf{x})$ as the class predicted using f for $\mathbf{x}$. To evaluate the accuracy performance of a decision model, f, we measure the error rate of class predictions that are yielded by using f to predict the class of samples of testing set $Ts \subseteq \mathbb{R}^m \times \{0, 1\}$ with $Tr \cap Ts = \oslash$. In particular, given the decision model, f, the error rate function, $e_f : \mathbb{R}^m \times \{0, 1\} \mapsto \{0, 1\}$, associated with f is defined so that $e_f(\mathbf{x}, y) = 0$ if $y = f(\mathbf{x})$ or 0 otherwise. The higher the accuracy of f, the lower the amount of testing set samples predicted with an error rate equal to zero. Section 4 illustrates several accuracy metrics computed to evaluate the accuracy performance of a decision model by combining cumulative measurements of error rates computed for different groups of predicted samples.

The term *adversarial machine learning* [7] denotes a sub-field of machine learning that studies attacks intended to deceive machine learning methods (*offensive adversarial learning*) and the defence against such attacks (*defensive adversarial learning*). Adversarial samples are carefully and maliciously well-crafted inputs to deceive a target decision model. In the *evasion setting*, the one considered in this article, adversarial samples are samples modified to evade the model being misclassified. For completeness, in the *poisoning setting*, adversarial samples are produced to contaminate the training dataset with data designed to increase errors in the output. When focusing on the evasion setting, attack methods to generate adversarial samples are divided into white-box and black-box attacks. In the *white-box* scenario, the attacker has complete knowledge of the target model internally (e.g., parameter values and architectures used), while in the black-box scenario, the attacker can only query the target model to recover the predicted class for a given sample. Independently of the attack scenario, the more effect an attack method has, the higher the number of adversarial samples generated in the evaluation stage to evade the decision model. *Adversarial training* is one of the defence strategies that aim to train decision models that are more robust to evasion. During the training process, both original and adversarial samples are used to train the final decision function. The more effective the adversarial training strategy, the lower the number of adversarial samples that evade the attacked model but do not evade the new decision model trained with the adversarial training strategy.

Since this article focuses its attention on the use of machine learning for Windows PE malware detection, both the evading ability of machine learning attack methods and the potential of the adversarial training strategy as a means to improve the robustness of machine learning decision models against evasion attacks are mainly considered with respect to the software category of Windows PE files.

2.2. Windows PE Files

The PE file is the standard binary format of DLLs and Executables in the Microsoft Windows family [24]. Its layout contains a Header area, a Section area and the Unmapped Information area. As shown in Figure 1, the Header area contains the DOS Header, DOS Stub, PE Header, PE Optional Header and Section Table. Both the DOS Header and DOS Stub are used for compatibility with the MS-DOS system. The PE Header contains global information about the PE file. For example, it contains the number of sections and the creation time. The PE Optional Header contains information to describe the alignment of

the binary data. Notice that the binary data must be recorded in the executable code by satisfying the constraints that each section of the PE file must start at an offset multiple to that field, and the size of each header must be a multiple of the file alignment. The PE Optional Header contains the offset pointers to find the functions in the Import Table or the Export Table. The Section Table provides global information about Sections such as names, offsets and sizes. The Section area contains the consecutive sections with the executable code (.text) and required data (.data), which include global and static variables. The number of sections in the Section area is not fixed a priori. Additional sections can be used to report the information regarding the address and size of the import table (.idata) and export table (.edata), resource data such as icons and menus (.rsrc), uninitialised data (.bss), extended text on additional linking (.textbss) or relocation information (.reloc). The Unmapped Data Section contains all data that are recorded in the PE file, although these data are never loaded into memory during the program execution. Examples of unmapped data include uninitialised data and debug information.

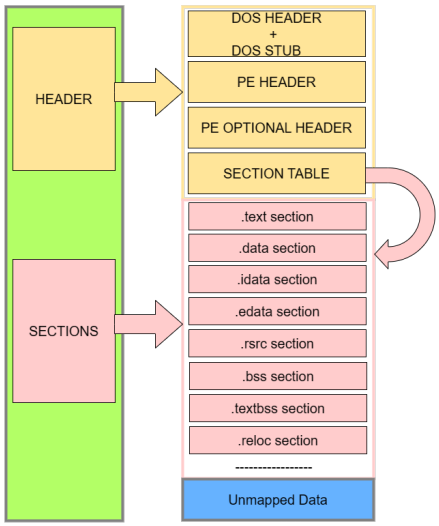


Figure 1. The structure of a PE file. The elements in yellow are included in the Header area. The elements in pink are included in the Section area. The elements in blue are included in the Unmapped Data Section area.

As reported in [8], "*a Windows PE malware is a PE file with a malicious intention, e.g., an executable accessing the system without user permissions, stealing private or confidential information, requiring a ransom*". Several machine learning methods have been studied for Windows PE malware detection in the last decade. They are commonly developed from a sufficient amount of labelled Windows PE files for which an engineering phase is performed to produce the numerical input expected for the machine learning methods. In particular, the developed detection models differ in the type of input features (e.g., byte sequences, strings, grayscale images, API calls and control flow graphs), the feature extraction approaches (e.g., the static analysis of the binary files without running them, the dynamic analysis of the run-time behaviour of executable files run in an isolated environment or hybrid analysis) and the learning methods (e.g., machine learning methods such as Naive Bayes, LGBM, SVM, Random Forest or deep neural networks such as CNNs, Graph Neural Networks or Long Short-Term Networks (LSTM). Extensive surveys of the relevant literature in the field are described in [8,25].

2.3. Windows PE File Collection

There are several online systems that are publicly available to check and, in fewer cases, download Windows PE files. For example, VirusTotal (https://www.virustotal.com/gui/home/upload, accessed on 30 April 2024) provides a free service to analyse and recognise the malicious behaviour of files submitted for a check. However, it provides for-pay

services to obtain recorded files. On the other hand, VirusShare (https://virusshare.com/, accessed on 30 April 2024) and MalShare (https://malshare.com/index.php, accessed on 30 April 2024) are among the most popular, continuously updated repositories of live malicious code. Both repositories provide a free service that can be used to download Windows PE malware for a research scope but no free service to obtain goodware files. We note that VirusShare appeared in 680 Google Scholar citations from 2023, while MalShare appeared in 73 Google Scholar citations from 2023. The query to obtain these statistics was performed on 30 April 2024 on Google Scholar with the keywords "VirusShare" and "MalShare", respectively.

By leveraging the online services described above and some private collections, several datasets have been created recently to conduct Windows PE malware detection studies. The most popular dataset used in the literature for Windows PE malware detection is EMBER [10] with around 1.1 million samples collected from VirusTotal between 2017 and 2018. More recently, researchers have also started using the SoReL dataset [26] with 20 million samples collected between 2017 and 2019. The authors of SoReL do not describe the source of these samples in [26]; however, the repository documentation (https://github.com/sophos/SOREL-20M, accessed on 30 April 2024) reports that a large amount of Windows PE malware recorded in the SoReL repository also appears to be available via VirusTotal. BODMAS [27] is another Windows PE dataset with around one hundred and thirty-four thousand samples collected between 2019 and 2020 from a security company's internal database. Finally, DasMalwerk (http://dasmalwerk.eu/, accessed on 30 April 2024) is a Windows PE repository that contains a collection of Windows PE malware retrieved in late 2018. The total amount of samples collected in DasMalwerk is 104, and 37 of them are contained inside the EMBER dataset. For EMBER, SoReL and BODMAS, pre-extracted features obtained through the static analyser LIEF (https://lief.re/, accessed on 30 April 2024) are publicly available for all the collected samples. However, the EMBER repository does not provide any binary files (for either malware or goodware samples). The SoReL repository provides the disarmed binaries of a few Windows PE samples only. The BODMAS repository provides the binaries of the Windows PE malware files, but it does not provide any binary file for goodware samples. Finally, the DasMalwerk repository, which contains only Windows PE malware files, provides the binaries of the collected files. Although EMBER is one of the oldest repositories, as reported also in [28], EMBER is still used more than SoReL, BODMAS and DasMalwerk in research studies (172 vs. 45 vs. 54 vs. 15 papers citing research studies describing EMBER [10], SoReL [26], BODMAS [27] and DasMalwerk, respectively, in Google Scholar from 2023). Finally, PEMML (https://practicalsecurityanalytics.com/pe-malware-machine-learning-dataset/, accessed on 30 April 2024) is an older Windows PE file repository that was made available by Practical Security Analytics LLC. It contains 201,549 Windows PE files collected before 2018 from VirusShare and MalShare. In this repository, the binary files are publicly available for all samples (both malware and goodware).

For completeness, we also cite Mal-API-2019 [29], which is a dynamic dataset that records API calls of around 7000 samples. Finally, AVAST-CTU [30] is a recently released dataset that contains static and behavioural data collected from around 50 malicious samples. Behavioural features were observed in a sandbox over an extensive period of time.

A summary of the characteristics of the discussed datasets is reported in Table 1.

Table 1. A short description of characteristics of the EMBER, SoReL, BODMAS and DasMalWerk datasets.

Dataset	Malware	Goodware	Collection Time	Binary File Availability
EMBER	400,000	400,000	2017–2018	No
SoReL	9,919,251	9,470,626	2017–2019	Malware
BODMAS	57,293	77,142	2019–2020	Malware
DasMalwerk	104	0	2018	Malware
PEMML	114,737	86,812	Before 2018	Goodware and malware
MAL-API-2019	7107	0	2019	No
AVAST-CTU	48,976	0	2017–2019	No

2.4. Adversarial Attacks and Defences

Regarding adversarial attacks, the authors of [31] reported that different types of adversaries can compromise the effectiveness of machine learning-based decision systems by taking advantage of vulnerabilities that may affect machine learning models. They also provided definitions of different attacks. Specifically, they reported that "*The evasion attacks are the most common type of attacks to machine learning models, with malicious inputs that are craftily modified to force the model to make a false prediction and evade detection*". In addition, they clarified that "*The poisoning attacks differ from the evasion attacks because the inputs are modified during training as the model is trained on contaminated inputs to obtain the desired output*". Several recent surveys [7,32–34] have extensively reviewed the literature related to adversarial attacks on imagery, text and tabular data. The authors of [35] analysed the recent state-of-the-art of research on poisoning attacks against machine learning models. The authors of [36] described a taxonomy of cybersecurity applications and reviewed methods of generating adversarial examples and suitable defences in multiple cybersecurity applications. Focusing on the Windows PE malware detection problems, the authors of [5,8] reviewed the state-of-the-art literature on adversarial attacks against Windows PE malware detection.

On the other hand, the authors of [37] surveyed the defence strategies against adversarial attacks also in PE malware detection. In particular, the adversarial training strategy is one of the mainstream adversarial defences in image analysis. Recent studies [38,39] have explored how the use of the adversarial training strategy can improve the effectiveness of the Windows PE malware detection models that can be trained with machine learning methods. In particular, the authors of [40] have recently described one of the first studies exploring the performance of adversarial training done with realistic, white-box, adversarial PE malware produced with [14] attacking a MalConv model. In this study, we extend the analysis of the performance of the adversarial training strategy to five realistic attack methods.

Finally, the authors of [41] investigated several types of adversarial example attacks on the Android system. The produced adversarial samples are Android malware applications that have been changed by keeping their malicious characteristics and capabilities. This study explores the effect of adversarial example attacks on malware detection systems. In [21], the authors studied the use of the adversarial training strategy, coupled with ensemble learning on Android applications. However, this study considers adversarial samples produced with general-purpose attack methods (e.g., FGSM or PGD) that perturb the handcrafted features of the input space of the machine learning model in place of the raw binaries. Hence, the adversarial samples generated in these studies to perform the adversarial training strategy are unrealistic files, as they may not preserve the executable format with malicious behaviour. The authors of [42] explored the performance of a two-level machine learning model that combines an Ensemble Learning method and a Stacked Generalization method. They show that the proposed model is able to accurately detect recent Android malware. However, this study did not explore the case of adversarial files

generated to fool the machine learning model. The authors of [43] also focused on the Android malware detection task. In particular, this study investigated the use of fuzzy set theory in the generation of synthetic malicious samples that are produced as valid Android applications that preserve their malicious behaviour. These samples are used to mitigate the imbalance condition that commonly occurs in Android malware detection environments where benign samples commonly outnumber malware samples. Notably, the study shows that a deep neural network, trained from the original dataset augmented with the synthetic malware samples, gains accuracy in detecting Android malware. However, this study, similar to [21,42], did not explore the robustness of the machine learning model to possible, realistic adversarial samples created to fool the decision model. The authors of [44] explored how a Markov process-based adversarial model, originally formulated for digital rights management (DRM) apps, can be adapted to detect vulnerable iOS devices and analyse (non-DRM) apps for vulnerabilities that can potentially be exploited. Notably, this study also provided some iOS device security guidelines to reduce the risk of malware attacks. For example, the study suggested that sensitive documents stored on iOS devices should fall under the NSFileProtectionComplete Data Protection Class, which provides secure passcode-based encryption. In addition, it recommended the use of Advanced Encryption Standard keys to generate complex passwords. Finally, it suggested that applications embed the organisation's certificate/public key.

3. Materials and Methods

In this section, we briefly introduce the machine learning models, the adversarial Windows PE generation methods and the XAI technique we used to conduct the evaluation study described in this study. The data used in the evaluation study are described in Section 4.

3.1. Machine Learning Models

We considered two publicly available machine learning models, namely MalConv and LGBM. These pre-trained models were produced for Windows PE malware detection using the labelled PE files recorded in the EMBER dataset [10]. The MalConv model was trained from the raw byte-based representation of PE files, while the LGBM model was trained from the engineered features extracted using the static analysis of binary PE files. Although the binary version of EMBER PE files used to yield the pre-trained MalConv model is not publicly available, the tabular representation of the engineered features used to yield the pre-trained LGBM model is publicly available with the machine learning method code to retrain the model on any new dataset.

MalConv [9] is an end-to-end deep neural model that takes the raw bytes of a PE file as input and predicts the malicious behaviour of the input file. The architecture includes both an embedding layer and a convolution layer. The embedding layer maps each input sequence of 1-dimensional byte values into the sequence of 8-dimensional real embeddings. This is done by handling every byte value as a categorical value and mapping it into a distinct 8-dimensional real point of the embedded space. This embedding represents bytes that have semantically similar behaviour as closer points in this space. The convolutional layer includes 128 filters to iterate over disjoint windows of 500 bytes each. The gated outputs of the convolutional layer integrate the global max pooling to select 128 features, achieving the largest activation. The selected features are subsequently used to feed a fully connected layer in charge of the final soft-max classification. In this study, we used the pre-trained MalConv model described in [10]. This model was trained using 1 Mb bytes of input Windows PE files. Specifically, the model was trained from the EMBER PE files by appending zeros to the end of PE files smaller than 1 Mb and removing the last bytes of PE files greater than 1 Mb. Notably, this model is also used in [5,11].

LGBM is a Light Gradient-Boosted Decision Tree Model trained with parameter set-up described in [45]. The pre-trained model was learned by processing 2381 features extracted through the static analysis of EMBER binary Windows PE files [10]. The static analysis was performed using the LIEF PE parsing library. The processed feature space included imports,

byte histograms, byte–entropy histograms, printable character histograms, header properties and section properties. Both the original study [10] and the subsequent study of [5] verified that LGBM outperforms sophisticated MalConv, highlighting the difficulty in outperforming the performance of domain knowledge via parsed features with featureless deep learning.

3.2. Attack Methods

As attack methods, we evaluated the performance of methods that generate realistic Windows PE malware by attacking the pre-trained MalConv model. We considered four gradient-based, white-box attacks and a gradient-free, black-box attack. All of these attacks were implemented in the secml-malware library. A short description of the attack methods considered in this study is reported below. It is mainly based on the material reported in [5].

3.2.1. White-Box Attack Methods

White-box attack methods basically work as described in [5]. First, they generate an initial raw byte-level perturbation of Windows PE malware by applying practical PE format- and functionality-preserving manipulations (e.g., injecting bytes in eligible zones). Then, they represent the manipulated binary PE file in the input space of the attacked model. Concerning MalConv, this corresponds to applying the byte embedding transformation $\Phi\colon \{0,\ldots,256\} \mapsto \mathcal{E}$ with $\mathcal{E} \subseteq \mathbb{R}^8$. Φ denotes the one-to-one mapping between each 1-dimensional binary input (corresponding to each byte in the initial 1 Mb sequence of the PE file) and its 8-dimensional real embedding in the MalConv input space. Notably, the input space of $\Phi(\cdot)$ includes the value 256 that is used to generate a padded embedding value whenever the embedding representation of a binary PE file with size smaller than 1 Mb must be padded to fit the fixed size of the embedding input dimension of the MalConv neural network. Subsequently, the embeddings of the manipulated input sample are iteratively perturbed to minimise the loss function. This is done by applying gradient-based update operations. Finally, the output sample produced in the embedding space is reconstructed through an inverse transformation, $\Phi^{-1}\colon \mathbb{R}^8 \mapsto \{0,\ldots,255\}$, from the 8-dimensional embedding space into the input 1-dimensional byte space. Φ^{-1} works by accounting for the embedding of one-to-one mapping, Φ. Let $\mathbf{z_i}$ be the i-th 8-dimensional vector of an output sample produced in the embedding space. Let $\mathbf{z'_i} = \arg\min_{\mathbf{z'_i} \in \mathcal{E}} ||\mathbf{z'_i} - \mathbf{z_i}||$ be the embedding vector value that is identified in the co-domain of the transformation Φ, as the closest to $\mathbf{z_i}$. The inverse reconstruction, $\Phi^{-1}(\mathbf{z_i}) = x'_i$, is recovered so that $\Phi(x'_i) = \mathbf{z'_i}$. The attack methods considered in this study, namely Extend, Full DOS, Shift and FGSM padding+slack, follow the attack schema described above, but they differ in the adopted manipulations.

Extend. This white-box attack method [5] creates a new area within the binary file by increasing the size of the DOS header. It uses the new area in the DOS header to add noise bytes. According to [5], this byte injection is done by keeping the functionality of the executable file. Specifically, the method operates in four steps. First, it determines how many bytes must be injected into the DOS header. Then, it identifies the PE header offset of the added area to record the injected bytes. Subsequently, it applies all the requested changes to make the new file compliant with the PE format constraints. For example, it increases the offset to the PE header, the size of the header field and the section entries. Finally, it applies the perturbation to the bytes that can be modified in the DOS header to create the adversarial payload.

Full DOS. This white-box attack method [5] applies noise to the bytes in the DOS header. It is based 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. In fact, the DOS header may be changed by keeping the functionality of the executable file except for the magic number "MZ" and the four-byte-long integer at offset "0 × 3c". In particular, the magic number identifies the file uniquely, while the four-byte-long integer at offset "0 × 3c" points to the starting point of the PE header in the binary code. Both cannot be changed in order to obtain an

executable file. Hence, this attack method 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.

Shift: This white-box attack method [5] applies the shift operation in order to the first section to recover room to add an adversarial byte chunk. The added binary chunk must have a size that is a multiple of the file alignment. This constraint must be satisfied to obtain an executable file that keeps its functionality. The method operates in three steps. First, it identifies the position of the first Section in the binary file. Then, it injects the noise bytes in the selected position. Finally, it updates the offset of each Section within the Section Table by considering that a new chunk of bytes has been injected in the first Section. In this way, the loader can still find the content of each section by neglecting the adversarial content injected before the first Section.

FGSM padding + slack. This white-box attack method [14] applies an iterative variant of the classical FGSM method [46] to the embedded representation of the binary file until it achieves evasion. It performs reconstruction at the end of the iterative perturbation process. In the reconstruction, each binary value that was appropriately perturbed within its embedded representation is transformed into a real byte within the raw byte input space through the application of the inverse transformation. To obtain an executable file that preserves its functionality, the noise is applied to a payload area that is injected into non-executable code sections. Specifically, this payload is injected through Slack Space and Padding manipulations. The Slack Space manipulation fills the space between sections. The compiler adds a chunk of zero bytes to each section to fill the gap. The Padding manipulation adds the padding bytes to the end of the code.

3.2.2. Black-Box Attack Methods

A black-box attack method produces adversarial malware, directly manipulating the binary Windows PE malware without considering its (embedding) feature representation and operating in a gradient-free setting. In particular, it observes the output of a query formulated to an attacked machine learning model by ignoring either what the model parameters are or how the model works to achieve its decisions. The authors of [8] surveyed several black-box adversarial attacks, illustrating a taxonomy of these methods. This taxonomy was formulated while accounting for the attack strategies, which may range from reinforcement learning, randomization and evolutionary methods to Generative Adversarial Network methods. In the following, we briefly describe GAMMA, which is an efficient attack method whose implementation is publicly available in the secml-malware library.

GAMMA. This black-box attack method [15] uses an evolutionary algorithm that injects an adversarial perturbation into the Windows PE malware file. This method solves an optimisation problem by resorting to a penalty term to minimise the evasion probability, as well as the size of the binary content that is added to the PE file. The injected content is extracted from goodware binary files instead of being produced randomly. To find the optimised benign content, GAMMA selects benign content iteratively and optimises the selection and size of goodware-originated content using the selection, crossover and mutation functions. As in [15], we used the section evasion formulation of GAMMA, which resorts to the section injection operation to extract sections from goodware files and inject them as a new section into the produced adversarial malware file. In addition, a new section entry is added to the Section Table of the adversarial file. Notably, the authors of [15] showed that, although this operation changes both the byte distribution and the structure of the binary file, it preserves the code functionality by design.

3.2.3. Remarks on Adversarial PE Malware Execution and Structure

We note that, according to the analysis illustrated in [5], the attack methods implemented in the secml-malware library resort to manipulations of the PE file format that can

change the structure of Windows PE malware files but preserve their semantics. In fact, they apply manipulations to areas of the file that do not impact functionality by avoiding the need for computationally demanding validation steps to be applied to discard files that are not correctly executed in sandbox environments. However, in this study, we decided to randomly select 10 adversarial Windows PE malware generated from each attack method, and we verified that they were all correctly executed in the VirusTotal sandbox.

3.3. SHAP

SHAP (SHapley Additive exPlanations) [17] is a post hoc, XAI technique that has been recently used in several cybersecurity domains [47,48]. In this article, it was used to explain how input values mainly condition decisions of machine learning models, as well as to understand how the the presence of adversarial attacks may have an effect on decisions of an effective machine learning model produced for Windows PE malware detection. In particular, SHAP is based on a theoretical game that is performed to measure the effect of each dimension of the input feature space 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. Let $\phi\colon \mathbb{R}^d \mapsto Y$ be the decision model to be explained so that $\mathbb{R}^d$ denotes the input space (i.e., 1 million raw bytes in MalConv, engineered features extracted through the LIEF static analyser in LGBM) and Y the set of classes (i.e., "goodware" and "malware"). Let $\mathbf{x} \in \mathbb{R}^d$ denote the d-dimensional representation of a Windows PE file to be classified with ϕ. SHAP measures the effect of each input dimension value $x \in \mathbf{x}$ on the decision $\phi(\mathbf{x})$ as the average marginal contribution of a feature value on the various alternative decisions.

Let $\phi(\mathbf{x})[y]$ denote the confidence score according to ϕ seeing $\mathbf{x}$ assigned to class $y \in Y$. ϕ assigns $\mathbf{x}$ in the class for which the highest confidence score is computed. For each input feature $X \in \mathbb{R}^d$ to explain, for each input feature sub-space $\mathbf{X} \subseteq \mathbb{R}^d/\{X\}$, let $\phi_{\mathbf{X}}\colon \mathbf{X} \mapsto Y$ and $\phi_{\mathbf{X}\cup\{X\}}\colon \mathbf{X}\cup\{X\} \mapsto Y$ be surrogate models with the input feature spaces $\mathbf{X}$ and $\mathbf{X}\cup\{X\}$, respectively. SHAP computes the difference between the confidence scores determined via $\phi_{\mathbf{X}}$ and $\phi_{\mathbf{X}\cup\{X\}}$, respectively:

$$\varphi_{\mathbf{X},X,y}(\mathbf{x}) = \phi_{\mathbf{X}\cup\{X\}}(\pi_{\mathbf{X}\cup\{X\}}(\mathbf{x}))[y] - \phi_{\mathbf{X}}(\pi_{\mathbf{X}}(\mathbf{x}))[y], \tag{1}$$

where $\pi_{\mathbf{X}}\colon \mathbb{R}^d \mapsto \mathbf{X}$ and $\pi_{\mathbf{X}\cup\{X\}}\colon \mathbb{R}^d \mapsto \mathbf{X}\cup\{X\}$ represent the functions to select the input values enclosed in $\mathbf{X}$ and $\mathbf{X}\cup\{X\}$, respectively. The Shapley values are, finally, measured as the weighted average of the various alternative differences as follows:

$$\Psi_{X,y}(\mathbf{x}) = \sum_{\mathbf{X}\subseteq\mathbb{R}^d/\{X\}} \frac{|\mathbf{X}|!\,(d-|\mathbf{X}|-1)!}{d!}(\varphi_{\mathbf{X},X,y}(\mathbf{x})), \tag{2}$$

where $|\bullet|$ represents the vector size. The higher the value of $\phi_{X,y}(\mathbf{x})$, the most important effect of X in the decision of the model of predicting $\mathbf{x}$ in the class y. In this study, we used SHAP to explain decisions produced from the considered machine learning models for the class "malware".

3.4. Adversarial Training

The adversarial training strategy was originally illustrated in [4] as a mechanism to make deep neural models robust to adversarial attacks by training deep neural models from clean samples augmented with adversarial samples. Initially defined for computer vision problems, adversarial training has been recently used with general-purpose attack methods in cybersecurity [21], mainly to mitigate the overfitting risk and gain accuracy on clear data at testing time. As general-purpose attack methods may not generate realistic Windows PE files, in this study, adversarial training was performed with adversarial samples generated using attack methods specifically defined in the literature to generate realistic adversarial Windows PE malware. The performance of the adversarial training strategy was evaluated

by resorting to a 3-fold cross-validation strategy with the LGBM method. For each attack method, realistic adversarial Windows PE malware was added to the training set. Both the original training set and the augmented training set were used to train the two models, $LGBM^{O}$ and $LGBM^{\mathsf{AT}_{\mathsf{attack}}}$, respectively. attack denotes the Windows PE attack method (Extend, Full DOS, Shift, FGSM padding+slack and GAMMA) used to generate the realistic adversarial Windows PE malware. Finally, both models were evaluated using real testing of Windows PE files (O), as well as realistic adversarial Windows PE malware generated from the testing malware (O + A).

4. Data and Evaluation Metrics

This evaluation study was conducted by considering a collection of 27,035 Windows PE files: 13,494 goodware and 13,541 malware. Notably, we made the dataset we prepared to conduct this evaluation study publicly available for future research (see the Data Availability Statement of this article). Goodware was selected from the public PEMML repository, which is the only online repository that we have found to publicly provide binaries of goodware for free. Specifically, we selected 980 goodware files recorded in 2017 and 12514 goodware files recorded in 2018. Malware files were selected from VirusShare. In particular, we randomly selected 6459 malware files recorded in 2021, 83 in 2022 and 6999 in 2023. The following three queries were performed on VirusShare to retrieve the Windows PE malware files:

1. "filetype: "PE32 executable" extension:exe after 1 January 2021",
2. "filetype: "PE32 executable" extension:exe after 1 January 2022",
3. "filetype: "PE32 executable" extension:exe after 1 January 2023".

The queries were performed between March and June 2023. Malware files were downloaded in that period one by one. Notably, the dataset used is larger and contains more recent Windows PE files than the dataset used in [5], which considered 104 Windows PE malware files retrieved from DasMalwerk in late 2018. A short description of the PEMML and DasMalwerk repositories, as well as the VirusShare service, is reported in Section 2.2.

We selected the first 1 Mb of each binary Windows PE file to be processed with MalConv. According to the description reported in [6], for each Windows PE file whose size was smaller than 1 Mb, the embedded input representation of this file was padded with the embedded value $\Phi(256)$. We recall that, as described in Section 3.2.1, MalConv applies the byte-embedding transformation $\Phi(\cdot)$ to transform each integer value recorded in a byte into 8-dimensional real embedding that feeds the embedding input space of a fixed size of MalConv. Therefore, whenever the Windows PE file size is smaller than 1 Mb, the sequence of the 8-dimensional real embeddings that were generated from the sequence of bytes actually recorded in the binary file is padded with the 8-dimensional real embedding of the value 256 until the expected size of the embedding input space of MalConv is reached. In addition, we used the static analyser of LIEF (version 0.9) (https://github.com/lief-project/LIEF, accessed on 30 April 2024) to perform the static analysis of each binary Windows PE file and extract the vector of 2381 engineered features to be processed with LGBM.

We measured the performance of both MalConv and LGBM models by computing standard metrics commonly used to evaluate the predictive ability of binary classification models. Specifically, let us consider tm—the amount of Windows PE malware that is correctly predicted as malware; fm—the amount of Windows PE goodware that is wrongly predicted as malware; tg—the amount of Windows PE goodware that is correctly predicted as goodware; and fg—the amount of Windows PE malware that is wrongly predicted as goodware. We computed the following metrics:

- *Overall accuracy* (oa) measures the proportion of correctly classified Windows PE files, regardless of the class, out of all the predicted files, i.e., $\mathsf{oa} = \frac{\mathsf{tm}+\mathsf{tg}}{\mathsf{tm}+\mathsf{tg}+\mathsf{fm}+\mathsf{fg}}$. This metric estimates the overall ability of a decision model to correctly classify a sample in its proper class, regardless of the class value.

- *Precision* (prec) measures how many Windows PE malware files are correctly classified as malware, given all predictions of the malware class, i.e., $\mathsf{prec} = \frac{\mathsf{tm}}{\mathsf{tm}+\mathsf{fm}}$. This metric estimates how often the decision model is correct when predicting the target class "malware".
- *Recall* (recall) measures how many Windows PE malware files are correctly classified as malware, given all occurrences of class malware, i.e., $\mathsf{recall} = \frac{\mathsf{tm}}{\mathsf{tm}+\mathsf{fg}}$. This metric estimates whether the decision model can find all samples of the target class "malware".
- *Fscore* (F) measures the harmonic mean of precision and recall, i.e., $\mathsf{F} = 2\frac{\mathsf{prec}\cdot\mathsf{recall}}{\mathsf{prec}+\mathsf{recall}}$. As precision and recall are equally important, the Fscore is measured to estimate the trade-off between precision and recall. In particular, the higher the Fscore, the better the balance between precision and recall achieved by the evaluated approach.
- *The false negative rate* (fnr) measures the probability that Windows PE malware is wrongly classified as goodware, i.e., $\mathsf{fnr} = \frac{\mathsf{fg}}{\mathsf{tm}+\mathsf{fg}}$. The lower the false negative rate, the lower the number of malware files that are undetected.
- *The false positive rate* (fpr) measures the probability that Windows PE goodware is wrongly classified as malware, i.e., $\mathsf{fpr} = \frac{\mathsf{fm}}{\mathsf{tg}+\mathsf{fm}}$. The lower the false positive rate, the lower the number of goodware files that are wrongly detected as malware.

The higher the values of oa, prec, recall and F, the better the decision model. The lower the values of fnr and fpr, the better the decision model.

As an additional metric to measure the evasion ability of an attack method against a malware detection model, we consider evasion, which is the number of Windows PE malware files that are correctly classified with the model but whose adversarial counterparts, generated via the attack method, are wrongly classified as goodware according to the model. Formally, let tm be the number of true malware files recovered on the set of Windows PE malware files and tm^{A} be the number of true malware files recovered on the set of original Windows PE malware files where original malware files were replaced with adversaries whenever adversaries exist. Hence, $\mathsf{evasion} = \mathsf{tm} - \mathsf{tm}^{\mathsf{A}}$. The higher the evasion value, the higher the number of adversarial Windows PE malware files that evaded the decision model, and consequently, the lower the integrity of the model versus adversarial attacks.

5. Results

The main goals of this evaluation study were as follows:

- To evaluate the accuracy of the two pre-trained models, MalConv and LGBM, on the Windows PE dataset prepared for this study (Section 5.1).
- To evaluate the integrity of the two pre-trained models, MalConv and LGBM, with realistic adversarial Windows PE malware produced via the attack methods considered in this study (Section 5.2).
- To analyse the distance between original Windows PE malware and its adversarial counterparts (see Section 5.3).
- To explain why the LGBM model that was learned using engineered features may work differently when it produces decisions related to the malicious behaviour of the adversarial Windows PE malware files generated via the study attack methods (Section 5.4).
- To investigate the performance of adversarial training done with adversarial malware samples produced through realistic Windows PE attack methods (Section 5.5).

Experiments with adversarial training were performed using a three-fold cross-validation of the Windows PE dataset prepared for this study.

5.1. Accuracy Analysis of Pre-Trained MalConv and LGBM

Table 2 reports oa, prec, recall, F, fnr and fpr computed by measuring the accuracy performance of the pre-trained MalConv and LGBM models on the Windows PE dataset prepared to conduct this evaluation study.

Table 2. Accuracy performance analysis of the pre-trained MalConv and LGBM models. The accuracy metrics (oa—overall accuracy, prec—precision, recall—recall, F—Fscore, fnr— the false negative rate and fpr—the false positive rate) were measured on the Windows PE dataset prepared for this evaluation study. The best results are underlined.

Model	oa	prec	recall	F	fnr	fpr
MalConv	0.8034	0.9766	0.6224	0.7603	0.3776	0.0150
LGBM	0.9294	0.9857	0.8717	0.9252	0.1283	0.0127

The results confirm that the conclusions drawn in [10] are still valid, even whenconsidering the newest Windows PE malware files we collected for this evaluation study. In particular, despite the increased model size of MalConv (an input space composed of 1 million raw byte-based features against 2381 engineered features extracted through the static analysis of PE files), the feature engineering step still allowed LGBM to account for PE file characteristics that contribute to better disentangling malware from goodware.

5.2. Integrity Analysis of Pre-Trained MalConv and LGBM

Table 3 reports the evasion metric measured for the pre-trained MalConv and LGBM models. The evasion metric was computed on the Windows PE malware files of the study dataset for which we were able to generate realistic adversarial Windows PE malware by attacking MalConv with the attack methods: Extend, Full DOS, Shift, FGSM padding+slack and GAMMA. Notably, the evasion metric measured on the LGBM model allowed us to verify the transferability of the evasion ability of the adversarial Windows PE malware files produced by fooling the pre-trained MalConv model versus the pre-trained LGBM model. In fact, the higher the evasion measured on the LGBM model, the higher the number of adversarial Windows PE malware files that were generated to fool the MalConv model but also evaded the LGBM model. In addition, we note that a negative value of evasion means that the Windows PE malware files generated with the considered attack method were all correctly detected in the "malware" class using the LGBM model, even when the original malware counterparts from which the adversarial malware files were generated were wrongly classified as goodware according to the same model.

Table 3. The integrity performance (measured through the evasion metric) of the pre-trained MalConv and LGBM models and computed with respect to the following attack methods: Extend, Full DOS, Shift, FGSM padding + slack and GAMMA. The attack methods were used with the Windows PE malware of the study dataset to attack the pre-trained MalConv model.

Attack Method	MalConv	LGBM
Extend	7951	306
Full DOS	6115	−10
Shift	3423	41
FGSM	4384	−157
GAMMA	6857	1266

So, based upon the considerations reported above, the results of the evasion metric achieved using the pre-trained MalConv model show that Extend is the attack method that was able to generate the higher number of Windows PE malware files that evaded the MalConv model (i.e., they were wrongly classified as goodware), while their original counterparts (i.e., the malware files used for adversarial file generation) were correctly classified as malware according to the MalConv model. GAMMA and Full DOS were the runner-up attack methods for the pre-trained MalConv model. On the other hand, the analysis of the transferability of the realistic adversarial malware files, which fooled both the MalConv and LGBM models, shows that GAMMA achieved the highest transferability using the produced adversarial Windows PE malware files with Extend as the runner-up. In fact, GAMMA measured the highest value of evasion using the LGBM model, as it generated

the highest number of adversarial samples fooling the LGBM model (in addition to the MalConv model). These results support conclusions already drawn in [5] by processing a smaller dataset of older Windows PE malware files. Notably, Full DOS, which generated 6115 adversarial malware files to fool the MalConv model, achieved a negative evasion value with LGBM. This means that 10 malware samples of the original dataset were wrongly classified according to the LGBM model as goodware, while their adversarial counterparts generated to fool the MalConv model with the Full DOS method were correctly detected as malware according to the LGBM model. A similar behaviour was observed for the Shift method.

To complete this analysis, Figure 2a,b shows the F and fnr metrics measured for the pre-trained MalConv and LGBM models on the modified version of the evaluation dataset prepared for this study. Specifically, to this aim, for each attack method, we produced a modified version of the study dataset by replacing each original PE malware for which the attack method was able to generate realistic adversarial malware that fooled MalConv with the produced adversarial malware. We performed this analysis while considering F and fnr, as both metrics may have been affected by the evading ability of adversarial samples. In this analysis, the lower the value of F and the higher the value of fnr, the higher the evasion ability of the attack method with respect to the decision model, and consequently, the lower the integrity of the study decision model. Notably, both metrics confirmed that GAMMA is the attack method for this evaluation study; it was able to evade LGBM with a higher number of realistic adversarial samples generated by fooling MalConv.

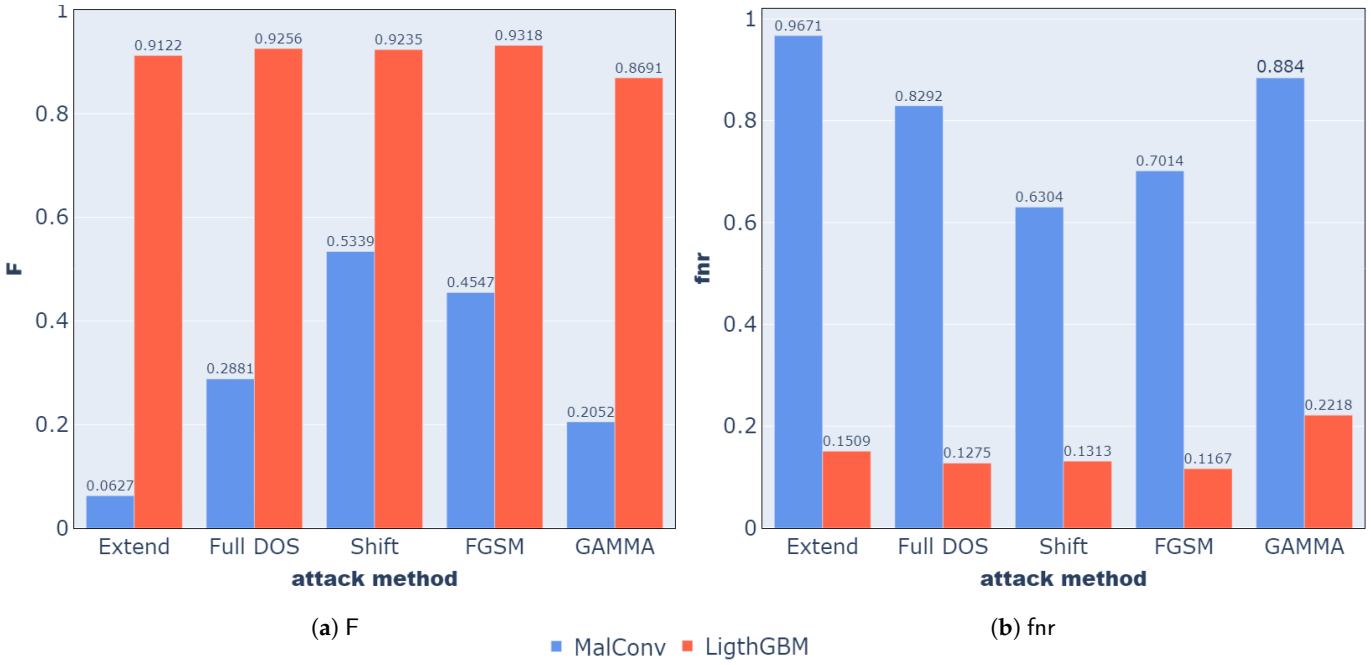


Figure 2. Evasion analysis of the pre-trained MalConv and LGBM models: F score (**a**) and fnr (**b**), measured on adversarial PE malware produced from the attack methods: Extend, Full DOS, Shift, FGSM padding + slack and GAMMA.

5.3. Distance Analysis of Adversarial Windows PE Malware

In this section, we analyse the distribution of the distance values that were computed between the original PE malware and the adversarial PE malware generated from each attack method considered in this evaluation study. This distance was measured through the Euclidean distance computed in both the raw byte-based input space of MalConv (where attacks were produced) and the engineered feature-based input space of LGBM (where the transferability of the generated attacks was examined). The Euclidean distance values were computed after scaling the input dimensions of all files between 0 and 1. Each distance value

was divided by the number of dimensions in the input space in which it was calculated. This distance analysis was conducted to verify the existence of a possible relationship between the overall amount of changes introduced in the binary code of the adversarial PE files with each attack method and the corresponding amount of changes observed in the engineered features extracted for the same samples in the input space of LGBM. In addition, we intended to explore whether this relationship may be somehow related to the transferability of the attack method. The existence of such a relationship can be considered as a mechanism to foresee and explain the transferability of an attack method. Figure 3 shows the box plots of the Euclidean distance values computed between the original PE malware and the adversarial PE counterpart malware generated from each attack method considered in this evaluation study.

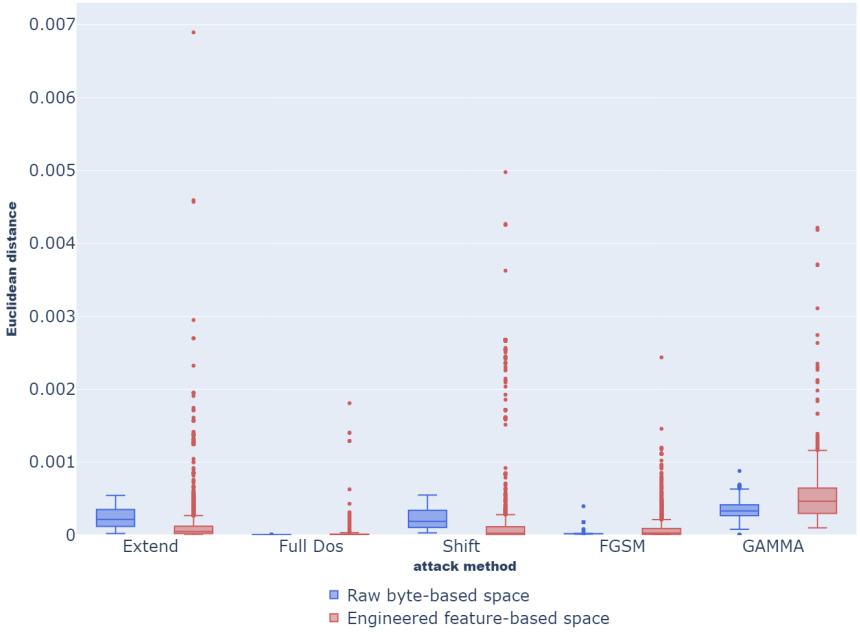


Figure 3. Box plots of Euclidean distance values computed in both the raw byte-based input space of MalConv (blue boxes) and the engineered feature-based input space of LGBM (red boxes) between original PE malware and its realistic adversarial malware counterparts generated with Extend, Full DOS, Shift, FGSM padding + slack and GAMMA.

The results collected in the raw byte-based input space show that both Full DOS and FGSM padding + slack produced the adversarial malware files of this evaluation study closest to the binary files of the original counterpart malware. We recall that no adversarial malware that was generated from both of these methods by attacking MalConv was able to also evade LGBM. Even some adversarial PE files generated from both these attack methods were correctly classified as malicious. In contrast, the original PE files were wrongly classified as goodware according to LGBM (see the negative evasion results reported in Table 3). Hence, this analysis suggests that a low raw byte distance between the adversarial malware and its original counterpart negatively affects the transferability of the adversarial malware (i.e., the possibility of evading a model different from the one for which it was generated). This conclusion is also supported by the fact that the highest raw byte distances are commonly observed in GAMMA, which is the attack method that produced the higher amount of adversarial malware able to evade both MalConv and LGBM. On the other hand, distances measured in the engineered feature-based input space show that GAMMA is the attack method that produced adversarial malware files that are generally furthest from the original counterpart malware also in the engineered feature-based space.

To complete this study, we examined in-depth distances measured on the adversarial samples produced using GAMMA. Figure 4 shows the box plots of the Euclidean distances

computed for the adversarial malware produced with GAMMA. To perform this analysis, we grouped GAMMA adversarial malware into two groups with respect to their ability to evade LGBM (in addition to MalConv). We note that distances measured in the engineered feature-based space show that the adversarial malware that evaded LGBM are generally furthest from the original counterpart malware files than the adversarial malware that non-evaded LGBM. This supports the conclusion that an attack method that modifies a PE file in the raw byte space should be able to induce a remarkable change in the engineered features subsequently generated through the static analysis of the file. This allows the produced PE files to evade an accurate decision model like LGBM learned in a sophisticated feature-engineered space.

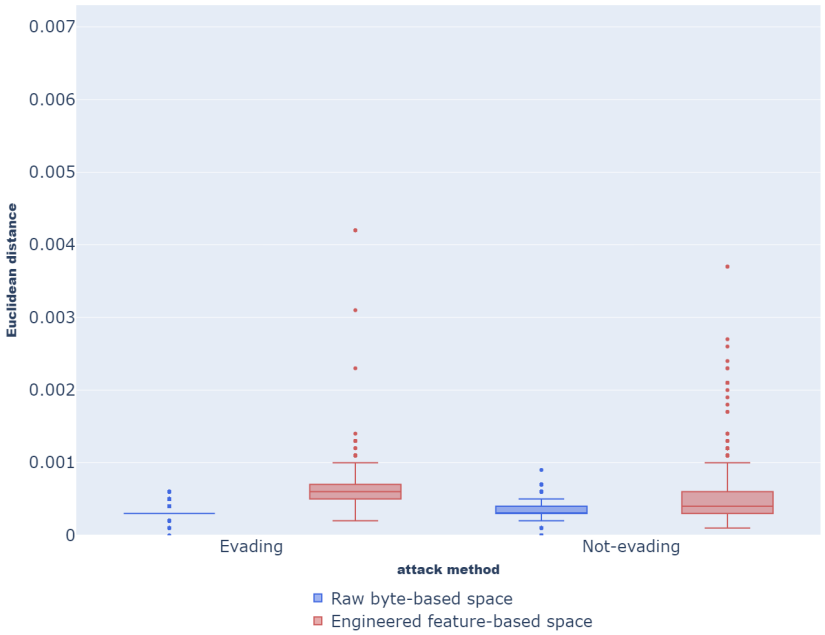


Figure 4. Box plots of Euclidean distance values (axis *Y*) computed in both the raw byte-based input space of MalConv (blue boxes) and the engineered feature-based input space of LGBM (red boxes) between original PE malware and their realistic adversarial malware counterparts generated with GAMMA and grouped with respect to their ability to evade or not-evade the pre-trained LGBM model (axis *X*).

5.4. XAI-Based Analysis of the Effect of Adversarial Windows PE Malware on Engineered Features

In this section, we consider the pre-trained LGBM model as the decision model and GAMMA as the attack method. LGBM was selected since it was identified as the most accurate decision model in this study (see Table 2). GAMMA was selected since it was identified as the attack method that achieved the highest evasion ability in this study by generating a higher number of adversarial malware files that evaded the LGBM model (see Table 3). We plot the top 20 engineered features of the LGBM input space sorted by their global importance as it was computed using SHAP. In particular, for each engineered feature of the input space of LGBM, the feature's global importance was determined as the average Shapley value computed for the feature with respect to the "malware" class. Due to the focus of this study on the adversarial behaviour, single Shapley values were computed for each Windows PE malware file of the study dataset for which an adversarial Windows PE malware file was actually generated via GAMMA. The results of the analysis of the Shapley values for the remaining attack methods (i.e., Extend, Full DOS, Shift and FGSM padding + slack) are illustrated in Appendix A.

Figure 5 shows the Shapley values plotted with respect to the feature values measured in the explained samples for the top-ranked dimensions of the input feature space of the pre-trained LGBM model. In particular, the left-side chart (Figure 5a) shows the Shapley

values that were computed to explain the decisions produced for the original malware files for which GAMMA was able to generate adversarial samples that evaded LGBM, while the right-side chart (Figure 5b) shows the Shapley values that were computed to explain the decisions produced for the adversarial malware counterparts of the selected samples produced with GAMMA. In these two charts, the colour scale expresses how the values observed for each feature are distributed in the explained samples; "new entry" means that a feature appeared in the top 20 ranking of the input features that mainly affected the decisions yielded via the LGBM model for the study's adversarial Windows PE malware files, but the same feature did not appear in the top 20 ranking of the features that affected the decisions yielded using the LGBM model for the original Windows PE malware files (used to generate the study's adversarial samples). Additionally, "up" (or "down") means that a feature gained (or lost) positions in the top 20 ranking of the input features that mainly affected the decisions yielded via the LGBM model for the study's adversarial malware files compared to the top 20 ranking of the features that affected the decisions yielded for the study's original malware files. Finally, "changed feature value" means that the distribution of the values observed for the feature changed from the plot produced for the study's original samples to the plot produced for the study's adversarial samples. The plots show that numerous changes appear in the top 20 Shapley-based feature rankings. Our feeling is that the high number of changes seen in the explanation of decisions yielded for the adversarial Windows PE malware files produced from GAMMA highlights which input dimensions of the LGBM model were mainly fooled by these adversarial malware files. Notably, this analysis shows that transferable attacks produced with GAMMA introduce some relevant changes in decisions yielded via LGBM with transferred adversarial samples.

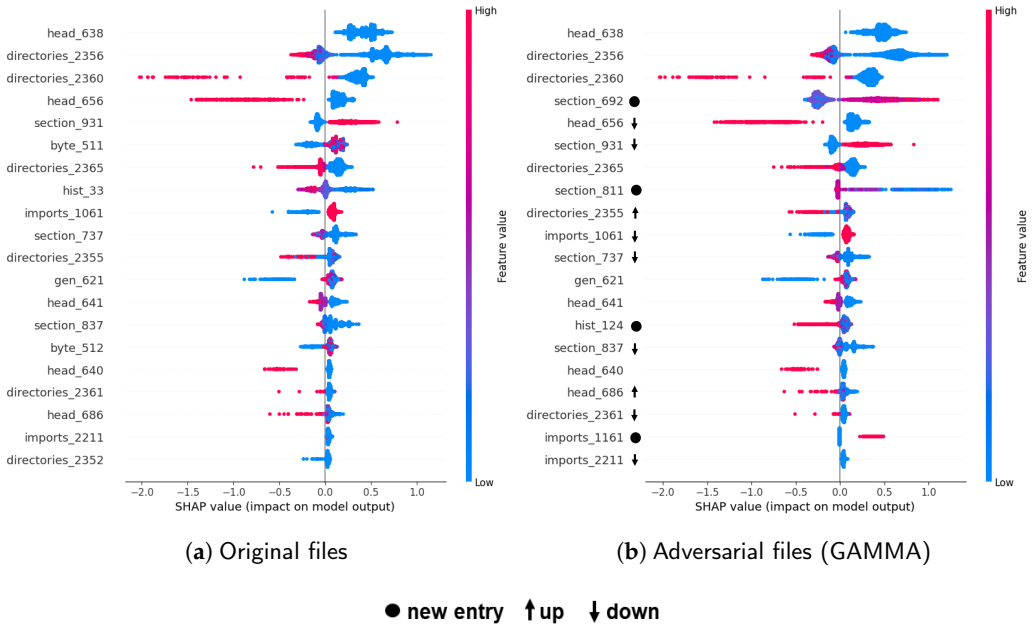


Figure 5. Shapley values measured for the top 20 input features (axis *Y*) of the LGBM input space and plotted with respect to the values measured for the features in the explained samples (axis *X*) for the original 6857 PE malware files (**a**) and the adversarial malware counterparts produced from GAMMA (**b**). Changes in the Shapley value-based feature ranking are marked in (**b**).

5.5. Performance Analysis of Adversarial Training with LGBM and Realistic Windows PE Attack Methods

In this section, we analyse the accuracy performance of the adversarial training strategy by exploring the effect of this strategy on the accuracy performance of the machine learning model trained with LGBM when this strategy is evaluated using a collection of the original Windows PE files (denoted as O), as well as when it is evaluated using the collection of the original Windows PE files extended with the adversarial Windows PE malware (denoted

as O + A). The study was conducted considering the adversarial Windows PE malware files generated with Extend, Full DOS, Shift, FGSM padding+slack and GAMMA. In this study, we used a 3-fold cross-validation (CV) of the study's dataset. For each fold, we learned both $LGBM^{O}$ from the Windows PE files falling in the two hold-out folds and $LGBM^{AT}$ from the Windows PE files falling in the two hold-out folds augmented with the realistic adversarial malware produced with a selected attack method. Both models were used to classify the Windows PE files of the selected fold. The accuracy metrics were, finally, computed using the decisions produced with the 3-fold CV methodology for all the Windows PE files (and eventually their adversarial counterparts) of the study's dataset.

We started the analysis of the results by examining the performance achieved in configuration O to evaluate the accuracy of predictions produced for the collection of the original Windows PE files. Table 4 reports the values collected for both the detailed accuracy metrics (tg—the amount of true goodware, fm—the amount of false malware, fg—the amount of false goodware and tm—the amount of true malware) and the summary accuracy metrics (oa—overall accuracy, F—Fscore, fnr— the false negative rate and fpr—false positive rate). These metrics were measured using the collection of predictions produced from $LGBM^{O}$ and $LGBM^{AT}$ in the evaluation configuration O. These results show that the use of the adversarial training strategy worsened the performance of LGBM in terms of tg, fm, oa, F and fpr regardless of the attack method used to perform the adversarial training strategy. On the other hand, the use of the adversarial training strategy allowed $LGBM^{AT}$ to perform better than (or equally to) $LGBM^{O}$ in terms of fg, tm and fnr when adversarial samples considered in the adversarial training stage of $LGBM^{AT}$ were produced with Extend, Full DOS and GAMMA. Specifically, when considering these attack methods, the injection of realistic adversarial samples in the training stage allowed us to learn about an LGBM model that increases the number of malware decisions while also considering the original PE files only for evaluation. Notably, this had the consequence of increasing the amount of true malware detected. However, this also had the consequence of reducing the amount of true goodware detected by increasing the amount of false malware. Although we are aware that handling false malware is an inefficient use of time and resources, which may prevent a cybersecurity team from handling actual malware, we also note that false goodware may raise a serious cyber risk affecting the capacity to promptly mitigate cyber threats' consequences. From this point of view, any reduction in fnr may be considered a desirable behaviour, as long as it does not come at the expense of an excessive increase in fpr.

We continued the analysis by examining the performance achieved in the configuration A + T to evaluate the accuracy of the predictions produced for the collection of the original Windows PE files augmented with the realistic Windows PE malware produced with one of the attack methods considered in the evaluation stage. Table 5 reports the values collected for both the detailed accuracy metrics and the summary accuracy metrics measured using the collection of predictions produced from $LGBM^{O}$ and $LGBM^{AT}$ in the evaluation configuration O + A. In this evaluation setting, the LGBM model learned with the adversarial training strategy gained accuracy with respect to all the summary accuracy metrics of this study when the realistic adversarial Windows PE malware considered in both the training and evaluation stages was produced with GAMMA. In fact, in examining the detailed accuracy metrics, we noted that the adversarial training strategy performed with GAMMA decreased the amount of true goodware (tg) by increasing the amount of false malware (fm) detected. However, it also increased the amount of true malware (tm) by decreasing the amount of false goodware (fg) detected. As the gain in the ability to correctly detect malware was greater than the reduction in the ability to correctly detect goodware, the use of the adversarial training strategy may be considered beneficial with GAMMA. In addition, the use of the adversarial training strategy gained accuracy with respect to fg, tm, oa, F and fnr when the realistic adversarial PE malware was produced with Extend. We recall that GAMMA was identified in this evaluation study as the most effective attack method, with Extend as the runner-up. So, this analysis highlights that adversarial training

with GAMMA-produced samples can be a strategy to strengthen the LGBM model against this attack type. Notice that this analysis was performed on LGBM, which was, in the end, the most effective decision model for the Windows PE malware detection task in this study.

Table 4. Detailed accuracy metrics (tg—amount of true goodware, fm—amount of false malware, fg—amount of false goodware and tm—amount of true malware), and summary accuracy metrics (oa—overall accuracy, F—Fscore, fnr—the false negative rate and fpr—the false positive rate) of $LGBM^{O}$ and $LGBM^{AT}$ measured in the evaluation configuration O [1]. The evaluation was conducted through the 3-fold CV. Adversarial Windows PE malware files used to perform the adversarial training strategy in the training stages of $LGBM^{AT}$ were produced with Extend, Full DOS, Shift, FGSM padding + slack and GAMMA. Metrics for which $LGBM^{AT}$ outperformed (or performed equally to) $LGBM^{O}$ are underlined.

Model	Test Set (O)							
	tg	fm	fg	tm	oa	F	fnr	fpr
$LGBM^{O}$	13,432	62	60	13,481	0.9955	0.9955	0.0044	0.0046
$LGBM^{AT}$ (Extend)	13,423	71	54	13,487	0.9954	0.9954	0.0040	0.0053
$LGBM^{AT}$ (Full DOS)	13,422	72	55	13,486	0.9953	0.9953	0.0041	0.0053
$LGBM^{AT}$ (Shift)	13,423	71	60	13,481	0.9952	0.9952	0.0044	0.0053
$LGBM^{AT}$ (FGSM)	13,430	64	62	13,479	0.9953	0.9953	0.0046	0.0047
$LGBM^{AT}$ (GAMMA)	13,425	69	61	13,480	0.9952	0.9952	0.0045	0.0051

[1] O denotes the collection of decisions produced for the study's collection of original Windows PE files.

Table 5. Detailed accuracy metrics (tg—amount of true goodware, fm—amount of false malware, fg—amount of false goodware and tm—amount of true malware), and summary accuracy metrics (oa—overall accuracy, F—Fscore, fnr—the false negative rate and fpr—the false positive rate) of $LGBM^{O}$ and $LGBM^{AT}$ measured in the evaluation configuration O + A [1]. The evaluation was conducted through the 3-fold CV. Adversarial Windows PE malware files were produced with Extend, Full DOS, Shift, FGSM padding + slack and GAMMA. Metrics for which $LGBM^{AT}$ outperformed (or performed equally to) $LGBM^{O}$ are underlined.

Model	Test Set (O + A)							
	tg	fm	fg	tm	oa	F	fnr	fpr
$LGBM^{O}$	13,432	62	95	21,398	0.9955	0.9963	0.0044	0.0046
$LGBM^{AT}$ (Extend)	13,423	71	69	21,424	0.9960	0.9967	0.0032	0.0053
$LGBM^{O}$	13,432	62	84	19,571	0.9956	0.9963	0.0043	0.0046
$LGBM^{AT}$ (Full DOS)	13,422	72	77	19,578	0.9955	0.9962	0.0039	0.0053
$LGBM^{O}$	13,432	62	77	16,887	0.9954	0.9959	0.0045	0.0046
$LGBM^{AT}$ (Shift)	13,423	71	79	16,885	0.9951	0.9956	0.0047	0.0053
$LGBM^{O}$	13,432	62	81	17,844	0.9954	0.9960	0.0045	0.0046
$LGBM^{AT}$ (FGSM)	13,430	64	83	17,842	0.9953	0.9959	0.0046	0.0047
$LGBM^{O}$	13,432	62	135	20,265	0.9942	0.9952	0.0066	0.0046
$LGBM^{AT}$ (GAMMA)	13,425	69	63	20,337	0.9961	0.9968	0.0031	0.0051

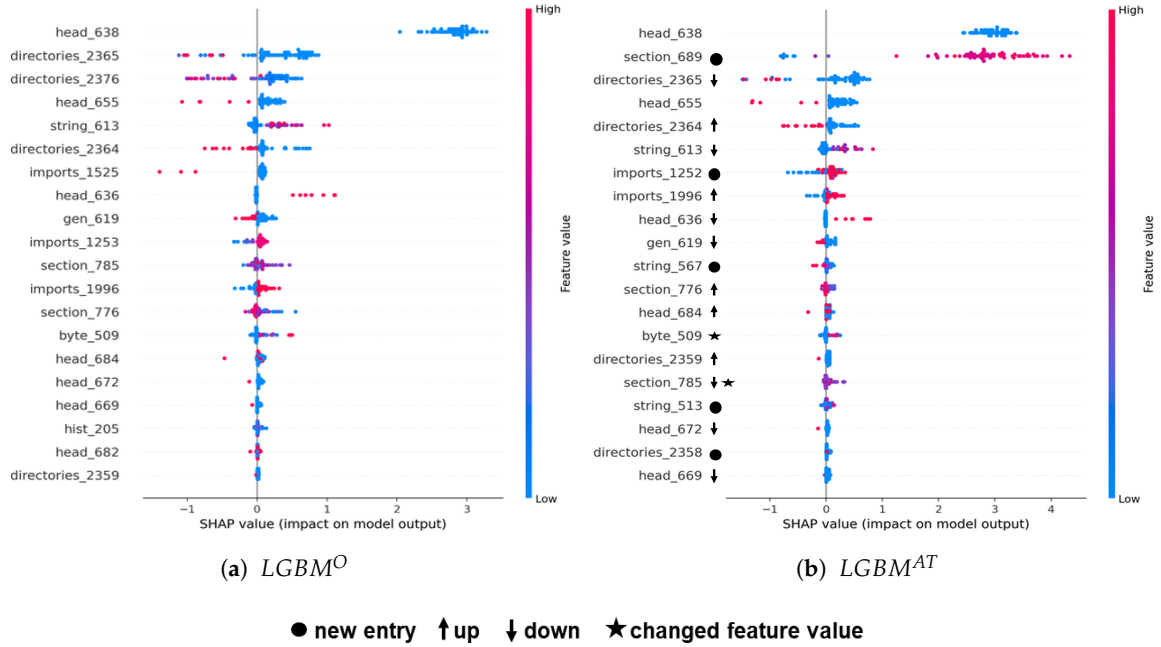
[1] O denotes the collection of decisions produced for the study's collection of original Windows PE files, while A denotes the collection of decisions produced for the collection of realistic adversarial malware produced from the study's original PE malware recorded in O.

These considerations are also supported by the in-depth analysis of the accuracy performance of both $LGBM^O$ and $LGBM^{AT}$ measured with the subset of the adversarial Windows PE malware files that were used in the evaluation stage of the $A + O$ setting of each attack method. Table 6 reports the number of adversarial Windows PE malware files correctly classified in the "malware" class (true malware—tm) and the remaining number of adversarial Windows PE malware files wrongly classified in the "goodware" class (false goodware—fg) from both decision models. These results show that the adversarial training strategy is ineffective with attacks generated via Full DOS, Shift and FGSM padding + slack, while it allows us to reduce the number of mis-classified adversarial Windows PE malware files (fg) when Extend and GAMMA are used as attack methods. In any case, the highest performance improvement was achieved using the adversarial training strategy with GAMMA. In fact, in this case, $LGBM^{AT}$ achieved the greatest reduction in the number of adversarial Windows PE malware files that were wrongly classified as goodware compared to $LGBM^O$ by passing from 75 to 2 false goodware decisions (fg). This result better supports our findings about the effectiveness of the adversarial training strategy with adversarial Windows PE malware files produced with GAMMA.

Table 6. The number of "true malware" decisions (tm) and the number of "false goodware" decisions (fg) yielded via $LGBM^O$ and $LGBM^{AT}$ for the adversarial Windows PE malware files produced with the Extend, Full DOS, Shift, FGSM padding+slack and GAMMA attack methods and used in the evaluation stage of the $O + A$ settubg considered in Table 5.

Attack Method	$LGBM^O$		$LGBM^{AT}$	
	tm	fg	tm	fg
Extend	7916	35	7936	15
Full DOS	6091	24	6093	22
Shift	3406	17	3404	19
FGSM	4363	21	4363	21
GAMMA	6782	75	6855	2

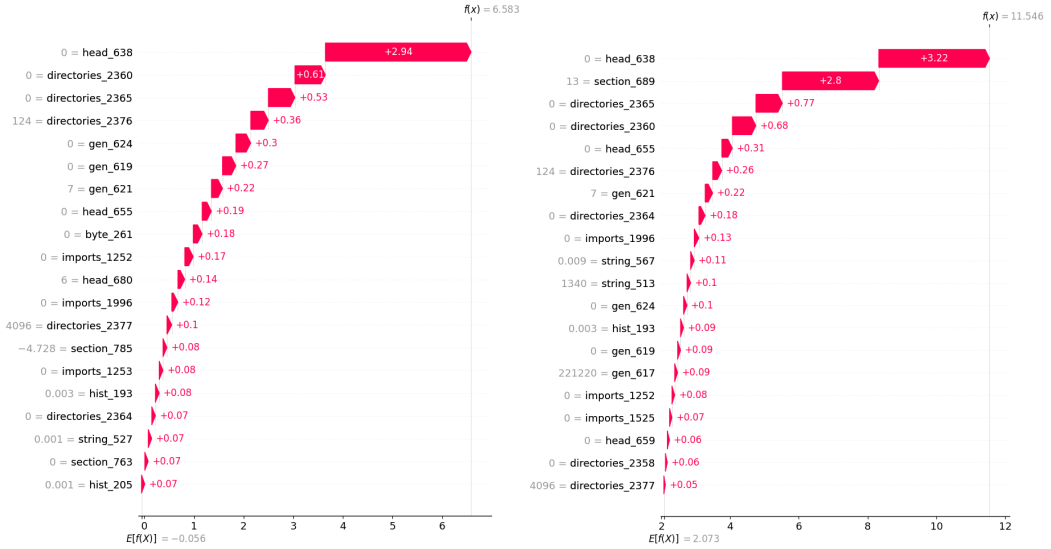
To complete the analysis of the performance of adversarial training with GAMMA, we examined explanations computed with SHAP for decisions yielded via $LGBM^O$ and $LGBM^{AT}$, respectively. We conducted this explanation analysis using the adversarial PE malware produced with GAMMA, which changed from being misclassified as goodware with $LGBM^O$ to being correctly classified as malware with $LGBM^{AT}$. Figure 6 shows the distribution of the top 20 Shapely values averaged on the decisions yielded via $LGBM^O$ and $LGBM^{AT}$ with the selected samples. The plots show that the use of adversarial training caused several changes in the top 20 Shapley-based feature ranking of $LGBM^O$ and $LGBM^{AT}$. The changes observed in the effect of features on decisions explain how adversarial training modified the decision-making process of LGBM to allow the decision model to recognise some of the adversarial PE malware produced with GAMMA and originally misclassified with $LGBM^O$. For example, we note that both high values of "section_689" and high values of "import_1252" gained relevance from $LGBM^O$ to $LGBM^{AT}$, in order to recognise the malicious behaviour of the selected adversarial PE malware. Instead, we note a change in the relationship between the Shapley value distribution and the feature value distribution of "byte_509". In fact, the higher, positive Shapley values of "byte_509" that were measured in correspondence with high values of "byte_509" for $LGBM^O$ decisions were measured in correspondence with the low values of "byte_509" for $LGBM^{AT}$ decisions.



(**a**) $LGBM^{O}$ (**b**) $LGBM^{AT}$

Figure 6. Global Shapley values measured for the top 20 input features (axis *Y*) of $LGBM^{O}$ (**a**) and $LGBM^{AT}$ using GAMMA (**b**). The global Shapley values are plotted with respect to the feature value (axis *X*) for the 72 PE adversarial malware generated with GAMMA, which were wrongly classified in the "goodware" class according to $LGBM^{O}$ and correctly classified in the "malware" class according to $LGBM^{AT}$. Changes in the top ranking of the importance of input features for the model's decisions are marked in (**b**).

To complete the analysis of explanations reported in Figure 6, we selected a single Windows PE malware file from the 72 PE malware files explained in Figure 6. Figure 7 shows the top 20 local Shapley values of the input features that have a higher effect on the decisions yielded for the selected sample using $LGBM^{O}$ and $LGBM^{AT}$, respectively. This figure also shows values measured for the study sample with the top-ranked input features, as well as the range of values that each top-ranked feature assumes in the set of selected Windows PE malware files explained in Figure 6. The local explanation values produced for the two decisions yielded via both $LGBM^{O}$ and $LGBM^{AT}$ for this sample confirm that the correct classification achieved with $LGBM^{AT}$ can be partially explained by the fact that feature section_689 (which assumes a value equal to 13, which is in the middle part of the feature range [3, 40]), string_567 (which assumes the value equal to 0.009, which is in the lower part of the feature range [0.0056486, 0.0421279), string_513 (which assumes the value 1340, which is in the lower part of the feature range [52, 18,190]) and directories_2358 (which assumes the feature value 0, which is in the lower part of the feature range [0, 48]) gain importance in explaining the model's decision when the correct decision is yielded via the decision model $LGBM^{AT}$, which was trained using adversarial training.



(**a**) $LGBM^{O}$ (**b**) $LGBM^{AT}$

feature	range	feature	range	feature	range
byte_261	[0, 0.00022]	directories_2358	[0, 48]	directories_2360	[0, 11,704]
directories_2364	[0, 56]	directories_2365	[0, 37,654]	directories_2376	[0, 22,520]
directories_2377	[0, 37,642]	gen_617	[4608, 4141139]	gen_619	[0, 1]
gen_621	[1, 636]	gen_624	[0, 1]	head_638	[0, 0]
head_655	[0, 1]	head_659	[−2, 0]	head_680	[2, 12]
hist_193	[0.00077, 0.01101]	hist_205	[0.00026, 0.01829]	imports_1252	[−2, 0]
imports_1253	[−2, 2]	imports_1525	[0, 1]	imports_1996	[−2, 0]
section_689	[3, 40]	section_763	[−213.6437, 5.14889]	section_785	[−16.38742, 4.461381]
string_513	[52, 18190]	string_527	[0, 0.00534]	string_567	[0.00564, 0.04212]

(**c**) Feature range

Figure 7. Local Shapley values (axis X) measured for the top 20 input features (axis y) of the decisions yielded via $LGBM^{O}$ (**a**) and $LGBM^{AT}$ (**b**), respectively, for a a Windows PE malware file generated with GAMMA. This is one of the files in the file set explained in Figure 6. It was wrongly classified as goodware according to $LGBM^{O}$, while it was correctly classified as malicious according to $LGBM^{AT}$. For each input feature, the value assumed by the feature in the study file is shown in the feature name reported on the axis Y in the FeatureName = FeatureValue format. The range of values that the ranked input features assume in the 72 PE malware files explained in Figure 6 is shown in (**c**). The Shapely value measured for each feature is reported in the corresponding feature bar. The higher the Shapley value, the more important the effect of the input feature on the decision using the LGBM model for the considered sample.

6. Conclusions

Over the past decade, the proliferation and application of machine learning methods in several Windows PE malware detection studies have spurred the evolution of a new generation of Windows PE malware detection systems that have been integrated into several cyber defence platforms. On the other hand, recent advancements in adversarial learning have shown that, alongside traditional cyber attacks, machine learning methods have created an additional attack vector. In fact, machine learning models may be subject to cyber attacks called adversarial samples, like any other software system. Such attacks may have severe consequences on cyber defence systems, as adversaries could potentially bypass the machine learning-based Windows PE anti-malware systems. Therefore, machine learning models for Windows PE malware detection must be extensively evaluated in the context of, and possibly secured against, realistic adversarial machine learning attacks.

In this paper, we have illustrated the results of an extensive evaluation study on the performance of five state-of-the-art attack methods used to produce realistic adversarial Windows PE malware files. The study aimed to understand how an adversarial method could actually fool a machine learning model trained for Windows PE malware detection and explore how

the adversarial training strategy could be used as a means of improving the security of the machine learning model against adversarial attacks. Notably, in this study, we considered realistic Windows PE attacker methods that use the machine learning model feedback (i.e., gradient information in white-box attack methods and model decisions in black-box attack methods) to modify unused locations of Windows PE malware files by preserving the executable structure of Windows PE binary files, and guaranteeing that the functionality of the binary files is not compromised [15]. Although attacks were produced in the raw byte-based space, we analysed their transferability to the LIEF feature engineered-based space, where a more accurate decision model can be learned to address the Windows PE malware detection task. We performed a new exploratory study to explain how the attack can also fool this model. More importantly, we explored the effectiveness of adversarial training as a defensive strategy to make the decision model learned in the feature-engineered-based space more robust to the adversarial malware considered in this study. Our work is founded on the idea that machine learning methods for Windows PE malware detection must demonstrate their robustness to adversarial actions to transition from academia to become a crucial component of the cyber defence security line of public and private companies. Exploring the potential vulnerabilities of machine learning models is the first step to performing the adversarial defences for machine learning-based Windows PE malware detection against adversarial attacks.

We are aware that several studies have recently surveyed Windows PE attack methods [5,8], also conducting a systematic evaluation study of their offensive performance [5]. However, to the best of our knowledge, we have conducted a comparative analysis using a larger collection of more recent Windows PE files. As an additional contribution, we explored possible relationships between the actual offensive ability of the considered attack methods and the distance of realistic adversarial PE malware from the original malware. The distance analysis was performed in both the raw byte-based space and the LIEF engineered feature-based space. In addition, we used an XAI method—SHAP—to explain the effect of adversarial Windows PE malware on decisions produced according to engineered features. This explanation analysis was conducted to explore possible relationships between the offensive ability of attack methods and changes observed in decision explanations. On the other hand, this explanation analysis was also conducted to explain the effect of an adversarial training strategy.

In future work, we plan to extend this investigation to further Windows PE attack methods comprising poisoning methods [32], which are commonly studied to subvert learning with injected poisoned samples. In addition, we note that a work conducted under the umbrella of adversarial learning must be orthogonal to a work concerning common adversary tactics that are not specifically tailored to evading machine learning models. Modern malware uses common obfuscation techniques such as encryption, oligomorphic, polymorphic, metamorphic, stealth and packing methods to make the detection process more difficult [49]. In addition, armouring techniques can be used in malware to delay or stop the analysis of an executable through behavioural observation and/or reverse engineering [50]. Based on these considerations, our future investigations will include an exploration of the effect of adversarial training on files generated using re-coding/armouring adversary tactics. Finally, we intend to conduct a similar study in the field of Android malware detection [51], taking into account the fact that the Android operating system (OS) has been the leading platform for mobile devices since 2012. In this domain, [52] recently formalised the problem of realistic adversarial Android attacks, while attack methods against Android malware decision models have been studied in [53,54].

Author Contributions: Conceptualization, A.A., D.M. and M.I.; methodology, A.A., D.M. and M.I.; software, M.I.; validation, M.I.; investigation, A.A., D.M. and M.I.; data curation, M.I.; writing—original draft preparation, A.A. and D.M.; writing—review and editing, A.A., D.M. and M.I.; visualization, M.I.; supervision, A.A. and D.M.; project administration, A.A.; funding acquisition, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: The research of Muhammad Imran was supported by PON RI 2014–2020—Artificial Intelligence in malware and intrusion detection, CUP H99J21010060001. Annalisa Appice and Donato Malerba were partially supported by project SERICS (PE00000014) under the NRRP MUR National Recovery and Resilience Plan funded by the European Union—NextGenerationEU.

Data Availability Statement: The study's data and codes are available in a publicly accessible repository (https://github.com/MuhammdImran/Windows-PE-Adversarial-Attacks.git, accessed on 30 April 2024).

Acknowledgments: The authors wish to thank the anonymous reviewers for their helpful suggestions to improve this paper.

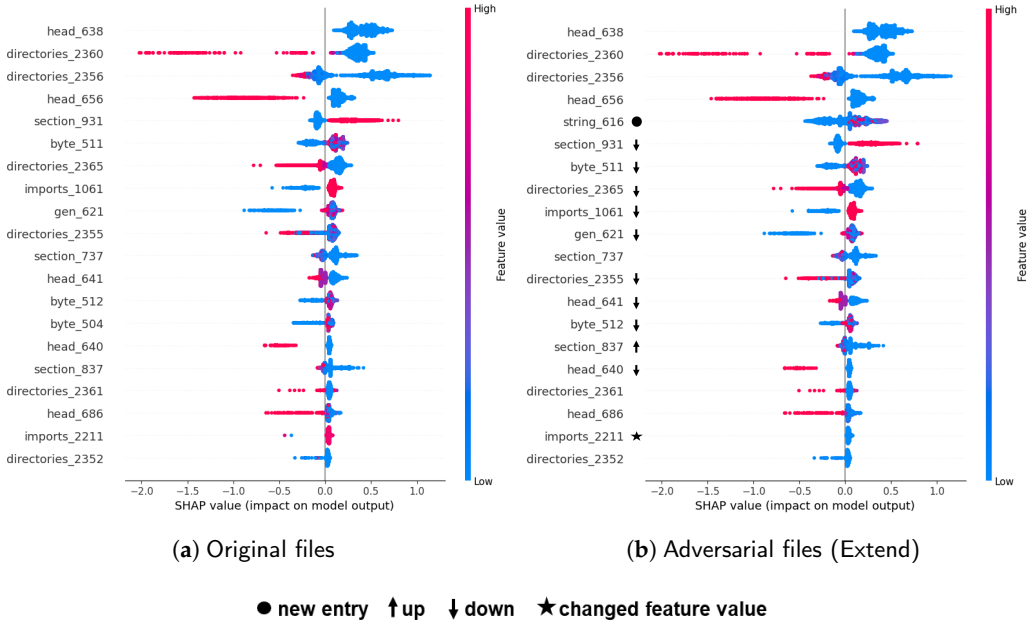
Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

In this appendix, we report the results of the study's analysis of the Shapley values computed for the decisions yielded with the pre-trained LGBM model for the adversarial malware files generated with the attack methods Extend, Full DOS, Shift and FGSM padding + slack. We consider the adversarial Windows PE malware files that fooled the LGBM model. We recall that the analysis of the Shapley values produced for GAMMA is illustrated in Section 5.4.

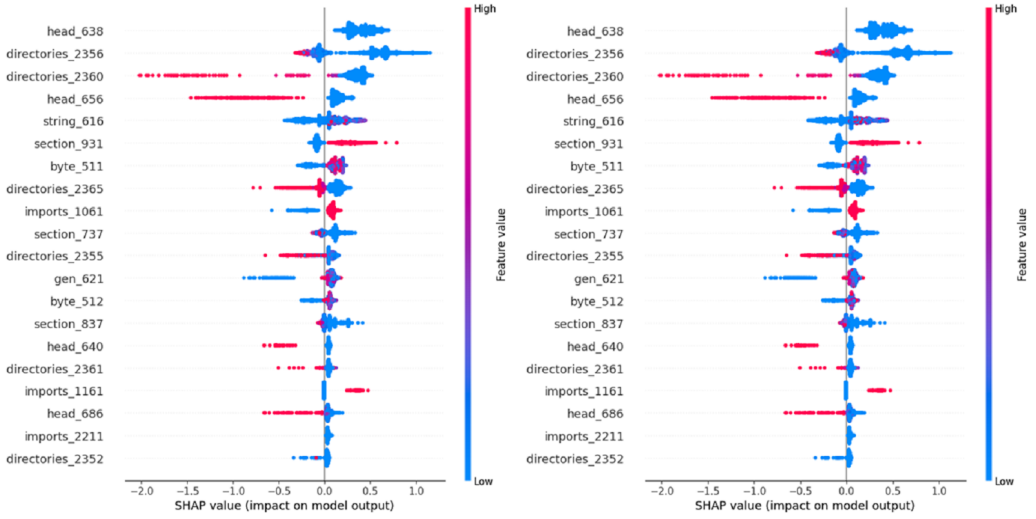
Figures A1–A4 show the distribution of the average Shapley values computed with respect to the "malware" class for the top 20 engineered features of the input feature space of LGBM. In particular, the left-side charts (Figures A1a–A4a) show the Shapley values computed to explain the decisions concerning the original Windows PE malware, while the right-side charts (Figures A1b–A4b) show the Shapley values computed to explain the decisions concerning its adversarial malware counterparts.

Notably, the plots do not show any change in the top 20 features ranked according to the Shapley values for Full DOS, which are shown in Figure A2. This means that LGBM works quite similarly to decide about both Windows PE malware files and their adversarial counterparts produced through Full DOS. This is unsurprising since no adversarial sample produced via Full DOS evaded LGBM. Even 10 Windows PE malware files that were originally misclassified according to LGBM were correctly detected as malware when processed as they were modified via Full DOS (see Table 3). On the other hand, only a few changes appeared in the top 20 features ranked according to the Shapley values for FGSM padding + slack, which are shown in Figure A4. This is the other attack method of this evaluation study that generated adversarial PE malware capable of evading MalConv but not LGBM (see Table 3). On the contrary, numerous changes appeared in the top 20 Shapley-based feature rankings of Extend, shown in Figure A1, and Shift, shown in Figure A3. Both methods, similar to GAMMA, were able to generate some realistic adversarial malware capable of evading LGBM in addition to MalConv (see Table 3). The higher number of changes is seen in the explanation of decisions produced for adversarial malware files produced using Extend, which was the runner-up for GAMMA as the most effective attack method in this study. Hence, this additional exploratory analysis, which was performed to explain how the attack methods Extend, Full DOS, Shift and FGSM padding+slack change the effect of features on decisions yielded via the LGBM model, supports the conclusion drawn in Section 5.4: there is a relationship between the effectiveness of the transferability of a Windows PE attack from MalConv to LGBM and the ability of this attack to introduce some relevant changes in decisions produced via LGBM with transferred adversarial samples.



(**a**) Original files (**b**) Adversarial files (Extend)

Figure A1. Global Shapley values measured for the top 20 input features (axis *Y*) of the LGBM input space and plotted with respect to the feature value (axis *X*) for the original 7951 PE malware files (**a**) and the adversarial counterparts (**b**) produced using Extend. Changes in the Shapley value-based feature ranking are marked in (**b**).



(**a**) Original files (**b**) Adversarial files(Full DOS)

Figure A2. Shapley values measured for the top 20 input features (axis *Y*) of the LGBM input space and plotted with respect to the feature value (axis *X*) for the original 6115 PE malware files (**a**) and the adversarial counterparts (**b**) produced using Full DOS. The two charts show that the same feature ranking is obtained in the two groups of files.

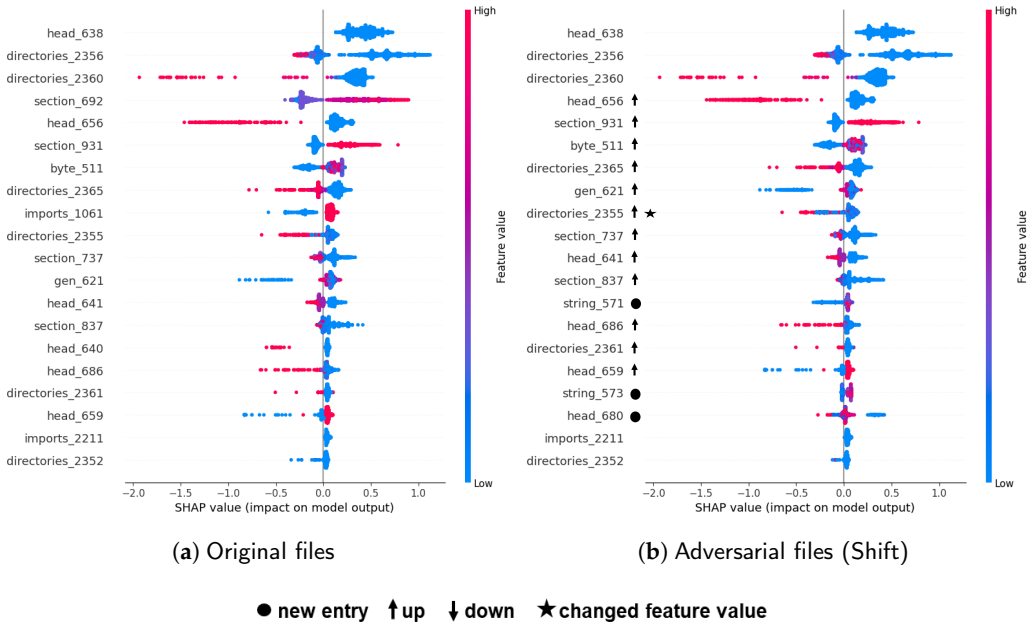


Figure A3. Shapley values measured for the top 20 input features (axis *Y*) of the LGBM input space and plotted with respect to the feature value (axis *X*) for the original 3423 PE malware files (**a**) and the adversarial counterparts (**b**) produced using Shift. Changes in the Shapley value-based feature ranking are marked in (**b**).

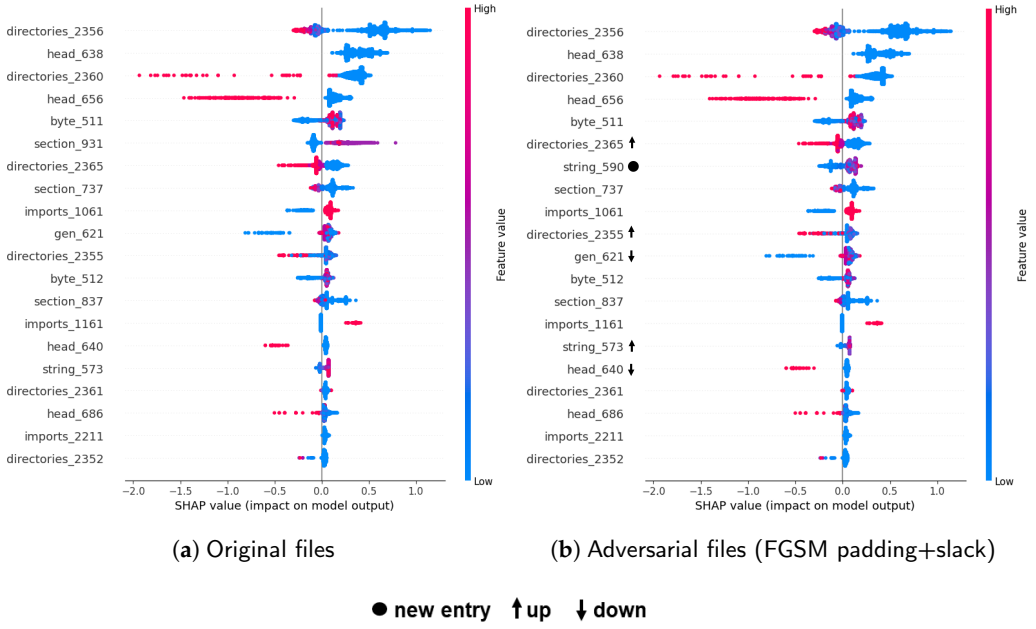


Figure A4. Shapley values measured for the top 20 input features (axis *Y*) of the LGBM input space and plotted with respect to the feature value (axis *X*) for the original 4384 PE malware files (**a**) and the adversarial counterparts (**b**) produced using FGSM padding+slack. Changes in the Shapley value-based feature ranking are marked in (**b**).

References

1. Singh, J.; Singh, J. A survey on machine learning-based malware detection in executable files. *J. Syst. Archit.* **2021**, *112*, 101861. [CrossRef]
2. Tayyab, U.e.H.; Khan, F.B.; Durad, M.H.; Khan, A.; Lee, Y.S. A Survey of the Recent Trends in Deep Learning Based Malware Detection. *J. Cybersecur. Priv.* **2022**, *2*, 800–829. [CrossRef]
3. Gopinath, M.; Sibi Chakkaravarthy, S. A comprehensive survey on deep learning based malware detection techniques. *Comput. Sci. Rev.* **2023**, *47*, 100529. [CrossRef]

4. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.J.; Fergus, R. Intriguing properties of neural networks. In Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, 14–16 April 2014; Conference Track Proceedings; Bengio, Y., LeCun, Y., Eds.; pp. 1–10; arXiv:1312.6199.
5. Demetrio, L.; Coull, S.E.; Biggio, B.; Lagorio, G.; Armando, A.; Roli, F. Adversarial EXEmples: A Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection. *ACM Trans. Priv. Secur.* **2021**, *24*, 27. [CrossRef]
6. Demetrio, L.; Biggio, B.; Roli, F. Practical Attacks on Machine Learning: A Case Study on Adversarial Windows Malware. *IEEE Secur. Priv.* **2022**, *20*, 77–85. [CrossRef]
7. Liang, H.; He, E.; Zhao, Y.; Jia, Z.; Li, H. Adversarial Attack and Defense: A Survey. *Electronics* **2022**, *11*, 1283. [CrossRef]
8. Ling, X.; Wu, L.; Zhang, J.; Qu, Z.; Deng, W.; Chen, X.; Qian, Y.; Wu, C.; Ji, S.; Luo, T.; et al. Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art. *Comput. Secur.* **2023**, *128*, 103134. [CrossRef]
9. Raff, E.; Barker, J.; Sylvester, J.; Brandon, R.; Catanzaro, B.; Nicholas, C.K. Malware detection by eating a whole exe. In Proceedings of the Workshops at the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
10. Anderson, H.S.; Roth, P. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *arXiv* **2018**, arXiv:1804.04637.
11. Li, K.; Guo, W.; Zhang, F.; Du, J. GAMBD: Generating adversarial malware against MalConv. *Comput. Secur.* **2023**, *130*, 103279. [CrossRef]
12. Liu, L.; Kuang, X.; Liu, L.; Zhang, L. Defend against adversarial attacks in malware detection through attack space management. *Comput. Secur.* **2024**, *141*, 103841. [CrossRef]
13. Barut, O.; Zhang, T.; Luo, Y.; Li, P. A Comprehensive Study on Efficient and Accurate Machine Learning-Based Malicious PE Detection. In Proceedings of the 2023 IEEE 20th Consumer Communications & Networking Conferencem CCNC 2023, Las Vegas, NV, USA, 8–11 January 2023; pp. 632–635. [CrossRef]
14. Kreuk, F.; Barak, A.; Aviv-Reuven, S.; Baruch, M.; Pinkas, B.; Keshet, J. Adversarial Examples on Discrete Sequences for Beating Whole-Binary Malware Detection. *arXiv* **2018**, arXiv:1802.04528.
15. Demetrio, L.; Biggio, B.; Lagorio, G.; Roli, F.; Armando, A. Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 3469–3478. [CrossRef]
16. Demetrio, L.; Biggio, B. secml-malware: A Python Library for Adversarial Robustness Evaluation of Windows Malware Classifiers. *arXiv* **2021**, arXiv:2104.12848.
17. Lundberg, S.M.; Lee, S.I. A Unified Approach to Interpreting Model Predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS 2017, Long Beach, CA, USA, 4–9 December 2017; NIPS; Curran Associates Inc.: Glasgow, UK, 2017; pp. 4768–4777. [CrossRef]
18. Chen, B.; Ren, Z.; Yu, C.; Hussain, I.; Liu, J. Adversarial Examples for CNN-Based Malware Detectors. *IEEE Access* **2019**, *7*, 54360–54371. [CrossRef]
19. Adeke, J.M.; Liu, G.; Zhao, J.; Wu, N.; Bashir, H.M. Securing Network Traffic Classification Models against Adversarial Examples Using Derived Variables. *Future Internet* **2023**, *15*, 405. [CrossRef]
20. Alotaibi, A.; Rassam, M.A. Adversarial Machine Learning Attacks against Intrusion Detection Systems: A Survey on Strategies and Defense. *Future Internet* **2023**, *15*, 62. [CrossRef]
21. Al-Essa, M.; Andresini, G.; Appice, A.; Malerba, D. PANACEA: A Neural Model Ensemble for Cyber-Threat Detection. *Mach. Learn. J.* **2024**, 1–44. *in press*. [CrossRef]
22. Chen, H.; Babar, M.A. Security for Machine Learning-based Software Systems: A Survey of Threats, Practices, and Challenges. *ACM Comput. Surv.* **2024**, *56*, 1–38. [CrossRef]
23. Bishop, C.M.; Nasrabadi, N.M. Pattern Recognition and Machine Learning. *J. Electron. Imaging* **2007**, *16*, 049901. [CrossRef]
24. Kattamuri, S.J.; Penmatsa, R.K.V.; Chakravarty, S.; Madabathula, V.S.P. Swarm Optimization and Machine Learning Applied to PE Malware Detection towards Cyber Threat Intelligence. *Electronics* **2023**, *12*, 342. [CrossRef]
25. Ucci, D.; Aniello, L.; Baldoni, R. Survey of machine learning techniques for malware analysis. *Comput. Secur.* **2019**, *81*, 123–147. [CrossRef]
26. Harang, R.E.; Rudd, E.M. SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection. *arXiv* **2020**, arXiv:2012.07634.
27. Yang, L.; Ciptadi, A.; Laziuk, I.; Ahmadzadeh, A.; Wang, G. BODMAS: An open dataset for learning based temporal analysis of PE malware. In Proceedings of the 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 27 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 78–84. [CrossRef]
28. Svec, P.; Balogh, S.; Homola, M.; Kluka, J. Knowledge-Based Dataset for Training PE Malware Detection Models. *arXiv* **2022**, arXiv:2301.00153.
29. Catak, F.O.; Yazı, A.F.; Elezaj, O.; Ahmed, J. Deep learning based Sequential model for malware analysis using Windows exe API Calls. *PeerJ Comput. Sci.* **2020**, *6*, e285. [CrossRef] [PubMed]
30. Bosansky, B.; Kouba, D.; Manhal, O.; Sick, T.; Lisy, V.; Kroustek, J.; Somol, P. Avast-CTU Public CAPE Dataset. *arXiv* **2022**, arXiv:2209.03188.
31. Chakraborty, A.; Alam, M.; Dey, V.; Chattopadhyay, A.; Mukhopadhyay, D. A survey on adversarial attacks and defences. *CAAI Trans. Intell. Technol.* **2021**, *6*, 25–45. [CrossRef]

32. Tian, Z.; Cui, L.; Liang, J.; Yu, S. A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning. *ACM Comput. Surv.* **2022**, *55*, 1–35. [CrossRef]
33. Khamaiseh, S.Y.; Bagagem, D.; Al-Alaj, A.; Mancino, M.; Alomari, H.W. Adversarial Deep Learning: A Survey on Adversarial Attacks and Defense Mechanisms on Image Classification. *IEEE Access* **2022**, *10*, 102266–102291. [CrossRef]
34. Muoka, G.W.; Yi, D.; Ukwuoma, C.C.; Mutale, A.; Ejiyi, C.J.; Mzee, A.K.; Gyarteng, E.S.A.; Alqahtani, A.; Al-antari, M.A. A Comprehensive Review and Analysis of Deep Learning-Based Medical Image Adversarial Attack and Defense. *Mathematics* **2023**, *11*, 4272. [CrossRef]
35. Cinà, A.E.; Grosse, K.; Demontis, A.; Biggio, B.; Roli, F.; Pelillo, M. Machine Learning Security Against Data Poisoning: Are We There Yet? *Computer* **2024**, *57*, 26–34. [CrossRef]
36. Macas, M.; Wu, C.; Fuertes, W. Adversarial examples: A survey of attacks and defenses in deep learning-enabled cybersecurity systems. *Expert Syst. Appl.* **2024**, *238*, 122223. [CrossRef]
37. Li, D.; Li, Q.; Ye, Y.F.; Xu, S. Arms Race in Adversarial Malware Detection: A Survey. *ACM Comput. Surv.* **2021**, *55*, 1–35. [CrossRef]
38. Galovic, M.; Bosanský, B.; Lisý, V. Improving Robustness of Malware Classifiers using Adversarial Strings Generated from Perturbed Latent Representations. *arXiv* **2021**, arXiv:2110.11987.
39. Tong, L.; Li, B.; Hajaj, C.; Xiao, C.; Zhang, N.; Vorobeychik, Y. Improving Robustness of ML Classifiers against Realizable Evasion Attacks Using Conserved Features. In Proceedings of the 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, 14–16 August 2019; Heninger, N., Traynor, P., Eds.; USENIX Association: Berkeley, CA, USA, 2019; pp. 285–302.
40. Lucas, K.; Pai, S.; Lin, W.; Bauer, L.; Reiter, M.K.; Sharif, M. Adversarial Training for Raw-Binary Malware Classifiers. In Proceedings of the 32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, 9–11 August 2023; Calandrino, J.A., Troncoso, C., Eds.; USENIX Association: Berkeley, CA, USA, 2023; pp. 1163–1180.
41. Bala, N.; Ahmar, A.; Li, W.; Tovar, F.; Battu, A.; Bambarkar, P. DroidEnemy: Battling adversarial example attacks for Android malware detection. *Digit. Commun. Netw.* **2022**, *8*, 1040–1047. [CrossRef]
42. Shafin, S.S.; Ahmed, M.M.; Pranto, M.A.; Chowdhury, A. Detection of android malware using tree-based ensemble stacking model. In Proceedings of the 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), Brisbane, Australia, 8–10 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6. [CrossRef]
43. Khoda, M.E.; Kamruzzaman, J.; Gondal, I.; Imam, T.; Rahman, A. Malware detection in edge devices with fuzzy oversampling and dynamic class weighting. *Appl. Soft Comput.* **2021**, *112*, 107783. [CrossRef]
44. D'Orazio, C.J.; Lu, R.; Choo, K.K.R.; Vasilakos, A.V. A Markov adversary model to detect vulnerable iOS devices and vulnerabilities in iOS apps. *Appl. Math. Comput.* **2017**, *293*, 523–544. [CrossRef]
45. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS 2017, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Glasgow, UK, 2017; pp. 3149–3157.
46. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015; Conference Track Proceedings; Bengio, Y., LeCun, Y., Eds.; 2015; pp. 1–11; arXiv:1412.6572.
47. Šarčević, A.; Pintar, D.; Vranić, M.; Krajna, A. Cybersecurity Knowledge Extraction Using XAI. *Appl. Sci.* **2022**, *12*, 8669. [CrossRef]
48. Ndichu, S.; Kim, S.; Ozawa, S.; Ban, T.; Takahashi, T.; Inoue, D. Detecting Web-Based Attacks with SHAP and Tree Ensemble Machine Learning Methods. *Appl. Sci.* **2022**, *12*, 60. [CrossRef]
49. Aslan, Ö.A.; Samet, R. A comprehensive review on malware detection approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]
50. Mohanta, A.; Saldanha, A.; Mohanta, A.; Saldanha, A. Armoring and Evasion: The Anti-Techniques. In *Malware Analysis and Detection Engineering: A Comprehensive Approach to Detect and Analyze Modern Malware*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 691–720.
51. Guerra-Manzanares, A. Machine Learning for Android Malware Detection: Mission Accomplished? A Comprehensive Review of Open Challenges and Future Perspectives. *Comput. Secur.* **2024**, *138*, 103654. [CrossRef]
52. Pierazzi, F.; Pendlebury, F.; Cortellazzi, J.; Cavallaro, L. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In Proceedings of the 2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, 18–21 May 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1332–1349. [CrossRef]
53. Chen, S.; Xue, M.; Fan, L.; Hao, S.; Xu, L.; Zhu, H.; Li, B. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. *Comput. Secur.* **2018**, *73*, 326–344. [CrossRef]
54. Li, D.; Li, Q. Adversarial Deep Ensemble: Evasion Attacks and Defenses for Malware Detection. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3886–3900. [CrossRef]