applied sciences

MDPI

Article

Leveraging Adversarial Samples for Enhanced Classification of Malicious and Evasive PDF Files

Fouad Trad *, Ali Hussein and Ali Chehab

Electrical and Computer Engineering, American University of Beirut, Beirut 1107-2020, Lebanon; ah203@aub.edu.lb (A.H.); chehab@aub.edu.lb (A.C.)
* Correspondence: fat10@mail.aub.edu

Abstract: The Portable Document Format (PDF) is considered one of the most popular formats due to its flexibility and portability across platforms. Although people have used machine learning techniques to detect malware in PDF files, the problem with these models is their weak resistance against evasion attacks, which constitutes a major security threat. The goal of this study is to introduce three machine learning-based systems that enhance malware detection in the presence of evasion attacks by substantially relying on evasive data to train malware and evasion detection models. To evaluate the robustness of the proposed systems, we used two testing datasets, a real dataset containing around 100,000 PDF samples and an evasive dataset containing 500,000 samples that we generated. We compared the results of the proposed systems to a baseline model that was not adversarially trained. When tested against the evasive dataset, the proposed systems provided an increase of around 80% in the f1-score compared to the baseline. This proves the value of the proposed approaches towards the ability to deal with evasive attacks.

Keywords: malicious documents; PDF malware detection; adversarial training; machine learning; evasion attacks; model robustness; evasion detection; evasive data generation

Citation: Trad, F.; Hussein, A.; Chehab, A. Leveraging Adversarial Samples for Enhanced Classification of Malicious and Evasive PDF Files. *Appl. Sci.* **2023**, *13*, 3472. https://doi.org/10.3390/10.3390/app13063472

Academic Editor: Luis Javier García Villalba

Received: 5 February 2023
Revised: 24 February 2023
Accepted: 27 February 2023
Published: 8 March 2023

1. Introduction

In recent years, the threat of cyberattacks has become an increasingly pressing issue for businesses and individuals alike [1]. As we become more reliant on technology in our daily lives, the potential for malicious actors to exploit vulnerabilities and disrupt operations has increased as well, which makes it essential to ensure the security of different systems. Cyberattacks come in many forms, from phishing and social engineering to network intrusions and denial of service attacks [2]. However, one of the most common forms of attacks is through the use of malicious files, which poses a significant risk to cybersecurity since such files can be used to steal sensitive information, cause system damage, and disrupt operations [3]. One of the most used file types for spreading malware is the Portable Document Format (PDF).

The Portable Document Format (PDF) is a popular format used for electronic file exchange across platforms and applications. The reasons behind this popularity include the flexibility of the format, its multi-purpose usage, and its independence from operating systems. Moreover, it supports multiple features like object embedding, system command injections, and JavaScript functionalities. However, these characteristics offer attackers additional options to spread malware [4]. Although traditional antivirus tools can detect malware in PDF files, the fast-evolving nature of cyberattacks gives rise to new types of attacks that cannot be detected using the traditional signature-based detection methods. Attackers are constantly developing new tactics and techniques to evade detection, rendering antivirus tools ineffective in detecting the newly emerging or previously unknown malware signatures [5].

To address this issue, researchers started using machine learning techniques [6–8], to extract features from PDFs that are useful to differentiate malware from benign files,

and to use them in the training process of malware classifiers. Machine learning-based security systems rely on statistical models and algorithms that can learn from data and adapt to new and emerging attacks. However, machine learning-based security systems also have their own challenges, as they are prone to adversarial attacks [9] where an attacker can carefully modify the samples to trick the machine learning model into predicting the wrong category. As such, the research is currently shifting into building malware classifiers that are robust against adversarial attacks [10], which is the main objective of this work.

In the context of PDF, evasion attacks are the most popular ones. The use of evasion attacks against machine learning models is fairly common. The idea behind evasion attacks is to carefully modify samples' features so that the model is tricked in the classification process [11]. In the PDF context, the goal of evasion attacks is to design samples that can trick the classifier into believing that a malicious file is a benign one or vice versa. A substantial amount of research has recently focused on making PDF malware classifiers more robust to evasion attacks. The proposed techniques rely on improved feature engineering, and on training more complex model architectures [9,12–14]. However, in these studies, non-evasive data are used for training the models, and evasive data are used to evaluate their robustness. Li et al. [15] tried to break the loop and performed active learning to deal with evasion; they assumed that evasive samples cause uncertain malware predictions, and hence, they require human labeling to check for malware content before retraining the model on these samples.

In summary, previous works focused on making the malware detectors more robust in terms of architecture, and some of them labeled the uncertain predictions—considering that they originate from evasion attacks—and used them for retraining. However, there are two issues with such an approach, (1) there is a need to identify the range of predicted scores for the uncertain classification, and (2) there is no guarantee that evasive samples are the ones falling in the uncertainty region.

To address these issues, we propose the idea of leveraging samples that are known to be evasive and to perform adversarial learning instead of active learning, where we train the malware classifier on data containing evasive and non-evasive samples. Moreover, instead of relying on the certainty of the malware classifier's predictions to detect evasion, we propose the idea of using this mix of evasive and non-evasive data to build standalone models that detect evasion. These two missing pieces are the focus of this study, and accordingly, we propose three approaches.

1. Building a robust malware classifier by performing the training on a mix of evasive and non-evasive data.
2. Building a hierarchical system that first classifies if a PDF is evasive or not, and then, checks for malware by forwarding the PDF to a model that deals exclusively with evasive data or another model that deals only with non-evasive data.
3. Building a multi-label classifier that detects evasion and maliciousness simultaneously and independently instead of relying on two dependent models as in the second approach. This classifier is also trained on a combination of evasive and non-evasive data.

The building blocks of the three approaches are illustrated in Figure 1.

To implement these approaches, we collected training data from various sources, and they fall under two categories: evasive and non-evasive.

- Evasive data: these were used for testing or performing attacks on PDF malware classifiers in multiple previous studies [9,16–19].
- Non-evasive data: we used the Contagio dataset [20] along with the Surrogate dataset [18], which include benign files from Google, and malicious files from VirusTotal [21].

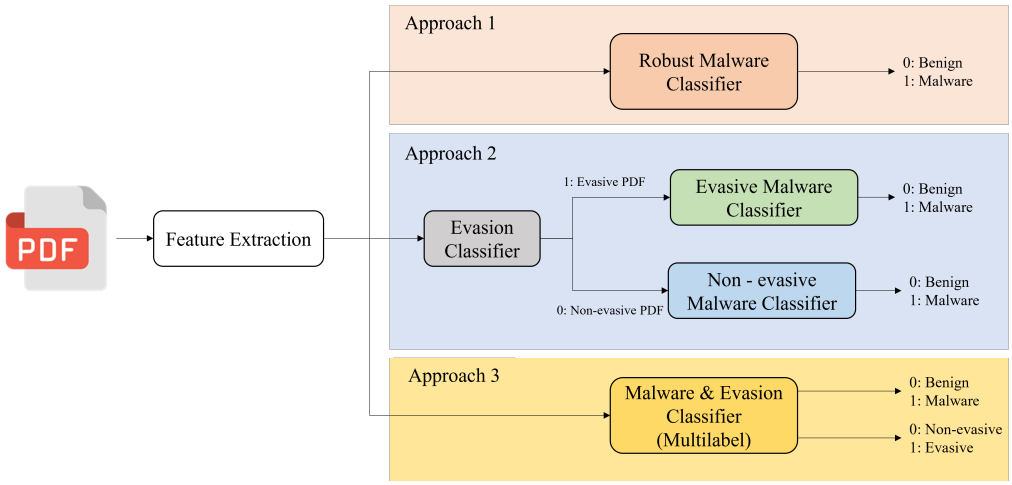

Figure 1. Building blocks of the three proposed approaches.

To assess the capabilities of the proposed approaches to detect malware under normal circumstances, as well as when dealing with evasive PDF files, we evaluated two datasets. The first one contains more than 100,000 samples collected from the network of a large university campus for 6 days [22]. The second one contains 500,000 evasive samples that we generated using a Tabular Variational AutoEncoder (TVAE) model [23] by relying on the Synthetic Data Vault (SDV) library [24]. Both test cases confirmed the robustness and reliable performance of the proposed approaches on all metrics. We compared the results with a baseline model that was trained on non-evasive samples only. The three approaches performed similarly to the baseline when tested against the university dataset: f1-score of 95% and AUC 0.99. On the other hand, the approaches provided an increase of 80% in the f1-score and an increase of 0.8 in the AUC compared to the baseline when tested against the generated evasive dataset. The f1-score of the baseline model is 15.32% and its AUC is 0.13, while the f1-score of our approaches is 95%, and the AUC is 0.95–0.97. Note that approaches 2 and 3 provided an accuracy of 88.29% and 94.23%, respectively, in detecting evasion when tested against the generated evasive dataset.

The contributions of this work are listed below:

- To the best of our knowledge, this work is the first (1) to apply adversarial retraining for PDF malware classifiers to enhance their robustness against evasion attacks and (2) to develop a standalone model capable of detecting evasion attacks in PDF files with a high accuracy, which lays the foundation for future extensions.
- We proposed two systems that can simultaneously detect evasion and maliciousness in PDF files.
- We generated a large dataset of 500,000 PDF samples with evasive-like signatures and made it publicly available, providing researchers with a valuable resource for building and testing evasion detection systems, as well as enhancing the robustness of malware detection systems.

The rest of the paper is organized as follows: some preliminaries and background information about the topic are presented in Section 2, along with the previous work that has been done, and how it differs from what is being proposed in this study. In Section 3, we introduce the adopted methodology to build the three systems along with the experimental setup, and Section 4 discusses the experiments conducted along with their results. Finally, Section 5 concludes this study and highlights potential future works.

2. Background and Preliminaries

In this section, we describe how a Portable Document Format could be used by attackers to inject malicious parts. Moreover, we discuss the basic concepts behind evasion attacks and how they can fool existing trained classifiers. We review some related works that target evasion attacks in the context of PDF files.

2.1. Portable Document Format

PDF has become the industry standard for document exchange since its inception by Adobe Systems in 1993. The format is light, easy to use, and most importantly, independent of hardware, software, and operating systems. In 2008, it became an open standard released as ISO 32000-1 [25]. PDF allows for the embedding of text, images, JavaScript, and Flash, as well as the opening of external resources from a local computer or the Internet [26]. Although these factors, among many others, contributed to the popularity of the format, yet they enabled attackers to exploit vulnerabilities in PDF document viewers. To understand how this can be done, we must look at the structure of a PDF file.

A PDF file is mainly composed of four parts as shown in Figure 2: the header, the main body, the cross-reference table, and the trailer [27].

- The Header: it is the first line of a PDF. It specifies the version of the PDF used when the document was produced. The format of the header is given by %PDF-a.b where a.b represents the version.
- The Body: it holds the main content of the PDF file represented as objects, which are the basic building blocks of a PDF, with multiple types depending on the element they hold. Note that any type of object can be obfuscated in the file structure and only de-obfuscated at runtime. This makes obfuscation an option that attackers could exploit to evade classic signature-based malware detection software.
- The cross-reference table (xref table): it lists the byte offsets of all the objects in the main body. The xref table is used to quickly access a certain object without having to search.
- The trailer: it gives the location of the xref table and the root object in the body. The trailer ends with the format %%EOF, which marks the end of the document as well.

When a reader parses a PDF file, it starts by parsing the trailer to get the reference of the root object in the file structure. Then, it uses the xref table to traverse the objects in the PDF body and render them.

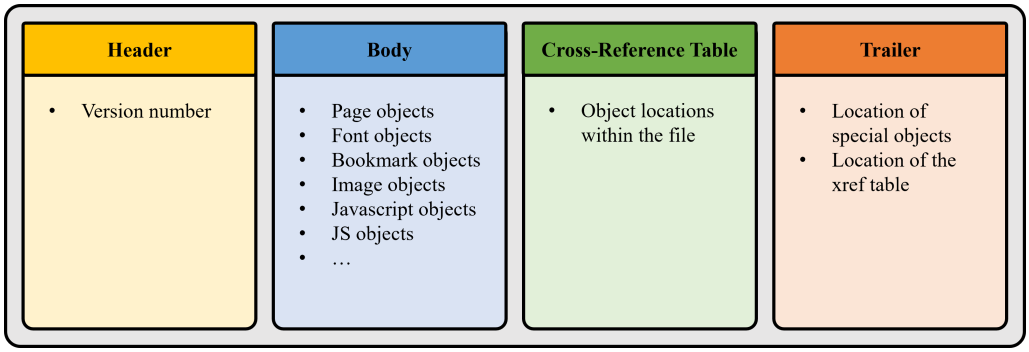

Figure 2. Structure of a PDF file.

2.2. PDF to Deliver Malware

Given that a PDF file is a great way to deliver malware, a malicious PDF exploits the vulnerabilities in the PDF reader to carry out malicious actions. There are many places where an attacker might inject malicious code. For example, she could use JavaScript functions to execute snippets of code to perform specific API calls. These functions are usually embedded in /Javascript or /JS objects [28]. In such a case, the attack code can be contained within one object, and it can span across multiple objects for the purpose of confusion. Another way would be to make use of action-based malware, where the malicious code is executed after being triggered by a specific action. The /OpenAction object, for example, would contain code that executes once the PDF is opened [29]. Moreover, there is another way to exploit vulnerabilities, by embedding other files within the PDF. These files can have various formats like tiff, bmp, exe, or even other PDF files [30].

Therefore, an attacker has multiple options to consider when trying to spread malware through a PDF file, and thus, machine learning techniques will almost certainly use features related to the structure of the PDF to classify whether a file contains malware.

2.3. PDF Evasion Attacks

Evasion attacks performed against machine learning models are very popular. The idea is to carefully manipulate the features of a sample belonging to a particular class in a way to trick the model into considering it part of another class [11]. The literature contains examples of evasion attacks against machine learning and deep learning models. Some of these attacks include the auto-projected gradient-descent attack [31], the boundary attack [32], the Brendel & Bethge adversarial attack [33], the DeepFool attack [34], the Elastic Net attack [35], the HopSkipJump attack [36] and many more. In recent years, several techniques have been proposed to improve the adversarial robustness of machine learning models. These include adversarial training, where models are trained on adversarial examples to improve their robustness against similar attacks [37], and defensive distillation, which involves training a model to mimic the output of an ensemble of models to improve its generalization performance [38]. Other techniques include input transformations, such as image randomization and feature squeezing [39], and model-based defenses such as gradient masking [40].

In the context of PDF files, evasion attacks are very popular. The most known attacks aim at making a malicious file look benign for a classifier while maintaining its malicious behavior. Some of these attacks include:

(a) EvadeML [16]: this is an evasion technique that allows object insertion, deletion, and swapping. It is stronger than other realizable attacks in the literature, which normally just permit insertion to ensure that malicious functionality is kept. This technique assumes that the adversary has black-box access to the classifier, and can only obtain the classification scores of PDF files provided as input. This technique uses genetic programming to automatically find instances that can evade the classifier while maintaining the malicious behavior, and all of this by adding, removing, and swapping objects.

(b) Mimicry [18]: this attack presupposes that an attacker is fully aware of all the features that the target classifier is using. A malicious PDF file is then modified as part of the mimicking attack to closely resemble a selected, benign PDF. Mimicry can be applied easily and without the use of a specific classification model.

(c) Reverse Mimicry [17]: this attack assumes that an attacker has zero knowledge (knows nothing) about the malware classifier. The goal of this attack is to create malware samples that have a structure similar to benign ones (which we call targets). To reduce the structural difference between the produced samples and targets, the primary idea is to inject malicious payloads into target benign files.

(d) Parser Confusion Attack [19]: this attack simply consists of obfuscating malicious Javascript code inside a PDF. This way, PDF malware classifiers would be evaded and the malicious behavior would still be executed when the files are opened.

Evasion attacks are diverse, and some of them are targeted toward a specific model (like EvadeML), while others are model-independent (like mimicry and reverse mimicry). However, in all cases, evasion attacks are transferable; an attack against a specific ML model has a high chance of being effective against a different model performing the same task [41,42]. Hence, it is essential to provide countermeasures to deal with evasion, which is what many researchers are trying to do in the context of PDF files, and we will discuss the adapted approaches in the related work section.

2.4. Related Work

The various studies that target malware detection in PDF files fall under two categories: Static Analysis and Dynamic Analysis [43].

Static analysis is the most widely used method, and it has significant benefits, including faster detection speed, lower implementation costs, and ease of use [44]. Researchers typically extract and process specific features from input files [12]; these features could be generic attributes (metadata) like the PDF size and the number of pages or retrieved from the internal structure of the PDF document (structural features). Then, these features are fed to machine learning or deep learning models to classify the corresponding files as malicious or benign. The main problem associated with static analysis is that it is prone to evasion attacks. On the other hand, the dynamic analysis examines the code's activity in real-time. Hence, the input files are run on a Virtual Machine to prevent damage to the host computer [43]. In general, dynamic analysis is more reliable than static analysis, and attackers have a harder time evading it. It is particularly beneficial for detecting malicious JavaScript code embedded in PDF files because malicious JavaScript code is frequently obfuscated, making static detection less effective.

The main disadvantage associated with dynamic analysis is that it cannot be performed directly on a personal computer because it might execute malicious behavior, and this allowed static analysis to be the preferred technique of choice. To overcome the drawbacks of static analysis, researchers resorted to enhancing feature engineering and training more complex model architectures [9,12–14]. In these studies, non-evasive data are used for training the models, and evasive data are used to evaluate their robustness. On the other hand, Li et al. [15] performed active learning to deal with evasion; the authors assumed that evasive samples cause uncertain malware predictions, and hence, they require human labeling to check for malware content before retraining the model. In fact, associating evasive samples with uncertain predictions started with the study of Smutz and Stavrou [45], which leveraged the mutual disagreement analysis between base learners in a random forest malware detector to sense the presence of an evasion attack. This method has two major drawbacks. First, we must specify the standard of uncertainty according to the prediction score (such as scores falling between 25% and 75% are uncertain and we might consider another interval as well). Second, we cannot guarantee that the samples falling in the uncertainty region are evasive because sometimes evasion attacks are strong and let the classifier predict the wrong class with high confidence. Moreover, uncertain predictions might originate from non-evasive samples. A famous example that proves these ideas is the one presented by Goodfellow et al. in [46] where an original image of a panda is correctly classified as a panda by GoogLeNet [47] with only 57.7% confidence. Then, when this image was manipulated by adding some noise to it, it was wrongly classified with a confidence of 99.3% as a gibbon, which confirms that uncertainty is not always associated with evasion. To deal with these issues, we propose the idea of leveraging samples that are known to be evasive to:

- Perform adversarial learning instead of active learning for PDF malware detectors, where we train the malware classifier on data containing evasive and non-evasive samples.
- Build standalone models that detect evasion instead of relying on the certainty of the malware classifier's predictions.

The way we can apply these two ideas will be presented in the Methodology section.

3. Materials and Methods

3.1. Methodology

To deal with the issues presented in the previous section, we propose three approaches, as shown in Figure 1. The training process is illustrated in Figure 3.

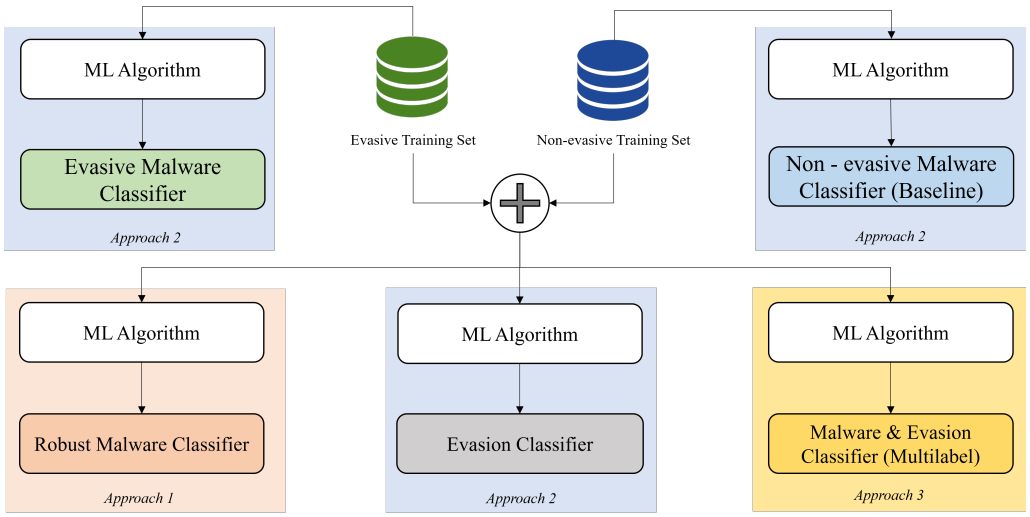

Figure 3. Training in the three approaches.

3.1.1. Approach 1

We perform adversarial training by combining the evasive and the non-evasive datasets to train a PDF malware classifier. This enables the classifier to learn the representation of evasive samples as well as normal samples to make it robust against evasion attacks, especially if they come from a distribution similar to the one it has been trained on. Although such a system can perform well on evasive data, the only disadvantage is that we cannot know if a sample is evasive or not. This information is crucial because after deploying the model, we need to retrain it on new evasive samples that it receives constantly. Having a model that detects evasion will help us know which samples are evasive and which ones are not. It will not fully automate the classification process, but it would help in reducing the human intervention for continuous retraining.

3.1.2. Approach 2

The system consists of three separately trained models operating hierarchically. At the first level, we want to detect whether a PDF is evasive or not, and at the second level, we want to detect whether it is malicious or not. The models are as follows:

1. Evasion Classifier: This model is trained on a dataset containing evasive and non-evasive samples. The goal is to detect whether a PDF is evasive or not by analyzing its features.
2. Evasive Malware Classifier: This model is trained on the evasive dataset solely, and, therefore, it expects to receive an evasive sample to classify whether it carries malware or not.
3. Non-evasive Malware Classifier: This model is trained using only the non-evasive dataset and it expects to receive a non-evasive sample to classify whether it carries malware or not.

We hierarchically combine these three models to form a system that receives a PDF file as input, extracts its corresponding features, and then feeds them to the Evasion Classifier. If the PDF is evasive, features are passed to the Evasive Malware Classifier to check for malware. If the PDF is not evasive, features are passed to the Non-evasive Malware Classifier to check for malware. The advantage of this system over the first one is that we can detect evasion in addition to detecting maliciousness. Even though in this approach the training happens separately for each model, during inference, the models depend on each other as the prediction of maliciousness is depending on whether the file was predicted to be evasive or not. For example, if an evasive PDF sample is wrongly classified as non-evasive, then it would be forwarded to the Non-evasive Malware Classifier, which might not be able to guess if it contains malware or not.

3.1.3. Approach 3

To address the problem presented in the previous approach, we use one multi-label system that predicts at the same time whether a PDF sample is evasive or not, and malicious or not. The training includes a mix of evasive and non-evasive data. The first advantage is that this system is equivalent to training two separate classifiers independently, one for evasion detection and one for malware detection. Then, during inference time, the two models will also operate independently, which means that the evasion prediction will not affect the malware prediction. The second advantage of this method over the previous one is that we are using two models instead of three. This would result in less consumption of computational resources.

3.2. Experimental Setup

3.2.1. Dataset

We gathered data from multiple sources including evasive and non-evasive samples. For the non-evasive data, the sources are

1. The Contagio malware dump dataset [20]
2. The Surrogate dataset used in [18] which contains benign files from Google, and malicious files from VirusTotal [21].

For the evasive data, the sources are

- Public datasets that were the result of different evasion attacks in previous studies: evadeML [16], mimicry [18], reverse mimicry [17], and the parser confusion attack [19]. The common about these datasets is that they only contain evasive samples that are malicious. Even though some of these datasets targeted the evasion of a particular classifier (like evadeML for instance), these datasets can conserve their evasive characteristics when dealing with other models because of the evasion transferability discussed in Section 2.3.
- EvasivePDFMal2022: a real evasive dataset that was used to test the robustness of a malware detection model in [9]. This dataset is not focused on a specific kind of attack like the previous ones. It contains real samples that hold an evasive signature. Moreover, the difference about this dataset is that it contains some samples that are evasive, yet benign, which means that these benign files will trick a classifier into considering them malicious. The kind of attack used to create such samples is not provided by the authors, but such attacks can increase the false positive rate of a malware detection system, which makes it less reliable in practice.

For the rest of the paper, these datasets would be referred to respectively as "evasive dataset" and "non-evasive dataset". The numbers of benign and malicious samples in the datasets are summarized in Table 1.

Table 1. Data Characteristics.

	Evasive Data	**Non-Evasive Data**	**Total**
Malicious	15,723	9624	25,347
Benign	4336	11,023	15,359
Total	20,059	20,647	40,706

Since the gathered data comes from multiple sources, we preprocessed the data to make it consistent. Moreover, we dropped all duplicate samples across the combination of different datasets, and we ended up with an evasive dataset that contains 20,059 samples and a non-evasive dataset that contains 20,647 samples. Next, we divided each of these datasets between training (80%) and validation (20%). For reproducibility of the results, while dividing the original data, we set the random_state parameter to 2, and we stratified it according to the class label (that determines if a PDF file is malicious or not).

3.2.2. Machine Learning Models

To obtain the best model (or models) for each of the proposed approaches, we tested in each case multiple ML algorithms (e.g., Logistic Regression, Support Vector Machine, Naïve Bayes, Random Forest Classifier, AdaBoost, LightGBM) and compared their outcomes. The process of selecting the appropriate models for each approach will be further discussed in the Training and Validation Experiments section, but as a final outcome, to train all individual models in the first two approaches, we used the same algorithm (LightGBM [48]) with the same hyperparameters: 500 base estimators with a max_depth of 7 and a max_leaf_nodes of 2. In the third approach, we used the LightGBM algorithm with 300 decision trees, each of them having a max_depth of 20 and a max_leaf_nodes of 5. The other parameters were kept as default, as provided by the sci-kit learn python's library [49]. We used the Python language (version 3.7.13) for programming on a Google Collaboratory environment.

4. Results and Discussion

4.1. Training and Validation Experiments

In this section, we describe the training experiments, starting from an initial experiment to assess the effect of evasive data on a classifier that was trained only on non-evasive data. Then, the next experiments will address the problem by applying the proposed approaches. The experimental results include the following classification metrics: accuracy, confusion matrix (TPR, TNR, FPR, FNR), and the f1-score.

4.1.1. Experiment 0: Illustrating the Problem and Establishing a Baseline

To illustrate the main problem in question, we present this experiment where we train several malware classifiers on the non-evasive training set. We trained 8 different models, selected the one that performs best, and fine-tuned it to consider it as a baseline. The algorithms we tried are Logistic Regression, Support Vector Machine (with RBF and Polynomial kernels with degrees of 1 and 2), Naïve Bayes, Random Forest Classifier, AdaBoost, and LightGBM. We first used these algorithms with their default hyperparameters as set by the sci-kit learn python's library. The performance of the resulting models on the non-evasive validation set is shown in Table 2. The results show an excellent performance of the different chosen models, with a slight improvement for the tree-based ensemble models (Random Forest, AdaBoost, and LightGBM). As such, we deferred the choice of the model to adopt. To evaluate the robustness of these models, we test each of them against the evasive validation set, and the results are reported in Table 3. We can see that the performance dropped significantly, and the classifiers that performed well on the non-evasive data, were "fooled" by the evasive data, and this can be seen from the high FNR. In the next three experiments, we implement the proposed approaches where we adopt LightGBM since it was the most robust model with an FNR of 41.14%, while the others had an FNR higher than 67.92%. We fine-tuned the hyperparameters of LightGBM using a 5-fold cross-validation method to obtain the best possible performance on the non-evasive validation data. The final hyperparameters are the ones mentioned in the Experimental Setup section. We will refer to this model as "the baseline", which we will use to compare against the outcomes of the other approaches. We evaluated this model (after fine-tuning) on the evasive and non-evasive validation sets, and the results are shown in Figure 4.

Table 2. Performance of the tested models on the non-evasive validation data.

	Logistic Regression	SVM (RBF)	SVM (Poly 1)	SVM (Poly 2)	Naïve Bayes	Random Forest	AdaBoost	LightGBM
Accuracy	94.43%	98.74%	92.32%	96.03%	82.81%	99.66%	99.32%	99.32%
TNR	91.97%	98.96%	88.30%	94.74%	79.14%	99.95%	99.55%	99.55%
FPR	8.03%	1.04%	11.70%	5.26%	20.86%	0.05%	0.45%	0.45%
FNR	2.75%	1.51%	3.06%	2.49%	12.99%	0.68%	0.94%	0.94%
TPR	97.25%	98.49%	96.94%	97.51%	87.01%	99.32%	99.06%	99.06%
F1 Score	94.21%	98.65%	91.71%	95.81%	82.51%	99.63%	99.27%	99.27%

Table 3. Performance of the tested models on the evasive validation data.

	Logistic Regression	SVM (RBF)	SVM (Poly 1)	SVM (Poly 2)	Naïve Bayes	Random Forest	AdaBoost	LightGBM
Accuracy	37.73%	44.99%	38.53%	39.15%	34.47%	39.66%	39.96%	66.28%
TNR	80.62%	91.81%	81.55%	87.20%	88.70%	97.00%	94.58%	93.19%
FPR	19.38%	8.19%	18.45%	12.80%	11.30%	3.00%	5.42%	6.81%
FNR	74.09%	67.92%	73.32%	74.09%	80.48%	76.15%	75.10%	41.14%
TPR	25.91%	32.08%	26.68%	25.91%	19.52%	23.85%	24.90%	58.86%
F1 Score	39.48%	47.76%	40.49%	40.03%	31.84%	38.26%	39.40%	73.23%

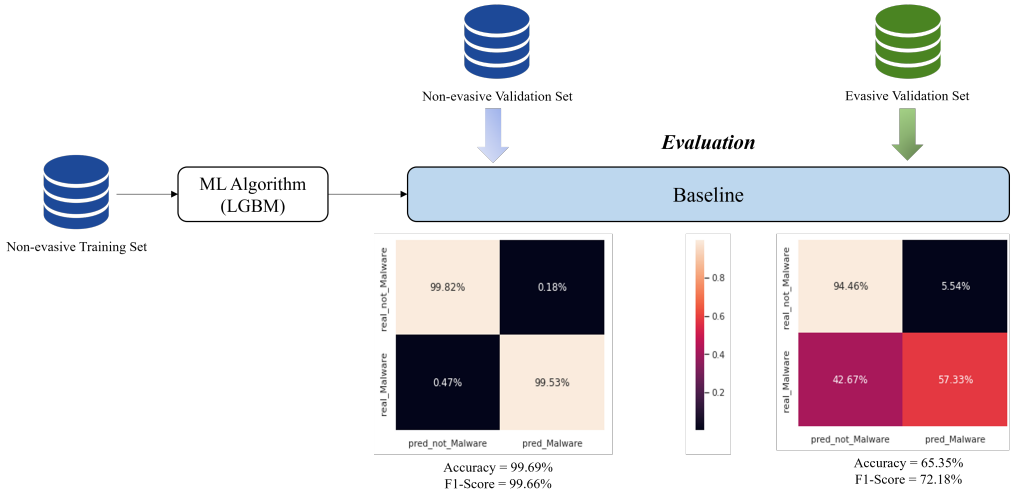

Figure 4. Performance of the baseline on the non-evasive and evasive validation datasets.

4.1.2. Experiment 1

In this experiment, we implement the first approach introduced in the Methodology section. We combine the evasive training set along with the non-evasive training set, and we train the baseline model on this mix of data to detect malware. Then, we evaluated the malware detection on the evasive validation set and the non-evasive validation set, and we obtained: accuracy = 99.48%, TNR = 99.48%, TPR = 99.49%, and f1-score = 99.59%; an excellent performance compared to the baseline.

4.1.3. Experiment 2

In this experiment, we implement the second approach, and we train and validate each of the models separately. We use the same naming convention we followed in the Methodology section.

1. Evasion Classifier: The goal of this classifier is to detect whether a PDF sample is evasive or not. Thus, this classifier would be trained on a mix of evasive and non-evasive data. To select a suitable model that can detect evasion, we test multiple algorithms. As an initial step, we keep their default hyperparameters as set by the scikit learn python's library. We train the models on the mix of evasive and non-evasive training sets and then evaluate each of them on the mix of evasive and non-evasive validation sets. The results are summarized in Table 4.
   We can see that not all models perform very well on this data, and this is expected, because detecting evasion is not an easy task with the limited features we have, and this is what explains the low performance of simple models and the good performance of the more powerful models (Random Forest, AdaBoost, and LightGBM) that are based on ensemble techniques. From the results, we can see that the most promising models to detect evasion are the Random Forest Classifier and the LightGBM. We fine-tune both and we achieve the best performance (accuracy = 96.84%, TNR = 97.97%, TPR = 95.69%, and f1-score = 96.76%) using the LightGBM algorithm with the set of hyperparameters as described in the Experimental Setup section.

2. Evasive Malware Classifier: We trained the same baseline architecture on the evasive training set only. We evaluated this model on the evasive validation set, and the results we obtained are: accuracy = 99.45%, TNR = 99.31%, TPR = 99.49%, and f1-score = 99.65%.
3. Non-evasive Malware Classifier: Here we simply used the baseline that has been trained and evaluated on the non-evasive training set and the non-evasive validation set, respectively.

After training the models individually, we combine the evasive validation set along with the non-evasive validation set and feed them to the pipeline. The malware detection metrics of the system are the following: accuracy = 99.44%, TNR = 99.45%, TPR = 99.43%, and f1-score = 99.55%. We see a slight decrease in performance compared to the first approach, but this is expected because the Evasion Classifier can make mistakes and accordingly, forward an evasive PDF to the Non-evasive Malware Classifier and vice versa.

Table 4. Performance of the Evasion Classifiers.

	Logistic Regression	SVM (RBF)	SVM (Poly 1)	SVM (Poly 2)	Naïve Bayes	Random Forest	AdaBoost	LightGBM
Accuracy	76.53%	80.59%	77.12%	78.75%	73.97%	96.28%	84.48%	95.19%
TNR	93.34%	98.18%	97.70%	96.88%	83.10%	97.82%	90.32%	97.82%
FPR	6.66%	1.82%	2.30%	3.12%	16.90%	2.18%	7.99%	2.18%
FNR	40.78%	37.51%	44.07%	39.91%	35.42%	5.31%	23.28%	7.53%
TPR	59.22%	62.49%	55.93%	60.09%	64.58%	94.69%	76.72%	92.47%
F1 Score	71.31%	76.04%	70.67%	73.60%	70.98%	96.17%	82.86%	94.98%

4.1.4. Experiment 3

In this experiment, we train a multi-label model that detects evasion and maliciousness concurrently. We test multiple models to select the best one and fine-tune it. Again, we start by using the algorithms with their default hyperparameters, then after selecting the best model, we fine-tune it accordingly. The accuracy, macro-precision, macro-recall, and macro-F1 score for each of the models are summarized in Table 5.

Table 5. Performance of each multilabel model to detect evasion and maliciousness.

	Logistic Regression	SVM (RBF)	SVM (Poly 1)	SVM (Poly 2)	Naïve Bayes	Random Forest	AdaBoost	LightGBM
Accuracy	66.37%	79.01%	69.26%	69.55%	51.29%	95.87%	82.66%	94.77%
Macro-Precision	85.04%	97.38%	90.91%	89.54%	77.74%	98.71%	95.24%	98.65%
Macro-Recall	78.57%	79.71%	77.02%	79.17%	78.91%	96.95%	86.32%	95.90%
Macro-F1 Score	79.82%	86.67%	81.13%	82.13%	74.09%	97.82%	90.34%	97.24%

We can see that the Logistic Regression, SVM and Naïve Bayes models achieve a lower performance compared to Random Forest, AdaBoost, and LightGBM. This is expected after the results we obtained in Experiment 2, which showed the weakness of these models to detect evasion. We can clearly see that the Random Forest and LightGBM models outperformed the others. This is why, we fine-tune both and we achieve the best performance (accuracy = 96.23%, macro-precision = 98.89%, macro-recall = 97.21%, macro-f1-score = 98.04%) using the LightGBM algorithm with the hyperparameters as mentioned in the Experimental Setup section.

The results associated with each of our approaches for malware prediction and evasion prediction are summarized in Table 6 for easier comparison. We also include the performance metrics of the baseline when tested against the mix of evasive and non-evasive validation sets.

Table 6. Results of Experiments 0, 1, 2 and 3.

	Task	Accuracy	TNR	FPR	FNR	TPR	F1-Score
Baseline	Malware Pred.	83.21%	97.95%	2.05%	25.72%	74.28%	84.64%
Approach 1	Malware Pred.	99.48%	99.48%	0.52%	0.51%	99.49%	99.59%
Approach 2	Malware Pred.	99.44%	99.45%	0.55%	0.57%	99.43%	99.55%
	Evasion Pred.	96.84%	97.97%	2.03%	4.31%	95.69%	96.76%
Approach 3	Malware Pred.	99.48%	99.48%	0.52%	0.51%	99.49%	99.59%
	Evasion Pred.	96.60%	98.21%	1.79%	5.06%	94.94%	96.49%

4.2. Testing Experiments

After building the three approaches, we assess how well they perform in the real world where they might receive data coming from a distribution different than the one initially trained on. Accordingly, we perform two test cases, one against a real dataset and one against an evasive dataset. Since evasive datasets are difficult to find, we generate our own dataset and then test our approaches against it.

4.2.1. Experiment 1: Testing on a Real Dataset

We test the proposed approaches against a dataset containing 110,844 samples that were collected by monitoring the network of a large university campus for six days [22]. Since we do not have information about the evasive aspect of these samples, we will only test our approaches regarding their ability to detect malware. The results are reported for the three approaches along with the baseline in Table 7 and Figure 5. We can see that all approaches have similar performance (accuracy and f1-score are almost the same and around 95%) and provide almost the same AUC (around 0.99). The same is true for the ROC curve as shown in Figure 5. This confirms that the approaches preserve good malware detection functionality when it comes to real data.

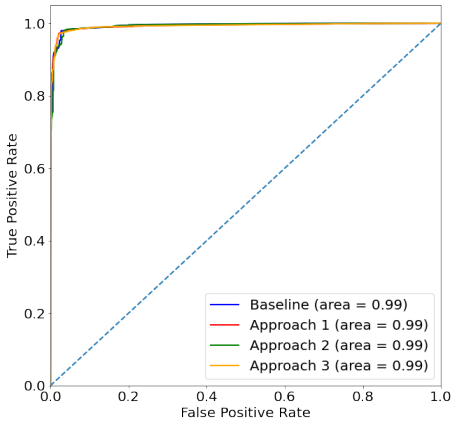

Figure 5. ROC curves for the baseline and the three approaches in the first test case.

Table 7. Classification metrics for the baseline and the three approaches in the first test case.

	Accuracy	TNR	FPR	FNR	TPR	F1-Score
Baseline	95.51%	99.26%	0.74%	9.02%	90.98%	94.84%
Approach 1	95.73%	98.90%	1.10%	8.09%	91.91%	95.13%
Approach 2	95.32%	98.84%	1.16%	8.93%	91.07%	94.64%
Approach 3	95.26%	98.94%	1.06%	9.19%	90.81%	94.55%

4.2.2. Experiment 2: Generating an Evasive Dataset

Since evasive datasets are rare, we generated our own evasive dataset, as described below, and we made it publicly available on GitHub for other researchers to use (it is

available at https://github.com/fouadtrad/Leveraging-Adversarial-Samples-for-Enhanced-Classification-of-Malicious-and-Evasive-PDF-Files, accessed on 4 February 2023).

- We feed all the evasive data collected to the baseline model. This classifier will correctly classify a portion of this data and will wrongly classify the other.
- We select the misclassified samples, and we consider them the most evasive samples.
- We train a Tabular Variational AutoEncoder (TVAE) model [23] on the most evasive dataset we obtained in the previous step. To do this, we rely on the Synthetic Data Vault (SDV) library [24]. The training happened for 500 epochs, with an embedding_dim of 256.
- We used this model to generate 500,000 samples of evasive data out of which 450,000 samples are malicious and 50,000 samples are benign.
- To verify that the samples provided have a real aspect, we used TensorFlow Data Validator to make sure that the generated samples adhere to the original evasive dataset schema.

At this stage, the evasive dataset is ready, and we can use it for testing our approaches.

4.2.3. Experiment 3: Testing on the Generated Evasive Dataset

The generated data are fed to the three systems and to the baseline. The results are shown in Table 8 and Figure 6. We can see that the baseline did not perform well against these samples (AUC = 0.13, accuracy = 9.69%, f1-score = 15.32). On the other hand, the proposed systems resulted in good outcomes (AUC 0.95–0.97, accuracy 91–92%, f1-score 95%). The little drop in performance compared to the validation sets is expected since the generated samples do not conform to the same distribution as the training sets, and this is an essential test case to assess how a system would behave with real scenarios. Since in this test case, all our samples are evasive, we wanted to assess the evasion detection capabilities of approaches 2 and 3. For approach 2, the evasion detection rate was 88.29% and in approach 3, the rate was 94.23%. In summary, and in terms of malware detection, the first and the third approach perform slightly better than the second one, having a superior AUC value. As for evasion detection, we can see that the model in approach 3 is performing better than the model in approach 2.

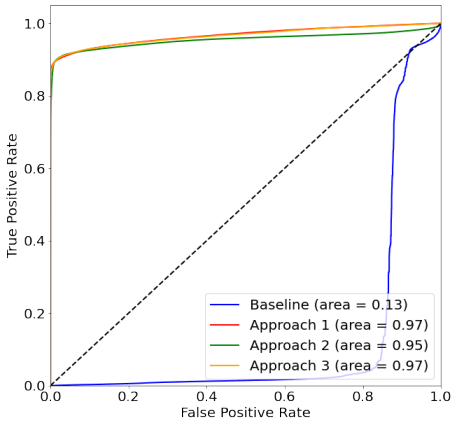

Figure 6. ROC curves for the baseline and the three approaches in the second test case.

Table 8. Results of the second test case.

	Accuracy	TNR	FPR	FNR	TPR	F1-Score
Baseline	9.69%	15.18%	84.82%	90.92%	9.08%	15.32%
Approach 1	92.37%	91.93%	8.07%	7.58%	92.42%	95.61%
Approach 2	92.36%	86.46%	13.54%	6.99%	93.01%	95.63%
Approach 3	92.49%	91.52%	8.48%	7.40%	92.60%	95.68%

5. Conclusions

In this paper, we leveraged adversarial samples that are usually designed to "fool" existing models and used them to build robust systems capable of detecting maliciousness and evasion associated with PDF files. We introduced three approaches that showed robustness against attacks, and two of them had the ability to detect evasion as well. We tested our systems and they achieved great performance. We also generated a dataset containing 500,000 evasive samples and made it available to the research community.

Despite the remarkable results we achieved, this study has some limitations, and the main one is the data availability. Although we were able to find datasets that have been used by previous studies, these datasets were just about CSV files that contain pre-extracted features of PDFs, and having the actual PDF files to perform better feature engineering was not an option in most of the cases. Thus, we were limited by the available set of features. Yet, despite this limitation, we were able to establish a proof of concept that can further be enhanced in future works. Another limitation is that although the proposed systems worked well in the tested scenarios, they might not perform well under some new emerging evasion attacks if they are significantly different than the ones considered during the training phase. In other words, just like any machine learning model, when a data drift is encountered, the model's performance is expected to drop. In our case, and to enhance the robustness of the proposed systems against all existing adversarial attacks, we incorporated samples related to all of them during the training phase. Moreover, when new attacks emerge and result in a distribution drift, the proposed approach will undergo adversarial retraining with these samples to keep the systems up to date. Moreover, if we have limited data when such attacks emerge, we can overcome this problem by generating data as discussed in the paper.

The encouraging results we obtained pave the way for an extension to supplement the previous works of researchers: we plan on performing better feature engineering along with using more complex model architectures while using adversarial learning, which is expected to further enhance the performance. In addition to our proposed adversarial retraining approach, we plan to investigate other adversarial robustness techniques to further enhance the resilience of PDF malware classifiers against evasion attacks. Moreover, we will investigate other methods to generate evasive data such as by performing real attacks against our systems, towards enhancing the robustness of existing malware detection systems. Another direction is to perform reverse engineering to generate real PDF files having the specifications of the evasive dataset we generated. This would allow researchers to use even more features that they can directly extract from typical PDF files.

Author Contributions: F.T.: Conceptualization, methodology, software, visualization, validation, data curation, writing—original draft preparation; A.H.: methodology, supervision, project administration; A.C.; writing—review and editing, supervision, project administration. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to acknowledge that this work has been supported by the Maroun Semaan Faculty of Engineering and Architecture (MSFEA) at the American University of Beirut (AUB), and the APC was funded by MSFEA.

Data Availability Statement: The data presented in this study is made publicly available on GitHub (https://github.com/fouadtrad/Leveraging-Adversarial-Samples-for-Enhanced-Classification-of-Malicious-and-Evasive-PDF-Files, accessed on 4 February 2023) for the research community.

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

References

1. Kaur, J.; Ramkumar, K.R. The recent trends in cyber security: A review. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 5766–5781. [CrossRef]
2. Li, Y.; Liu, Q. A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. *Energy Rep.* **2021**, *7*, 8176–8186. [CrossRef]
3. Aslan, O.A.; Samet, R. A Comprehensive Review on Malware Detection Approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]

4. Blonce, A.; Filiol, E.; Frayssignes, L. Portable Document Format (PDF) Security Analysis and Malware Threats. In Proceedings of the Europe BlackHat 2008 Conference, Amsterdam, The Netherlands, 24–28 March 2008; p. 20.
5. Fleury, N.; Dubrunquez, T.; Alouani, I. PDF-Malware: An Overview on Threats, Detection and Evasion Attacks. *arXiv* **2021**, arXiv:2107.12873.
6. Iwamoto, M.; Oshima, S.; Nakashima, T. A Study of Malicious PDF Detection Technique. In Proceedings of the 2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS), Fukuoka, Japan, 6–8 July 2016; pp. 197–203. [CrossRef]
7. Maiorca, D.; Biggio, B. Digital Investigation of PDF Files: Unveiling Traces of Embedded Malware. *IEEE Secur. Priv.* **2019**, *17*, 63–71. [CrossRef]
8. Torres, J.; De los Santos, S. Malicious PDF Documents Detection using Machine Learning Techniques—A Practical Approach with Cloud Computing Applications. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy, Funchal, Portugal, 22–24 January 2018*; SCITEPRESS—Science and Technology Publications: Funchal, Madeira, Portugal, 2018; pp. 337–344. [CrossRef]
9. Issakhani, M.; Victor, P.; Tekeoglu, A.; Lashkari, A. PDF Malware Detection based on Stacking Learning. In Proceedings of the 8th International Conference on Information Systems Security and Privacy, Online, 9–11 February 2022; pp. 562–570. [CrossRef]
10. Maiorca, D.; Biggio, B.; Giacinto, G. Towards Adversarial Malware Detection: Lessons Learned from PDF-based Attacks. *ACM Comput. Surv.* **2020**, *52*, 1–36. [CrossRef]
11. Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Šrndić, N.; Laskov, P.; Giacinto, G.; Roli, F. Evasion Attacks against Machine Learning at Test Time. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases, Prague, Czech Republic, 23–27 September 2013*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; pp. 387–402. ._25. [CrossRef]
12. Zhang, J. MLPdf: An Effective Machine Learning Based Approach for PDF Malware Detection. *arXiv* **2018**, arXiv:1808.06991.
13. Zhang, J. Machine Learning With Feature Selection Using Principal Component Analysis for Malware Detection: A Case Study. *arXiv* **2019**, arXiv:1902.03639.
14. Khorshidpour, Z.; Hashemi, S.; Hamzeh, A. Learning a Secure Classifier against Evasion Attack. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain, 12–15 December 2016; pp. 295–302. [CrossRef]
15. Li, Y.; Wang, X.; Shi, Z.; Zhang, R.; Xue, J.; Wang, Z. Boosting training for PDF malware classifier via active learning. *Int. J. Intell. Syst.* **2021**, *37*, 2803–2821. [CrossRef]
16. Xu, W.; Qi, Y.; Evans, D. Automatically Evading Classifiers. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 21–24 February 2016; p. 15. [CrossRef]
17. Maiorca, D.; Corona, I.; Giacinto, G. Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious PDF files detection. In Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, Hangzhou, China, 8–10 May 2013; pp. 119–130. [CrossRef]
18. Šrndić, N.; Laskov, P. Practical Evasion of a Learning-Based Classifier: A Case Study. In Proceedings of the 2014 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 18–21 May 2014; pp. 197–211. [CrossRef]
19. Carmony, C.; Zhang, M.; Hu, X.; Vasisht Bhaskar, A.; Yin, H. Extract Me If You Can: Abusing PDF Parsers in Malware Detectors. In Proceedings of the 2016 Network and Distributed System Security Symposium, San Diego, CA, USA, 21–24 February 2016. [CrossRef]
20. Mila. 16,800 Clean and 11,960 Malicious Files for Signature Testing and Research. Contagio Dataset. 2013. Available online: http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html (accessed on 18 April 2022)
21. VirusTotal. Virus Total Home Page. Available online: https://www.virustotal.com/gui/home/upload (accessed on 23 April 2022).
22. Smutz, C.; Stavrou, A. Malicious PDF detection using metadata and structural features. In Proceedings of the 28th Annual Computer Security Applications Conference—ACSAC '12, Orlando, FL, USA, 3–7 December 2012; p. 239. [CrossRef]
23. Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; Veeramachaneni, K. Modeling Tabular data using Conditional GAN. *arXiv* **2019**, arXiv:1907.00503.
24. Patki, N.; Wedge, R.; Veeramachaneni, K. The Synthetic Data Vault. In Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, Canada, 17–19 October 2016; pp. 399–410. [CrossRef]
25. *ISO:32000-1:2008*; Document Management—Portable Document Format—Part 1: PDF 1.7. ISO: Geneva, Switzerland, 2008. Available online: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/15/51502.html (accessed on 20 April 2022)
26. Tzermias, Z.; Sykiotakis, G.; Polychronakis, M.; Markatos, E.P. Combining static and dynamic analysis for the detection of malicious documents. In Proceedings of the Fourth European Workshop on System Security, Salzburg, Austria, 11 April 2011; pp. 1–6. [CrossRef]
27. Adobe. Adobe® PDF (Portable Document Format) 1.7 Reference; 2006.
28. Corona, I.; Maiorca, D.; Ariu, D.; Giacinto, G. Lux0R: Detection of Malicious PDF-embedded JavaScript code through Discriminant Analysis of API References. In Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, Scottsdale, AZ, USA, 7 November 2014; pp. 47–57. [CrossRef]

29. Munson, M.; Cross, J. *Deep PDF Parsing to Extract Features for Detecting Embedded Malware*; OSTI: Oak Ridge, TN, USA, 2011. [CrossRef]
30. Stevens, D. Malicious PDF Documents Explained. *IEEE Secur. Priv.* **2011**, *9*, 80–82. [CrossRef]
31. Croce, F.; Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv* **2020**, arXiv:2003.01690.
32. Brendel, W.; Rauber, J.; Bethge, M. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. *arXiv* **2018**, arXiv:1712.04248.
33. Brendel, W.; Rauber, J.; Kümmerer, M.; Ustyuzhaninov, I.; Bethge, M. Accurate, reliable and fast robustness evaluation. *arXiv* **2019**, arXiv:1907.01003.
34. Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. DeepFool: A simple and accurate method to fool deep neural networks. *arXiv* **2016**, arXiv:1511.04599.
35. Chen, P.Y.; Sharma, Y.; Zhang, H.; Yi, J.; Hsieh, C.J. EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. *arXiv* **2018**, arXiv:1709.04114.
36. Chen, J.; Jordan, M.I.; Wainwright, M.J. HopSkipJumpAttack: A Query-Efficient Decision-Based Attack. *arXiv* **2020**, arXiv:1904.02144.
37. Shafahi, A.; Najibi, M.; Ghiasi, M.A.; Xu, Z.; Dickerson, J.; Studer, C.; Davis, L.S.; Taylor, G.; Goldstein, T. Adversarial training for free! In *Proceedings of the Advances in Neural Information Processing Systems*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
38. Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 582–597. [CrossRef]
39. Xu, W.; Evans, D.; Qi, Y. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In Proceedings of the 2018 Network and Distributed System Security Symposium, Internet Society, San Diego, CA, USA, 18–21 February 2018. [CrossRef]
40. Papernot, N.; McDaniel, P.; Sinha, A.; Wellman, M.P. SoK: Security and Privacy in Machine Learning. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018; pp. 399–414. [CrossRef]
41. Papernot, N.; McDaniel, P.; Goodfellow, I. Transferability in Machine Learning: From Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv* **2016**, arXiv:1605.07277.
42. Demontis, A.; Melis, M.; Pintor, M.; Jagielski, M.; Biggio, B.; Oprea, A.; Nita-Rotaru, C.; Roli, F. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. *arXiv* **2019**, arXiv:1809.02861.
43. Yusirwan, S.; Prayudi, Y.; Riadi, I. Implementation of Malware Analysis using Static and Dynamic Analysis Method. *Int. J. Comput. Appl.* **2015**, *117*, 975–8887. [CrossRef]
44. Shafiq, M.Z.; Khayam, S.A.; Farooq, M. Embedded Malware Detection Using Markov n-Grams. In *Detection of Intrusions and Malware, and Vulnerability Assessment*; Zamboni, D., Ed.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5137, pp. 88–107. [CrossRef]
45. Smutz, C.; Stavrou, A. When a Tree Falls: Using Diversity in Ensemble Classifiers to Identify Evasion in Malware Detectors. In Proceedings of the 2016 Network and Distributed System Security Symposium, San Diego, CA, USA, 21–24 February 2016. [CrossRef]
46. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2015**, arXiv:1412.6572.
47. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. *arXiv* **2014**, arXiv:1409.4842.
48. 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 Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017*; Curran Associates, Inc.: Red Hook, NY, USA, 2017, Volume 30.
49. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.