Developing Robust Defense Mechanisms against Adversarial Evasion Attacks in Machine Learning-Based Malware Classifiers

Muhammad Zikria Tahir[1], Saba Firdous[1], Talha Ahsan[1], Muhammad Asad[1], Muhammad Ali Hasnaat[1], Ibn e Hassan[1], Abdul Haseeb Malik[2]
[1]Department of Computing Riphah International University Faisalabad, Pakistan
[2]Department of Computer Science, University of Peshawar 25120, Pakistan
*__Correspondence__: Talha.ahsen@riphahfsd.edu.pk

Citation | Tahir. M. Z, Firdous. S, Ahsan. T, Asad. M, Hasnaat. M. A, Hassan. I. E, Malik. A. H, "Developing Robust Defense Mechanisms against Adversarial Evasion Attacks in Machine Learning-Based Malware Classifiers", IJIST, Vol. 8 Issue. 10 pp 4089-4107, November 2026

Received | Oct 24,2026 **Revised |** Nov 06, 2026 **Accepted |** Nov 08, 2026 **Published |** Nov 10, 2026.

Machine learning (ML) classifiers are commonly employed in antivirus and endpoint detection and response (EDR) solutions to differentiate between malicious and legitimate software. But these classifiers can be defeated by an attacker who simply modifies the features of a malware file by a small amount without affecting the functionality of the file. This paper investigates this issue in the case of static Windows Portable Executable (PE) malware. We adopt the EMBER benchmark and a LightGBM classifier, and evaluate 2 attacks, FGSM and PGD, under white-box and black-box scenarios. The clean accuracy drops to 43.1% with PGD with a budget of 20 added features and to 31.4% with a budget of 50 added features without a defense. We investigate a layered defense, using adversarial training (AT) and feature squeezing (FS). With a budget of 20, AT+FS increased the robust accuracy to 91.2%, while reducing the evasion rate from 56.9% to 8.8%. It also reduces the black-box evasion rate from 31.4% to 6.1%. The cost is a 2.2 percentage-point drop in clean accuracy (97.8% to 95.6%) and a drop in F1-score from 0.976 to 0.954. An ablation study reveals that both AT and FS are helpful and that they are most beneficial when combined. All settings are reported so that others can replicate this study.

Keywords: Adversarial Machine Learning, Adversarial Training, Evasion Attacks, Feature Squeezing, Malware Detection, Robustness, EMBER Dataset

Introduction:

Background and Motivation:

Malware remains a top threat to computer systems, with many security products employing machine learning (ML) to detect it. Antivirus engines, intrusion detection and prevention systems (IDPS) and endpoint detection and response (EDR) platforms learn classifiers from Portable Executable (PE) files that include features like byte n-grams, import tables, section entropy, and header metadata. These classifiers achieve detection rates of over 97% for these types of samples on benchmark datasets like EMBER [1] and SOREL-20M [2] and have become a major component of modern cyber defense efforts [3][4].

But the use of ML based classifiers is facing a major problem: they are vulnerable to adversarial evasion attacks. In this kind of attack, the adversary tries to alter the sample with minor modifications in order to retain its functionality while causing the classifier to miss it and classify it as benign. If a sample is successful, it may evade the deployed security tools until it is detected via other means [5][4].

Malware classification is more difficult than image classification. An image can be attacked with freedom to modify the value of each pixel whereas a malware attack has to ensure that the file remains executable and malicious. With binary feature vectors, it implies that this attack problem is discrete, and mostly the attacker can only add features (0→1), but cannot freely remove them. The normal pipeline comprises of a PE file to feature vector conversion and classification by LightGBM as shown in Figure 1. An evasion attack is demonstrated in Figure 2, where the attacker modifies the features in order to fool the classifier. Figure 3 provides an overview of the proposed defense, which is based on adversarial training and feature squeezing.

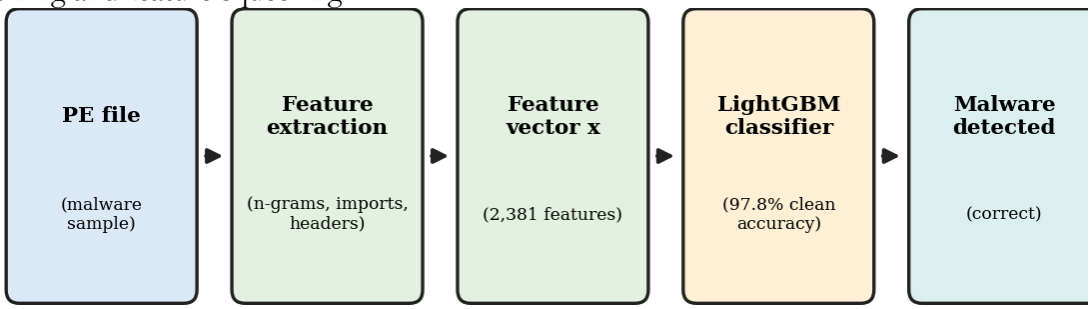


Figure 1. Normal malware detection without an attack: a PE file is converted to a feature vector and classified by LightGBM.

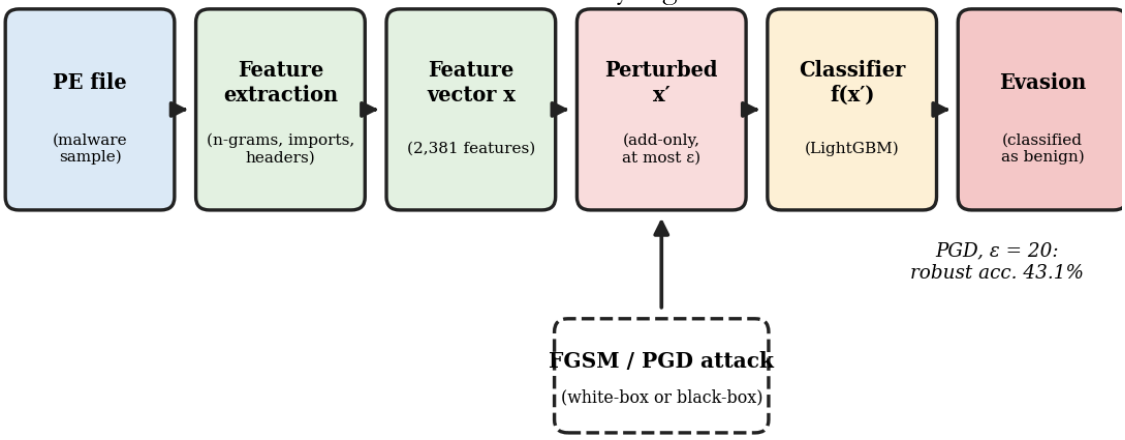


Figure 2. Adversarial evasion attack: FGSM or PGD adds a small number of features so that the classifier labels the malware sample as benign.

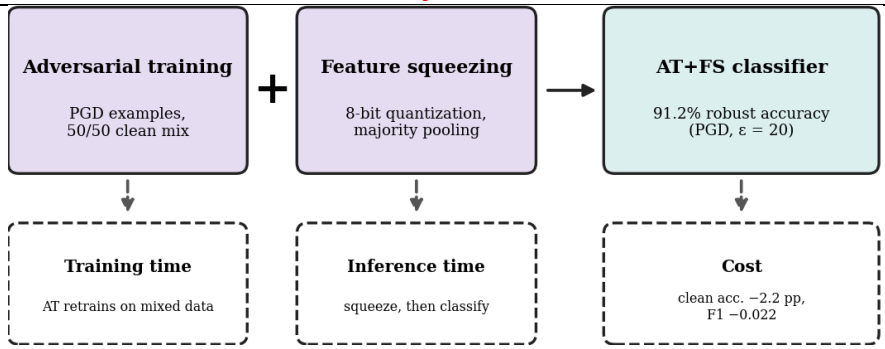


Figure 3. Proposed AT+FS defense: adversarial training is applied at training time and feature squeezing at inference time.

Recent Related Work (2022–2026):

Since 2022, research on this issue has increased rapidly. The initial works introduced the core concepts: FGSM [6] and PGD-based adversarial training [7] and the first attacks that target malware classifiers using discrete features [8] and reinforcement learning [9] and byte-level changes [10] respectively. This work has been extended in five directions in recent studies (2022–2026). First, new attacks have emerged, such as reinforcement-learning attacks and attacks in constrained space [11][12], attacks involving the generation of adversarial malware [13] and attacks based on memetic search [14] and attacks based on large language models (LLMs) [15][16]. Second, the adversarial training studies demonstrate that it is helpful, but that its effectiveness varies with the attack and the data [17][18][19][20]. Third, new defenses consist of certified or verifiable methods [21][22][23], randomized smoothing [24], byte masking [25], density-based training [26] and ensembles [27]. Fourth, evaluation practice was brought into doubt in recent research and more powerful testing is sought [28][29]. Fifth, there are recent surveys, taxonomies and benchmarks that organize the field [3][30][31][32]. The connections that these studies have with the present research are explained in the Literature Review, under "Recent Studies (2022–2026) and Relation to This Work.

Research Objectives:

The main aims of this study are:

To measure how much FGSM and PGD evasion attacks reduce the accuracy of an undefended LightGBM malware classifier on EMBER, under white-box and black-box threat models and with addition-only feature changes;

To design a layered defense, AT+FS, that combines adversarial training with feature squeezing;

To test whether AT+FS gives higher robust accuracy than adversarial training alone or feature squeezing alone, and to measure its cost in clean accuracy, F1-score, and training time; and

To describe the experiment in enough detail that other researchers can repeat it.

Research Gap and Novelty:

Previous research on malware classifiers have primarily focused on attacks, or one defense mechanism per study. Adversarial training has been primarily studied on its own [17][18][19] and feature squeezing was proposed for images [33] but has been studied little for PE feature vectors. Research also varies in data sets, cost and metrics making it difficult to compare the results [29][28]. In the studies surveyed in the Literature Review, we did not find any that test adversarial training + input-side feature squeezing, both white-box and black-box attacks, with a single set of budgets and metrics, on EMBER feature vectors.

This research is therefore novel in three ways. (1) It suggests a two-layer defence, squeezing the input features at the inference time of an adversarially-trained model according to a well-defined inference rule (Eq. (v)). (2) It adopts one evaluation protocol (same data split,

budget, metrics) for the baseline, AT, FS and AT+FS under white-box and black-box attacks. (3) It provides an ablation study and cost analysis (accuracy, F1-score and training time), demonstrating a balance between robustness and clean performance. This work remains in the feature space, as opposed to other works that work at the raw-binary or byte level [17][34]. It provides empirical robustness, rather than guarantees as do the certified methods [21] and [23].

Contributions:

The main contributions of this study are:

A threat model for PE malware classifiers involving addition-only, budget-limited feature changes under white-box and black-box access (See Threat Model).

An evaluation of FGSM and PGD evasion attacks on a LightGBM classifier trained on EMBER (see Attack Methods and Experimental Evaluation).

A layered AT+FS defense and an ablation study that separates the effect of each part (see Proposed Methodology and Experimental Evaluation).

A full description of the data, parameters, software, and hardware to support replication (see Experimental Evaluation).

Paper Organization:

The Literature Review presents a review of related work, incorporating findings from recent studies from 2022-2026. The Threat Model defines the attacker, its capabilities and the defender assumptions. The attack methods are described under Attack Methods. The methodology and AT+FS defense proposal are presented under Proposed Methodology. The Experimental Evaluation describes the experiments and reports the results. The results, accuracy-robustness trade-off, comparison with previous studies, practical implications and limitations are discussed in the Discussion. The paper ends with the Conclusion. Also remove the yellow highlight from this paragraph.

Literature Review:

Machine Learning-Based Malware Detection:

The field of machine learning for malware detection has shifted from hand-designed features to deep models. [35] demonstrated that simple binary programmable features can be used to detect malware without the need for manually designed features with a neural network. [36] proposed the raw-byte-reading convolutional network called MalConv. The EMBER [1] release enabled the community to have a large public benchmark with a fixed feature set (2,381 features per PE file) and a baseline using LightGBM [37] which also continues with the EMBER2024 release [31]. One reason for using LightGBM is that tree-based models have been shown to perform well on tabular data [38]. Yet, all these classifiers make the assumption that test inputs are similar to training data, and can be fooled by adversarial inputs [39][28].

Adversarial Machine Learning: Foundations:

Deep networks can be deceived by small changes in the input as demonstrated by Szegedy et al. [40]. In [6] Goodfellow et al. proposed a one-step attack FGSM and in [7] Madry et al. proposed a multi-step attack PGD, along with the adversarial training as a min–max defense. [41] demonstrated that many defenses can be broken by more powerful and adaptive attacks, and thus defenses need to be robust to attacks that know the defense. The NIST taxonomy [30] provides common names for these attacks and defenses and [42] demonstrates the usefulness of standardized evaluation.

Adversarial Evasion in Malware Classification:

[8] were one of the first to develop adversarial examples for malware classifiers by imposing discrete constraints on the features. [9] used reinforcement learning for black-box evasion of PE classifiers, and [11] extended this idea in MAB-Malware. Executables could be modified in real ways, as done by [10] with bytes and by [43][44] with real executables. Recent works use attack generators [13], memetic search [14] and evasion and poisoning as a pair [45]

as well as LLMs [15][16] and constructing adversarial malware datasets by family [46]. Surveys cover the field in detail [4][47][5][11]. Our attacks operate in feature space, which is less expensive than binary-level attacks. However, each modified feature vector does not necessarily correspond to a functional file, which is a limitation (see Limitations and Future Work).

Defense Mechanisms:

Adversarial training is the most widely studied defense [17][18][19][20][29]. Squeezing, introduced by [33], is a method which reduces precision of input and thus filters out small adversarial perturbations. Randomized and de-randomized smoothing [24][21], certified or verifiable robustness [22][48][23], byte masking [25] and binary-level changes [34] are other defenses; density-based training [26] and ensemble methods [27] are defenses for tabular data [49] that aim to reduce costs and utility loss. There are few studies comparing the layered defenses on PE feature vectors in terms of the accuracy–robustness trade-off [4][21]. This motivates the present work.

Recent Studies (2022–2026) and Relation to This Work:

Table 1 tabulates recent studies (2022-2026) most relevant to this study and the reason for their relevance. In brief, recent attack studies reveal evasion continues to become more robust, recent defense studies almost exclusively focus on a single method or on provable guarantees, and recent evaluation studies call for more conscientious testing. The current study falls in between these and provides a practical, empirical attack/defense (AT+FS) and evaluates it with the same protocol against both white-box and black-box attacks. The relation column indicates the relevance of each study to the present research.

Table 1. Recent Studies (2022–2026) and their Relation to This Work

Study	Main focus	Relation to this work
[11]	Reinforcement-learning framework for black-box adversarial malware	Basis for our black-box (transfer) setting; we use a surrogate model instead of query-based search.
[12]	Unified attack and defense in constrained feature space	Supports our use of feature-space attacks with validity constraints (addition-only, budget).
[28]	Pitfalls in ML for security evaluation	Guides our protocol: fixed splits, several seeds, and explicit threat model.
[17]	Adversarial training for raw-binary malware classifiers	Shows adversarial training is useful at the binary level; we study it at the feature level and add feature squeezing.
[18]	Feature-space Bayesian adversarial learning	A training-time defense in feature space; our AT is simpler, and we combine it with an input-side layer.
[22]	Verifiable learning for robust tree ensembles	Relevant because LightGBM is a tree ensemble; it offers verification, whereas AT+FS is empirical.
[19]	Effectiveness of adversarial training on malware classifiers	Directly supports AT; we measure what is gained by adding FS.
[21]	De-randomized smoothing with certified robustness for malware	Gives certified guarantees on raw binaries; our defense gives no guarantee but is easy to add to existing models.
[26]	Density-based strategy for performance, robustness, and sustainability	Another route to robustness; it could be combined with AT+FS.

[29]	Robustness evaluation of adversarial defenses in malware detection	Motivates our shared protocol and ablation for comparing defenses.
[16]	LLM-driven feature-level attacks on Android detectors	Shows that feature-level attacks are a realistic threat; our threat model is similar but for PE files.
[3]	Survey of datasets, techniques, and challenges for robust malware detection	Gives context for dataset choice and open challenges.
[50]	Co-evolution of malware and detectors (bilevel optimization)	Attacker and defender adapt to each other; we test a fixed set of attacks, so co-evolution is future work.
[23]	Certified malware detection with provable guarantees	Provable robustness versus our empirical robustness.
[27]	Explainable, robust ensemble framework for malware detection	An ensemble route to robustness; our base model is already a tree ensemble, and AT+FS is added on top.
[15]	LLMs used to evade CNN-based Windows malware detectors	LLM-guided attack on another model type; outside our scope (feature-space attacks on LightGBM).
[46]	Adversarial malware dataset by family and type	Suggests family-level evaluation, which we leave to future work.

Threat Model:

Target System and Attacker Goal:

The target system is the simple PE (Windows Portable Executable) Detector. A 2381-dimensional feature vector is used to represent each file, including byte level, structural and metadata features [1]. The behaviour of the file is not used as the detector does not execute the file.

The attacker's objective is evasion: to successfully get a known malware sample classified as benign, and not blocked. The attacker aims to make as few changes to the features as possible and the attacker needs the modified file to maintain its malicious behavior. The real name of the sample, e.g. malware, remains the same.

Attacker Knowledge and Access:

White-box: Attacker has access to the type of model, parameters, feature extraction pipeline, defense, etc. This is the maximum possible case, and provides an upper bound for the power of the attack. White-box attacks rely on the knowledge of the gradients, as described under Feature Space Representation.

Black-box: The attacker can only input and output prediction, but cannot see the model parameters and training data. Attacker trains a surrogate model on similar data, generates perturbations on the surrogate, and injects them to the target [11][12].

Attacker Capabilities and Limitations:

The attacker is limited by the following rules. (1) Addition only: a feature can only go from 0 to 1, but not from 1 back to 0, as removing content from a PE file might break it or remove its malicious activity. (2) Budget: the number of ε features to add to each sample is at most ε with $\varepsilon \in \{5, 10, 20, 50\}$ where the higher the value of ε the more powerful but costly the attacker. (3) Only feature space: feature vectors are directly modified by the attacker without building the modified executable files. This means that the experiments are simple and repeatable, but each altered vector is not necessarily realizable as an actual executable file.

(4) Test time only: Attacker does not have access to the training data and/or to the model after training.

Defender Assumptions:

The defender desires high accuracy for clean samples and perturbed samples and minimal false alarms for benign files. At test time, the defender is aware of the evasion attacks that might occur, but cannot determine which attack or budget will be applied. The defender can retrain the model and is allowed to use pre-processing at inference, however, is not given access to the attacker's surrogate model.

Out-of-Scope Threats:

This work does not consider data poisoning, model extraction, binary-level evasion, and dynamic-analysis evasion, which are left to future research [34][43].

Attack Methods:

Feature Space Representation:

Each PE sample is a binary vector $x \in \{0,1\}^d$ with d = 2,381, where x_j = 1 means that feature j is present. EMBER also supports features like entropy and byte histograms, which are count and real valued and are converted to presence/absence for discrete attacks (value > 0 gives 1). The classifier is $f: \{0,1\}^d \rightarrow [0,1]$ which is the probability that x is malware. The attacker is trying to find x' such that $f(x') < 0.5$, where $x' \geq x$ componentwise (i.e. only addition) and $\|x' - x\|_0 \leq \varepsilon$ (at most ε features change). The evasion condition is given in Eq. (i).

$$f(x') < 0.5 \quad \text{(i)}$$

At the most basic level, Eq. (i) is saying that the attack is successful if the malware score of the classifier is less than 0.5. The two rules above ensure that the change is small and is addition-only. LightGBM is a tree ensemble and thus has no gradient; the gradients $\nabla_x L$ used in the attacks below are taken from a differentiable stand-in for the ensemble model, which only ranks the features, and the final prediction is always made by the real LightGBM model.

Fast Gradient Sign Method (FGSM):

FGSM has 1 step of gradient. In the case of images, the step is $x' = x + \varepsilon \cdot \text{sign}(\nabla_x L)$ which shifts each pixel by a small amount in the direction that increases the loss (Eq. (ii)). The feature vector in binary malware cannot be modified by small continuous amounts, so we use the discrete version. (1) Compute the loss gradient for every feature. (2) Only keep the features that have a current value 0 as only these can be added. (3) List them in order of the magnitude of the positive gradient. (4) Change the top ε to 1 and keep all the other ε's the same. In this case, ε is thus a positive integer, not a small real number.

$$x' = x + \epsilon . sign(\nabla x L(\theta, x, y)) \quad \text{(ii)}$$

Put another way, Eq. (ii) states "change the input so that the classifier is most incorrect." The sign (·) function only preserves direction (up/down) and not size of the gradient, and ε bounds how much can change. In the case of binary features, "up" refers to changing a feature from 0 to 1.

Projected Gradient Descent (PGD):

PGD is a repetition of the FGSM concept. We start from a random valid starting point and use T = 50 rounds with a step size of α = 1 feature. At each step, PGD performs a gradient descent and then back-projection to the set of allowed steps S to remove any that force an illegal step (only addition, and only at most ε features can be added). PGD is stronger than FGSM because it iteratively updates the perturbation in each round and therefore used as the primary worst case test [7]. The update rule is given in Eq. (iii).

$$x_{t+1} = \Pi_{x+s}(x_t + \propto . sign(\nabla_{x_t} L(\theta, x_t, y))) \quad \text{(iii)}$$

In words, Eq. (iii) states: "Take the current sample x_t, take one step that increases the loss, and then step back to the closest allowed sample. Here α is the size of the step and Π is the projection onto the set of allowed values S.

Black-Box Attack via Transferability:

Black-box - Attacker has no access to the parameters of the target. We train a surrogate Random Forest (500 estimators) with the same training data, used to generate PGD perturbations on the surrogate and then test the perturbations on the LightGBM target and defended versions. This is a transferability attack [11][12]. The results are shown in Table 6 and Figure 9.

Attack Evaluation Protocol:

All attacks are tested on the 200,000 EMBER test samples (100,000 malicious and 100,000 benign). The attacks are applied to malicious samples, which are the samples an attacker aims to conceal. The following metrics are used. Clean accuracy is the percentage of well classified clean test samples. Robust accuracy (RA) is the share of the attacked malicious samples that are still detected. Evasion rate (ER) is 100% − RA. Accuracy drop is the difference between the clean accuracy and RA. F1 score is calculated on clean samples. All changes in performance are given in percentage points (pp). For reference, RA for the clean accuracy is set for the no-attack row of Table 3.

Attack budgets: The budget ε is the number of features added to a sample, with $\varepsilon \in \{5, 10, 20, 50\}$, which is 0.2% to 2.1% of the 2,381 features. The results for ablation, black-box and overhead are obtained with the primary setting of $\varepsilon = 20$, while $\varepsilon = 5$ and $\varepsilon = 10$ are low budgets and $\varepsilon = 50$ is the strongest budget. In all others, the $L\infty$ bound on continuous, scaled features is set at 0.3 and 0.2, respectively. The value is not the number of features and it's not a comparison with the other budgets.

All experiments are done with five different independent random seeds, and results are presented as mean ± SD [1][28]. The results are presented in Tables 3–7 and Figures. 6–9.

Proposed Methodology:

The data collection and evaluation process in the study is presented in Figure 4. The pipeline comprises six stages: (1) data collection (with the EMBER benchmark of PE feature vectors); (2) data preprocessing (with a class-balanced split of the training, validation and test sets and a binary feature view); (3) model training (LightGBM baseline and a surrogate Random Forest); (4) attacks (FGSM and PGD white-box and black-box attacks); (5) defense (adversarial training and feature squeezing and their combination); and (6) evaluation (clean accuracy, F1-score, robust accuracy, evasion rate, ablation study and overhead over five random seeds). It explains in a more detailed manner the actions of the attacks and the defense on the classifier in Figure 5.

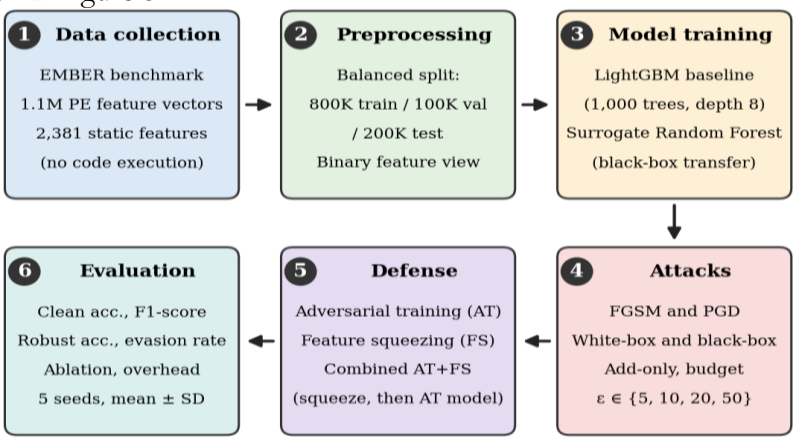


Figure 4. Study flow of this work: data collection, preprocessing, model training, attacks, defense, and evaluation.

Defense Design Principles:

The framework is based on three principles. (1) Adaptive-adversary awareness: The defense must be tested against a user who is aware of it [28][41]. (2) Little loss in accuracy: the increase in robustness should not be at the expense of clean-sample accuracy to a large extent. (3) Modularity: Every component can be tested and implemented separately.

Defense Component 1: Adversarial Training:

Adversarial training (AT): Train the model to resist the attack by presenting it with attacked samples during training. The augmented training set consists of clean samples and PGD generated adversarial examples (T = 50, α = 1) in a 50/50 ratio which is identified by cross-validation method. The same hyperparameters as the baseline [19][17][18] are then used to retrain the classifier for 1000 iterations of boosting. The min–max problem in Eq. (iv) is the training objective.

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}[\max_{\delta \in s} L(\theta, x+\delta, y)] \quad \text{(iv)}$$

The inner maximization in Eq. (iv) searches for the change δ, within the set S, that most damages the model; the outer minimization trains the parameters θ so that the model performs well on the worst change.

Defense Component 2: Feature Squeezing:

In feature squeezing (FS), The precision of the input features is reduced before classification, thus allowing small adversarial changes to be "squeezed out" [33]. Two squeezers are used. (1) Continuous features are encoded to 8 bits. Binary features in the same group are combined using maximum pooling within clusters of related features: a few 1s added in a group will not affect the group value unless the majority of features in the group are set. Instead of x' giving to classifier, giving x_sq = Q(x') When the inputs f(x') and f(x_sq) differ significantly, FS can also indicate suspicious inputs.

Combined Layered Defense:

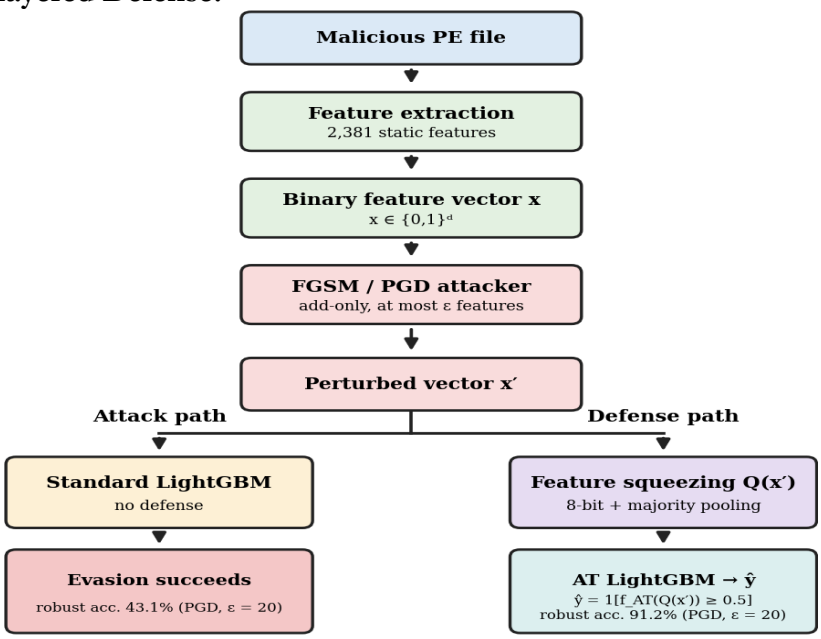


Figure 5. Architecture of the attack path and the defense path for a PE malware classifier. The attacker perturbs x to x'. Without a defense, the standard model is fooled. With AT+FS, the input is squeezed and then classified by the adversarially trained model.

The key differences are that AT works at different points and fixes different weaknesses. AT alters the model: learns a less sensitive decision boundary with added features. FS does not exhibit changes in input: it filters out a portion of the perturbation prior to

reaching the model. Even if it is used alone, there are perturbations that are not learned during the training on FS that fool it and viceverca. In the combined AT+FS defence, AT is only performed once at training time and FS is only applied at the inference time, which means there is no extra training cost.

The inference rule is $\hat{y} = 1[f_AT(Q(x')) \geq 0.5]$ (Eq. (v)), where the input is first squeezed, then the adversarially trained model can classify it. This layered design is intended to give robustness against both gradient-based white-box attacks and transfer-based black-box attacks; This is evaluated in the Experimental Evaluation section. The architecture is displayed in Figure 5.

$$\hat{y} = 1[f_{AT}Q(x')) \geq 0.5] \text{(v)}$$

Ablation Study Design:

The four configurations tested are: (1) the undefended LightGBM baseline; (2) adversarially trained classifier (AT only); (3) feature squeezing with the standard classifier (FS only); and (4) the proposed full layered defense (AT+FS). All are evaluated against FGSM/PGD. Results for all budgets are summarized in Table 4 and Figure 6, and those for the ablation at $\varepsilon = 20$ are summarized in Table 5 and Figure 7.

Experimental Evaluation:

Dataset:

The benchmark employed is EMBER [1] which describes each PE file with 2,381 features in total, consisting of 8 different feature groups. The EMBER collection reported to be used here has a total of approximately 2.3 million feature vectors. We use a class-balanced, labelled subset of 1.1 million samples for three reasons: (1) adversarial training and the PGD attack are expensive methods, and each setting is repeated for 5 seeds and multiple attack budgets are evaluated; (2) a class-balanced subset is used to avoid class-imbalance effects on accuracy and robust accuracy; and (3) only labelled samples can be used for computing accuracy or for supervised training. The subset consists of 800,000 training, 100,000 validation and 200,000 test samples which are randomly selected with a fixed seed such that no sample occurs in more than one split. We did not use the newer EMBER2024 benchmark [31] since our features are according to the 2,381-feature EMBER format.

Table 2. Class-Balanced Ember Subset used in this Study

Split	Total Samples	Malicious	Benign
Training	800,000	400,000	400,000
Validation	100,000	50,000	50,000
Test	200,000	100,000	100,000
Total	1,100,000	550,000	550,000

Classifier Configuration and Software:

The default values of the LightGBM [37] based baseline classifier are: n_estimators = 1,000, max_depth = 8, learning_rate = 0.05, min_child_samples = 50, other parameters are set to the defaults. On the clean test set, the baseline reaches 97.8% accuracy and an F1-score of 0.976, which is in line with known EMBER baselines [1]. A Random Forest with 500 estimators is used as a surrogate for the black-box experiments.

Experimental Setup and Reproducibility:

Software and hardware: For experiments, Python 3.10 was used with scikit-learn 1.3, LightGBM 4.0 and the Adversarial Robustness Toolbox (ART) v1.15 [51]. The hardware consisted of a workstation with a 16-core CPU, 64 GB of RAM, and Ubuntu 22.04.

Parameter settings: To use PGD we set T = 50, step size $\alpha = 1$ feature, and use a random start. Adversarial training is with a 50/50 ratio of clean and adversarial samples, and retraining is for 1,000 iterations. Feature squeezing uses 8-bit quantization and majority pooling. The

attack budgets are ε ∈ {5, 10, 20, 50}. The seeds for splitting, model training and random starts are five random values (0–4).

Statistical tests: The mean ± SD for the five seeds are reported. The SD for each value in Tables 3–7 was below 0.6 pp, and so omitted from the tables. The differences among configurations were tested using a paired t-test on the five seeds at 0.05 significance level, with AT+FS found to be significantly different from all other configurations (at $p < 0.01$).

Availability: The EMBER data are freely available [1]. The code and the configuration files will be published along with the definitive version of this paper.

Baseline Attack Results:

The results for the undefended classifier under FGSM and PGD are shown in Table 3. As ε increases, both attacks decrease robust accuracy and PGD is superior at all sizes of budgets. For instance, when ε = 20 the robust accuracy is 54.7% for the FGSM attack and 43.1% for the PGD attack. At ε = 50, PGD brings robust accuracy down to 31.4%, a drop of 66.4 pp from the clean value of 97.8%. This is an indication that a defense is required.

Table 3. Undefended Classifier Performance Under FGSM and PGD Attacks

Attack	ε	Clean Acc.	Robust Acc.	Evasion Rate	Acc. Drop (pp)
None	—	97.8%	97.8%	2.2%	0.0
FGSM	5	97.8%	81.3%	18.7%	16.5
FGSM	10	97.8%	69.4%	30.6%	28.4
FGSM	20	97.8%	54.7%	45.3%	43.1
FGSM	50	97.8%	41.2%	58.8%	56.6
PGD	5	97.8%	74.6%	25.4%	23.2
PGD	10	97.8%	58.3%	41.7%	39.5
PGD	20	97.8%	43.1%	56.9%	54.7
PGD	50	97.8%	31.4%	68.6%	66.4

Defense Evaluation Results:

As seen in Figure 6 and Table 4, all four configurations achieve measurable robust accuracy under PGD. AT+FS achieves the top robust accuracy at all budgets and gracefully suffers a drop from 94.1% at ε = 5 to 87.9% at ε = 50, with the baseline dropping from 74.6% to 31.4%. At any given budget (AT+FS, AT only, FS only and the baseline), the ranking does not change and the difference between AT+FS and AT only increases from 6.7 pp at ε=5 to 17.2 pp at ε=50.

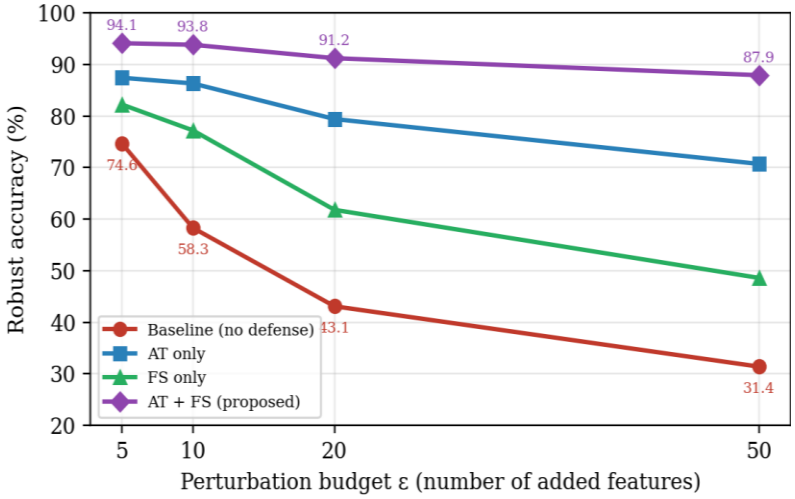


Figure 6. Robust accuracy versus perturbation budget ε under PGD for all defense configurations. The values are listed in Table 4. AT+FS stays at or above 87.9% for all budgets.

Table 4. Robust accuracy (%) under PGD for each defense and budget ε (data for figure. 6)

Configuration	ε = 5	ε = 10	ε = 20	ε = 50
Baseline (no defense)	74.6	58.3	43.1	31.4
AT only	87.4	86.3	79.4	70.7
FS only	82.2	77.2	61.8	48.6
AT + FS (proposed)	94.1	93.8	91.2	87.9

As shown in Table 5, the AT+FS configuration achieves the highest robust accuracy at ε = 20 (91.2%) and the evasion rate is the smallest (8.8%). The clean accuracy drops to 95.6% (2.2 pp below the baseline) and the F1-score drops from 0.976 to 0.954. AT alone raises robust accuracy from 43.1% to 79.4% (+36.3 pp), and FS alone raises it to 61.8% (+18.7 pp). The composite accuracy is +48.1 pp, which is greater than either component, but is smaller than the sum of the individual improvements (+55.0 pp) as robust accuracy cannot exceed 100%. Its effects are thus complementary, rather than just additive. Statistically significant differences between AT+FS and the other configurations (paired t-test, $p < 0.01$). Comparison of clean and robust accuracy is shown in Figure 7.

Table 5. Ablation study of the defense configurations under pgd (ε = 20)

Configuration	Clean Acc.	Robust Acc.	Evasion Rate	F1-Score
Baseline (no defense)	97.8%	43.1%	56.9%	0.976
AT only	95.9%	79.4%	20.6%	0.957
FS only	97.1%	61.8%	38.2%	0.969
AT + FS (proposed)	95.6%	91.2%	8.8%	0.954

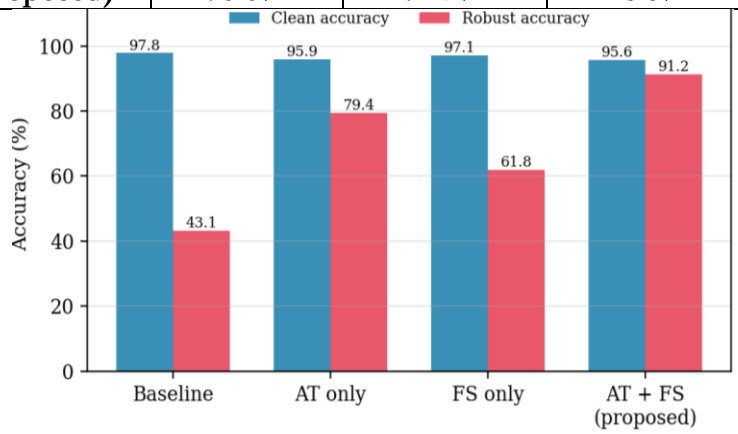


Figure 7. Clean and robust accuracy for each defense configuration under PGD (ε = 20). AT+FS has the highest robust accuracy at a clean-accuracy cost of 2.2 pp.

Evasion Rate Heatmap:



Figure 8. Evasion rate (%) for each attack and defense configuration. Green means low evasion (good defense) and red means high evasion (poor defense).

Evasion rates for FGSM and PGD at ε = 5, 20, and 50 are omitted in the heatmap in Figure 8, as they are shown in Table 4 and Figure 6 (ε = 10 is shown in Table 4 and Figure. 6 and omitted for space). AT+FS evasion rates are the lowest rates in all six columns ranging from 4.1% to 12.1%, with the baseline being from 18.7% to 68.6%. The corresponding PGD value at ε = 50 is 12.1% which is related to the robust accuracy of 87.9% shown in Figure. 6.

Black-Box Versus White-Box Transferability:

Table 6 gives PGD evasion rates (ε = 20) in the white-box and black-box settings. For all configurations, the black-box evasion rate is lower than the white-box evasion rate since perturbations on the surrogate transfer only partially to the target. For the baseline it falls from 56.9% to 31.4%, and for AT+FS from 8.8% to 6.1%. AT+FS is the lowest evasion rate in both modes. The black-box attack is weaker and low numbers for the black-box attack should not be interpreted as stronger evidence than numbers for the white-box attack. Figure. 9 shows a comparison of the two settings.

Table 6. White-Box and Black-Box Evasion Rates (PGD, ε = 20)

Configuration	Evasion Rate (Black-Box)	Evasion Rate (White-Box)
Baseline	31.4%	56.9%
AT only	12.3%	20.6%
FS only	22.7%	38.2%
AT + FS (proposed)	6.1%	8.8%

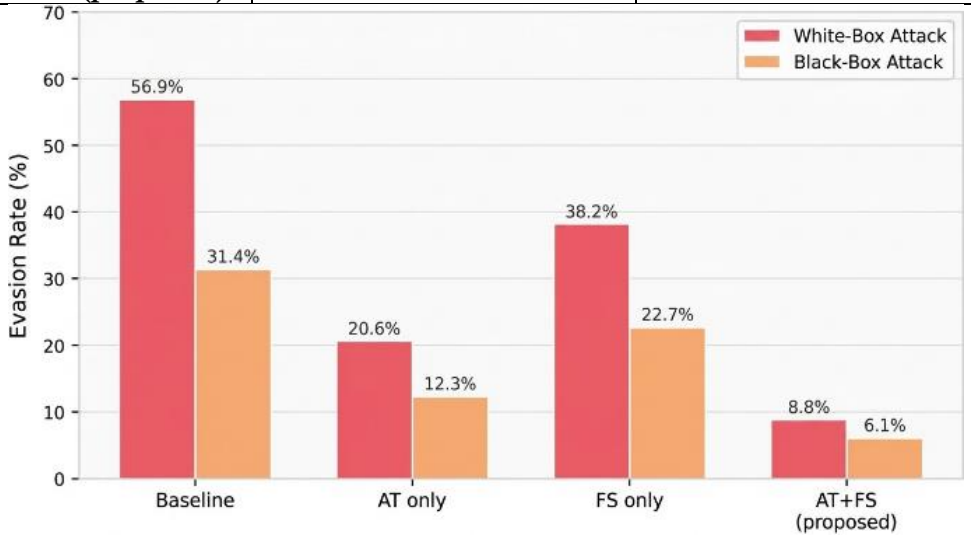


Figure 9. White-box versus black-box evasion rate for each defense configuration (PGD, ε = 20). Black-box attacks are weaker because adversarial examples transfer only partly.

Computational Overhead:

Adversarial training takes approximately 4.2x longer than regular training. The effect of feature squeezing is negligible at inference (< 2 ms per sample). So the training overhead of AT+FS is approximately 4.3×, which is quite acceptable in view of the enhanced robustness [19][17].

Supplementary FGSM Stress Test (ε = 0.1):

We performed a secondary FGSM stress test on the same 200,000 test samples to evaluate a higher and different stress level. Here, the bound on the continuous, min–max scaled feature values (each feature can change by at most 0.1) is taken as ε = 0.1, and the changed values are then converted back to the binary view described under Feature Space Representation. It is not a feature count like the budgets in Tables 3–6, and thus the results are not comparable with those tables, but are reported separately in Table 7. In this case, the undefended robust accuracy is 34.5%, which is less than that of FGSM with ε = 50 in Table 3.

Table 7. Supplementary FGSM stress test (ε = 0.1 on continuous scaled features)

Model	Attack	Clean Acc. (%)	Robust Acc. (%)	Evasion Rate (%)
Standard LightGBM	None	97.8	97.8	2.2
Standard LightGBM	FGSM (ε = 0.1)	97.8	34.5	65.5
Defended LightGBM (AT+FS)	FGSM (ε = 0.1)	95.6	91.8	8.2

As can be seen from Table 7, the undefended classifier is highly susceptible with a significant decline in its robust accuracy from 97.8% to 34.5%. This implies that 65.5% of the malware samples used to attack it are not detected. The defended AT+FS model maintains a strong accuracy of 91.8%, thereby decreasing the evasion rate from 65.5% to 8.2%, a 87.5% relative improvement. Its clean accuracy is 95.6%, which is 2.2 pp below the baseline and is the same as in Table 5. The accuracy – robustness trade-off is that there is a small reduction in clean accuracy in exchange for a large increase in robustness [52].

Discussion:

Interpretation of Results:

The results give three main findings. The first is that the undefended LightGBM classifier is highly susceptible: PGD brings down the robust accuracy to 43.1% at ε = 20 and to 31.4% at ε = 50 (Table 3). Second, AT+FS achieves the best robustness: 91.2% robust accuracy at ε = 20 (Table 5) and at least 87.9% at all the budget levels tested (Table 4) with a clean-accuracy loss of 2.2 pp. Third, AT and FS are complementary, meaning that both are helpful when used alone (79.4% and 61.8% at ε = 20) and that the combination is more effective than either alone. The results are valid for the data, attacks and budgets employed in this study.

The Accuracy–Robustness Trade-Off:

As demonstrated by [52] clean accuracy and robustness can be in opposition. Our results follow this pattern, but the cost is small: clean accuracy falls by 1.9 pp for AT only (97.8% to 95.9%), 0.7 pp for FS only (to 97.1%), and 2.2 pp for AT+FS (to 95.6%), while robust accuracy at ε = 20 rises by 48.1 pp for AT+FS. If the chance of a missed malware sample is expensive, as in the case of EDR on high-value systems, it is OK to lose 2.2 pp. For practitioners where false alarms are expensive, FS only may be preferable, as there isn't much loss of clean accuracy but less gain. Clean accuracy and F1-score are reported but not the false-positive rate separately – this should be verified in the target environment.

Comparison with Previous Studies:

Our findings are in a similar direction to previous studies. Adversarial training experiments for malware classifiers have typically indicated that the adversarial training leads to the improvement of the robustness against the used attacks [17][18][19]; our AT-only results confirm the same trend (robust accuracy from 43.1% to 79.4% at ε = 20). However, our FS-only results achieve a lesser improvement, which aligns with the idea that pre-processing defenses perform poorly without FS [41]. But certified and verifiable defenses [21][22][23] offer guarantees which our empirical results cannot offer, typically requiring special models or training. The black-box results align with the transferability studies [11][12] and indicate that perturbations can only be partially transferred so that black-box evasion is less than white-box evasion. This is not a direct numerical comparison with these studies as the datasets, feature sets, attacks and budgets vary [29][28] and thus we compare trends rather than numbers.

Practical Implications:

For organizations using traditional classifiers, adversarial evasion is a true threat, particularly in high-value environments. The AT+FS framework can be implemented using publicly available libraries (ART, scikit-learn, and LightGBM), and requires no special hardware. Feature squeezing can be done at a low cost – it takes less than 2ms per sample to

run at inference, and does not require retraining of the model. Adversarial training requires training around 4.2 times longer, which is not problematic for the frequently retrained models, as is anyway the case for malware which change over time [53].

Limitations and Future Work:

This study has the following limits. (1) Only use of feature space: perturbed feature vectors are not necessarily translated to working executables; some perturbed feature vectors may not be translated to real files [43][34]. (2) One dataset and one main model – other datasets like BODMAS [54] and EMBER2024 [31] and other model types are future work. (3) Limited attacks: Only FGSM, PGD, and one transfer attack are tested, but not stronger attacks such as squeezing step based, search based [14] or LLM guided [15]. (4) Empirical robustness only: AT+FS comes without a guarantee [21][23]. (5) Fixed data split: They tested their models on one data split and not on later data, thus fail to measure drift over time [53]. (6) Explanations: We do not study why the added features work, which explainable ML could help to show [55]. These issues will be addressed in future work.

Conclusion:

This work examined the adversarial robustness of the ML-based malware classifiers on EMBER feature vectors. The undefended LightGBM classifier is also easy to evade; when $\varepsilon = 20$, its strong accuracy drops from 97.8% to 43.1% and when $\varepsilon = 50$, it drops to 31.4%. The proposed AT+FS defense achieves the high accuracy of 91.2% at the setting of $\varepsilon = 20$ and maintains the accuracy of at least 87.9% within the range of $\varepsilon = 20$ to $\varepsilon = 50$. It reduces the white-box evasion rate from 56.9% to 8.8% and black-box evasion rate from 31.4% to 6.1%. This comes at a price of 2.2 pp lower clean accuracy, a 0.022 reduction in F1, roughly 4.3 times slower training, and less than 2ms per sample for additional inference. The ablation demonstrates that, AT and FS complement each other.

The usefulness of these results is that a simple defense based on public tools can take away much of the benefit of feature-space evasion for the attacks considered here. Based on the results, we suggest that practitioners (1) evaluate deployed classifiers on at least a PGD-type attack and a variety of budgets, and not just on clean data; (2) consider adding feature squeezing as a first layer; (3) consider using adversarial training if retraining is an option; and (4) report clean accuracy and robust accuracy alongside the attack budget.

The study has limitations: it operates in the feature space, it is based on one dataset and one primary model, and it is based on two gradient-based attacks and one transfer attack, and it provides empirical (but not certified) robustness (See Limitations and Future Work). Future studies should be done on binary level attacks, adaptive attacks, additional datasets and models, and on certified or explainable extensions. With the increasing integration of ML in the security infrastructure, operational robustness to adversarial attacks is not just an academic matter.

References:

[1] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," Apr. 2018, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/1804.04637

[2] R. Harang and E. M. Rudd, "SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection," *CEUR Workshop Proc.*, vol. 3095, pp. 13–21, Dec. 2020, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2012.07634

[3] T.-T. Huynh, D.-T. Huynh, and V.-Q. Trinh, "Toward Robust Malware Detection: A Survey of Datasets, Techniques, and Practical Challenges," *Eng. Technol. Appl. Sci. Res.*, vol. 16, no. 3, pp. 35064–35070, Jun. 2026, doi: 10.48084/ETASR.17500.

[4] D. Li, Q. Li, Y. Ye, and S. Xu, "Arms Race in Adversarial Malware Detection: A Survey," *ACM Comput. Surv.*, vol. 55, no. 1, Aug. 2021, doi: 10.1145/3484491.

[5] X. Ling *et al.*, "Adversarial attacks against Windows PE malware detection: A survey

of the state-of-the-art," *Comput. Secur.*, vol. 128, May 2023, doi: 10.1016/j.cose.2023.103134.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, 2015, Accessed: Apr. 22, 2026. [Online]. Available: https://www.researchgate.net/publication/403853041_Adversarial_Attacks_on_AI-Based_Intrusion_Detection_Systems

[7] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, Jun. 2017, Accessed: Sep. 23, 2026. [Online]. Available: https://arxiv.org/pdf/1706.06083

[8] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10493 LNCS, pp. 62–79, 2017, doi: 10.1007/978-3-319-66399-9_4/FIGURES/5.

[9] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth, "Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning," Jan. 2018, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/1801.08917

[10] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, and J. Keshet, "Deceiving End-to-End Deep Learning Malware Detectors using Adversarial Examples," Feb. 2018, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/1802.04528

[11] W. Song, X. Li, S. Afroz, D. Garg, D. Kuznetsov, and H. Yin, "MAB-Malware: A Reinforcement Learning Framework for Attacking Static Malware Classifiers," Mar. 2020, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2003.03100

[12] T. Simonetto, S. Dyrmishi, S. Ghamizi, M. Cordy, and Y. Le Traon, "A Unified Framework for Adversarial Attack and Defense in Constrained Feature Space," *IJCAI Int. Jt. Conf. Artif. Intell.*, pp. 1313–1319, Dec. 2021, doi: 10.24963/ijcai.2022/183.

[13] M. Kozák and M. Jureček, "Combining Generators of Adversarial Malware Examples to Increase Evasion Rate," *Proc. Int. Conf. Secur. Cryptogr.*, vol. 1, pp. 778–786, Apr. 2023, doi: 10.5220/0012127700003555.

[14] K. Ghanem, Z. Kherbache, and O. Ourdighi, "Enhancing Adversarial Examples for Evading Malware Detection Systems: A Memetic Algorithm Approach," *Int. J. Comput. Netw. Inf. Secur.*, vol. 17, no. 1, pp. 1–16, Feb. 2025, doi: 10.5815/IJCNIS.2025.01.01.

[15] M. M. Arifin, B. Abrew, D. Hubbs, and J. H. Yeh, "Adversarial Assembly: Leveraging LLMs to Evade CNN-Based Windows Malware Detection," *Lect. Notes Comput. Sci.*, vol. 16465 LNCS, pp. 305–318, 2026, doi: 10.1007/978-981-95-9846-5_26/SAVE-RESEARCH.

[16] T. Lan and F. Naït-Abdesselam, "LLM-Driven Feature-Level Adversarial Attacks on Android Malware Detectors," Dec. 2025, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2512.21404

[17] L. Breiman *et al.*, *Adversarial Training for {Raw-Binary} Malware Classifiers*, vol. 45, no. 1. in Request for Comments, vol. 45. IEEE, 2023. Accessed: Nov. 05, 2026. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/shen-meng

[18] B. G. Doan *et al.*, "Feature-Space Bayesian Adversarial Learning Improved Malware Detector Robustness," *Proc. 37th AAAI Conf. Artif. Intell. AAAI 2023*, vol. 37, pp. 14783–14791, Jan. 2023, doi: 10.1609/aaai.v37i12.26727.

[19] H. Bostani, J. Cortellazzi, D. Arp, F. Pierazzi, V. Moonsamy, and L. Cavallaro, "On the Effectiveness of Adversarial Training on Malware Classifiers," Dec. 2024, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2412.18218

[20] S. Dyrmishi, S. Ghamizi, T. Simonetto, Y. Le Traon, and M. Cordy, "On the Empirical Effectiveness of Unrealistic Adversarial Hardening Against Realistic Adversarial Attacks," *Proc. - IEEE Symp. Secur. Priv.*, vol. 2023-May, pp. 1384–1400, 2023, doi: 10.1109/SP46215.2023.10179316.

[21] S. Saha, W. Wang, Y. Kaya, S. Feizi, and T. Dumitras, "DRSM: De-Randomized Smoothing on Malware Classifier Providing Certified Robustness," *12th Int. Conf. Learn. Represent. ICLR 2024*, Mar. 2023, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2303.13372

[22] S. Calzavara, L. Cazzaro, G. E. Pibiri, and N. Prezza, "Verifiable Learning for Robust Tree Ensembles," *CCS 2023 - Proc. 2023 ACM SIGSAC Conf. Comput. Commun. Secur.*, vol. 1, pp. 1850–1864, May 2023, doi: 10.1145/3576915.3623100.

[23] N. Giri, A. K. A., S. Nicolazzo, A. Nocera, and V. P, "Towards Certified Malware Detection: Provable Guarantees Against Evasion Attacks," Apr. 2026, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2604.20495

[24] D. Gibert, G. Zizzo, and Q. Le, "Towards a Practical Defense against Adversarial Attacks on Deep Learning-based Malware Detectors via Randomized Smoothing," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 14399 LNCS, pp. 683–699, Aug. 2023, doi: 10.1007/978-3-031-54129-2_40.

[25] D. Gibert and F. Manyà, "ByteShield: Adversarially Robust End-to-End Malware Detection through Byte Masking," Dec. 2025, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2512.09883

[26] "Density Boosts Everything: A One-stop Strategy for Improving Performance, Robustness, and Sustainability of Malware Detectors - NDSS Symposium." Accessed: Nov. 05, 2026. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/density-boosts-everything-a-one-stop-strategy-for-improving-performance-robustness-and-sustainability-of-malware-detectors/

[27] A. Al Siam, M. Alazab, A. Obeidat, N. Faruqui, and I. Mahmud, "An explainable and robust ensemble-based framework for malware detection," *Peer-to-Peer Netw. Appl. 2026 193*, vol. 19, no. 3, pp. 81-, Apr. 2026, doi: 10.1007/S12083-026-02231-9.

[28] D. Arp *et al.*, "Dos and Don'ts of Machine Learning in Computer Security," *Proc. 31st USENIX Secur. Symp. Secur. 2022*, pp. 3971–3988, Oct. 2020, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2010.09470

[29] M. Jafari and A. Shameli-Sendi, "Evaluating the robustness of adversarial defenses in malware detection systems," *Comput. Electr. Eng.*, vol. 130, Feb. 2026, doi: 10.1016/j.compeleceng.2025.110845.

[30] A. Vassilev, A. Oprea, A. Fordyce, H. Anderson, X. Davies, and M. Hamin, "Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations," Mar. 2025, doi: 10.6028/NIST.AI.100-2E2025.

[31] R. J. Joyce *et al.*, "EMBER2024 -- A Benchmark Dataset for Holistic Evaluation of Malware Classifiers," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 2, pp. 5516–5526, Jun. 2025, doi: 10.1145/3711896.3737431.

[32] P. Mahesh, D. Sai, L. Amasala, T. Sree Bhimavarapu, and G. C. Garikipati, "A Malware Classification Survey on Adversarial Attacks and Defences," Dec. 2023, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2312.09636

[33] W. Xu, D. Evans, and Y. Qi, "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks," *25th Annu. Netw. Distrib. Syst. Secur. Symp. NDSS 2018*, Dec. 2017, doi: 10.14722/ndss.2018.23198.

[34] A. Abusnaina, Y. Wang, S. Arora, K. Wang, M. Christodorescu, and D. Mohaisen, "Burning the Adversarial Bridges: Robust Windows Malware Detection Against Binary-level Mutations," *Proc. ACM Conf.*, vol. 1, Oct. 2023, doi: 10.NNNN/NNNNNNN.NNNNNNN.

[35] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," *2015 10th Int. Conf. Malicious Unwanted Software, MALWARE 2015*, pp. 11–20, Feb. 2016, doi: 10.1109/MALWARE.2015.7413680.

[36] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas, "Malware Detection by Eating a Whole EXE," Oct. 2017, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/1710.09435

[37] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," *31st Conf. Neural Inf. Process. Syst.*, 2017, [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[38] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on tabular data?," Jul. 2022, Accessed: Sep. 10, 2026. [Online]. Available: https://arxiv.org/pdf/2207.08815

[39] A. McCarthy, E. Ghadafi, P. Andriotis, and P. Legg, "Functionality-Preserving Adversarial Machine Learning for Robust Classification in Cybersecurity and Intrusion Detection Domains: A Survey," *J. Cybersecurity Priv. 2022, Vol. 2, Pages 154-190*, vol. 2, no. 1, pp. 154–190, Mar. 2022, doi: 10.3390/JCP2010010.

[40] C. Szegedy *et al.*, "Intriguing properties of neural networks," *2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc.*, Dec. 2013, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/1312.6199

[41] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," *Proc. - IEEE Symp. Secur. Priv.*, pp. 39–57, Aug. 2016, doi: 10.1109/SP.2017.49.

[42] F. Croce *et al.*, "RobustBench: a standardized adversarial robustness benchmark," *Adv. Neural Inf. Process. Syst.*, Oct. 2020, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2010.09670

[43] K. Lucas, M. Sharif, L. Bauer, M. K. Reiter, and S. Shintre, "Malware Makeover: Breaking ML-based Static Analysis by Modifying Executable Bytes," *ASIA CCS 2021 - Proc. 2021 ACM Asia Conf. Comput. Commun. Secur.*, pp. 744–758, Oct. 2021, doi: 10.1145/3433210.3453086.

[44] L. Demetrio, S. E. Coull, B. Biggio, G. Lagorio, A. Armando, and F. Roli, "Adversarial EXEmples: A Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection," *ACM Trans. Priv. Secur.*, vol. 24, no. 4, Mar. 2021, doi: 10.1145/3473039.

[45] M. Kozák and M. Jureček, "Effectiveness of Adversarial Benign and Malware Examples in Evasion and Poisoning Attacks," *Mach. Learn. Deep Learn. AI Cybersecurity*, pp. 267–290, Jan. 2025, doi: 10.1007/978-3-031-83157-7_10.

[46] D. Košťál and M. Jureček, "Building an Adversarial Malware Dataset by Family and Type: Generation, Evasion, and Poisoning Evaluation," May 2026, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2605.25937

[47] K. Aryal, M. Gupta, M. Abdelsalam, P. Kunwar, and B. Thuraisingham, "A Survey on Adversarial Attacks for Malware Analysis," *IEEE Access*, vol. 13, pp. 428–459, 2025, doi: 10.1109/ACCESS.2024.3519524.

[48] Y. Chen, S. Wang, Y. Qin, X. Liao, S. Jana, and D. Wagner, "Learning Security Classifiers with Verified Global Robustness Properties," *Proc. ACM Conf. Comput.*

Commun. Secur., pp. 477–494, Dec. 2021, doi: 10.1145/3460120.3484776.

[49] K. Kireev, B. Kulynych, and C. Troncoso, "Adversarial Robustness for Tabular Data through Cost and Utility Awareness," *30th Annu. Netw. Distrib. Syst. Secur. Symp. NDSS 2023*, Aug. 2022, doi: 10.14722/ndss.2023.24924.

[50] O. Jurečková, M. Jureček, M. Kozák, and R. Lórencz, "Adversarial Co-Evolution of Malware and Detection Models: A Bilevel Optimization Perspective," *Proc. Int. Conf. Secur. Cryptogr.*, pp. 938–947, Apr. 2026, doi: 10.5220/0015191500004103.

[51] M.-I. Nicolae *et al.*, "Adversarial Robustness Toolbox v1.0.0," Jul. 2018, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/1807.01069

[52] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness May Be at Odds with Accuracy," *7th Int. Conf. Learn. Represent. ICLR 2019*, May 2018, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/1805.12152

[53] Y. Chen, Z. Ding, and D. Wagner, "Continuous Learning for Android Malware Detection," Jun. 2023, Accessed: Nov. 05, 2026. [Online]. Available: https://arxiv.org/pdf/2302.04332v2

[54] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh, and G. Wang, "BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware," *Proc. - 2021 IEEE Symp. Secur. Priv. Work. SPW 2021*, pp. 78–84, May 2021, doi: 10.1109/SPW53761.2021.00020.

[55] A. Nadeem et al, "SoK: Explainable Machine Learning for Computer Security Applications," *2023 IEEE 8th Eur. Symp. Secur. Priv. (EuroS&P), Delft, Netherlands*, pp. 221–240, 2023, doi: 10.1109/EuroSP57164.2023.00022.



Copyright © by authors and 50Sea. This work is licensed under Creative Commons Attribution 4.0 International License.