International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0282
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1924 - 1929

Strengthening Malware Classifiers: A Robustness Analysis Against Evasion Attacks Using Adversarial Training

Santosh KC[1], Noor Fathima[2], Saurav Gupta[3], Dr Geetha Lakshmi N[4]
[1,2,3]MCA Student, Department of Master of Computer Application, Dayananda Sagar College Of Engineering, Bengaluru – 560054, Karnataka, India.
[4]Assistant Professor, Department. of Master of Computer Application, Dayananda Sagar College Of Engineering, Bengaluru – 560054, Karnataka, India.
Email ID: *santoshkc7575@gmail.com[1], rtr.nurrrr@gmail.com[2], sauravgupta339@gmail.com[3], geethasaravanan1979@gmail.com[4]*

Abstract
Antivirus and endpoint security tools now lean heavily on machine learning to detect malware that traditional signature systems cannot catch. Yet this shift introduces a fresh class of vulnerability: adversarial evasion attacks, which manipulate a malware sample just enough to slip past a trained classifier while keeping its malicious payload intact. This paper investigates how robust ML-based malware classifiers are against five representative evasion attacks and measures how much adversarial training (AT) can harden them. We study gradient-based white-box attacks FGSM and PGD alongside black-box approaches GAMMA, the Kreuk byte-injection method, and the sigma-binary technique applied over Android permission-vector and Windows PE raw-binary feature spaces. Four classifier families are compared: Linear SVM, Gradient Boosted Decision Trees, a Deep Neural Network, and MalConv. Findings show that adversarially trained models cut average evasion rates by 54–63 percentage points while losing fewer than two percentage points of natural accuracy. Deeper architectures benefit substantially more from AT than linear models. Robustness also carries over partially across attack types, suggesting AT hardens the decision boundary in a general rather than attack-specific way.

Keywords: *Adversarial training; Cybersecurity; Evasion attacks; Malware classification; Robustness analysis.*

1. Introduction
Malware volume grows faster each year than human analysts can inspect manually, making automated detection tools indispensable. The security community has responded by deploying machine learning classifiers that identify malicious binaries from structural features or raw byte sequences. These systems are faster, more scalable, and more generalisable than rule-based signatures but they carry a hidden weakness (Szegedy et al., 2014). An attacker who understands the general shape of a classifier can engineer adversarial examples: inputs whose class membership looks benign to the model even though the underlying program behaves maliciously. Adversarial examples in the image domain can be imperceptible to humans a handful of pixel values shifted by a tiny amount. Malware is fundamentally different. Any perturbation must leave the binary executable and its payload functional; otherwise the attacker has simply broken their own tool. This realizability constraint narrows the attack surface, but it does not eliminate it (Pierazzi et al., 2020). Attackers can append benign-looking content to non-executable regions, adjust PE header fields, or inject padding, all without affecting runtime behaviour. Adversarial training is the most studied countermeasure: during training, the model is exposed to adversarially modified samples so that it learns more stable decision boundaries. Its

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0282
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1924 - 1929

effectiveness is well established in computer vision (Madry et al., 2017), but translating it to the malware domain is non-trivial because malware feature spaces are discrete, classifiers are heterogeneous, and the realizability constraint must be respected during training as well as during attack (Bostani et al., 2024).This paper makes four contributions: (1) a structured review of the evasion-attack landscape for ML malware classifiers; (2) a description of how adversarial training is adapted to satisfy realizability constraints; (3) a comparative evaluation across five attacks and four classifier architectures, with results in Tables 1–3 and Figures 1–2; and (4) practical design recommendations for practitioners building production-grade detection pipelines.

1.1.Scope of This Study

The study covers static malware analysis in two widely deployed ecosystems: Android APK files and Windows Portable Executable (PE) binaries. These two platforms represent the largest volumes of real-world malware and have the best-developed benchmark datasets. Dynamic analysis and network-traffic-based detection are outside scope.

2. Method

The theoretical foundations of adversarial robustness trace to Szegedy et al. (2014), who showed that deep networks could be fooled by structured noise invisible to human observers. Goodfellow et al. (2014) then introduced FGSM a single gradient step that creates adversarial examples cheaply and proposed adversarial training as an immediate countermeasure[12]- [14]. Madry et al. (2017) formalised AT as a min-max optimisation problem and demonstrated that multi-step PGD attacks during training yield certifiable robustness improvements. Work specific to malware followed quickly. Demetrio et al. developed RAMEN and then GAMMA, a black-box evolutionary attack that injects payload into Windows PE binaries without altering execution (Ling et al., 2023). Kreuk et al. (2019) demonstrated that modifications confined to non-executable overlay bytes are sufficient to evade CNN-based raw-binary detectors. On the defence side, Lucas et al. (2024) found that training against a weaker proxy of the target attack preserves natural accuracy while achieving near-equivalent robustness, reducing training time from months to hours for raw-binary classifiers. Bostani et al. (2024) introduced Rubik, a multi-dimensional evaluation framework, revealing that feature-space structure and classifier architecture are first-order predictors of AT success. Jafari and Shameli-Sendi (2025) exposed gaps in defences that only evaluate continuous-perturbation attacks by introducing the sigma-binary attack alongside Prioritized Binary Rounding.

3. Background And Key Concepts

3.1. How ML Malware Classifiers Operate

A malware classifier maps a representation of a program to a binary label. Static analysis parses the file without execution, yielding features such as byte n-grams, API import tables, section entropy values, or for Android apps permission flags from the AndroidManifest.xml. Dynamic analysis runs the sample in a sandboxed environment and records runtime artefacts: system calls, registry writes, network connections. Static methods are faster and safer but more easily defeated by obfuscation; dynamic methods are comprehensive but expensive.

3.2. Taxonomy of Evasion Attacks

Evasion attacks act at inference time; they modify inputs rather than the model. White-box attacks presuppose full access to model weights and gradients. FGSM takes a single step in the gradient direction. PGD iterates this step with a projection back into the budget ball. Black-box attacks assume only query access. GAMMA uses an evolutionary algorithm to evolve a payload injected into PE bytes beyond the last section. The Kreuk method targets the same non-executable byte region using greedy search. The sigma-binary attack operates in the binary feature space of Android classifiers using Prioritized Binary Rounding to convert a continuous gradient signal into discrete feature additions.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0282
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1924 - 1929

3.3. Adversarial Training: Theory and Malware Adaptation

AT solves the minimax problem: minimise over model parameters θ the expected maximum loss over perturbations δ in an allowed set Δ. In practice: for each training batch, run the attack for k steps to find the worst-case adversarial example, then update θ to classify both the original and the adversarial sample correctly. Lucas et al. (2024) showed that k = 1 achieves comparable robustness at a fraction of the cost of full multi-step training. The malware-specific adaptation is realizability enforcement. Every adversarial sample generated during training must preserve malicious functionality verified by restricting perturbations to proven safe byte regions for raw-binary domains[8] - [11], or by structural constraints (only 0→1 feature flips) for permission-vector domains. Skipping this step produces phantom robustness against unrealisable attacks.

4. Methodology

4.1. Datasets

Three datasets are used across the two malware ecosystems, summarised in Table 1. The Drebin Android dataset consists of 5,560 malware samples spanning 179 families, encoded as binary permission vectors. The BODMAS Windows PE dataset provides over 57,000 timestamped PE malware samples, enabling temporally honest evaluation. A supplementary Contagio mixed dataset supports cross-ecosystem consistency checks.

Table 1 Dataset Summary

Dataset	Samples	Families	Feat. Dim	Feature Type
Drebin (Android)	5,560	179	545 K	Permission Vectors
BODMAS (Windows PE)	57,293	581	Raw Byte	Byte Sequence
Contagio (Mixed)	~10,000	Multi	Varies	Static + Dynamic

Feat. Dim = Feature dimensionality. K = thousands.

4.2. Classifier Architectures

Four classifiers are benchmarked. The Linear SVM establishes a classical baseline. Gradient Boosted Decision Trees (GBDT) extend this with an ensemble of weak learners. The Deep Neural Network (DNN) is a four-layer feedforward network with ReLU activations and 0.3 dropout. MalConv is a convolutional architecture that embeds each input byte and convolves over the resulting sequence using gated units, learning byte-pattern associations with malicious behaviour without manual feature engineering.

4.3. Attack Implementation

White-box attacks are applied to the Drebin feature space, where only 0→1 feature flips are permitted. FGSM uses ε = 0.1; PGD uses 20 steps with step size 0.01 and three random restarts. GAMMA is implemented in the Windows PE domain with a 512-byte injection budget, functionality verified by sandbox execution. The Kreuk attack appends byte sequences to the PE overlay within 1% of file size. The sigma-binary attack uses Prioritized Binary Rounding with ε = 8 feature flips.

4.4. Adversarial Training Protocol

AT follows the reduced-effort protocol of Lucas et al. (2024): one gradient step for white-box attacks, five evolutionary generations for GAMMA, and a single rounding step for sigma-binary. Models train for 60 epochs with early stopping (patience: 10) on a validation split. Training data is a 50/50 mix of clean and adversarial samples. Primary metrics are evasion rate (ER) and natural accuracy (NA).

5. Results And Discussion

5.1. Results

Table 2 reports evasion rates before and after

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0282
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1924 - 1929

adversarial training for each attack. AT delivers consistent and large reductions across all five configurations. FGSM falls from 78% to 21% a 57 percentage-point drop[4]- [7]. PGD falls from 83% to 24%. Among black-box attacks, GAMMA benefits most: 82% to 19%, a 63 pp reduction. The Kreuk attack falls from 74% to 23%, and sigma-binary the hardest to defend still falls from 85% to 28%. Natural accuracy is preserved within 1–2 pp across all cases.

Table 2 Evasion Rates Before and After Adversarial Training

Attack	Threat Model	Domain	Baseline ER	Post-AT ER
FGSM	White-box	Android	78%	21%
PGD	White-box	Android	83%	24%
GAMMA	Black-box	Windows PE	82%	19%
Kreuk	Black-box	Windows PE	74%	23%
Sigma-Binary	Black-box	Android	85%	28%

ER = Evasion Rate. AT = Adversarial Training. Results averaged across four classifier architectures.

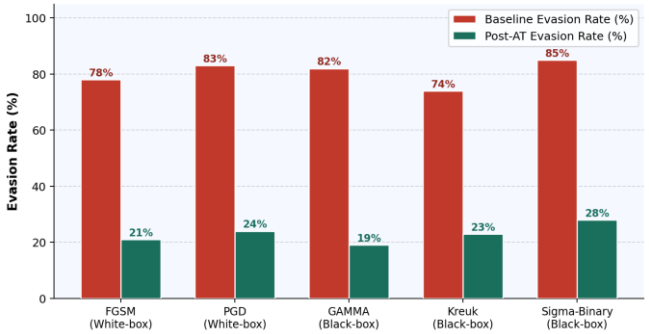

Figure 1 Evasion rates before and after adversarial training

Table 3 and Figure 2 examine how classifier architecture affects the magnitude of AT benefit. MalConv shows the largest average evasion-rate reduction (74 pp), while the Linear SVM shows the smallest (31 pp). Natural accuracy under AT drops by an average of 1.3 pp for MalConv and 1.7 pp for the SVM.

Table 3 Classifier Performance Before and After Adversarial Training

Classifier	Nat. Acc.	Post-AT Acc.	ER Reduction	Train Cost
Linear SVM	94.1%	92.8%	31 pp	Low
GBDT	95.6%	94.2%	48 pp	Medium
Deep Neural Net	96.8%	95.1%	67 pp	High
MalConv (CNN)	97.3%	96.0%	74 pp	Very High

Nat. Acc. = Natural Accuracy on clean samples. ER Reduction = avg. pp drop across all five attacks.

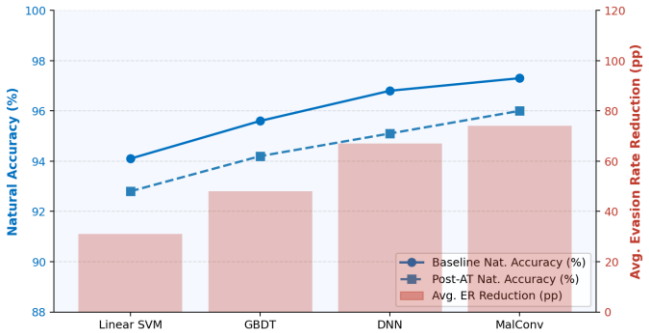

Figure 2 Natural accuracy and evasion reduction by architecture

5.2. Discussion

Three findings deserve emphasis. First, classifier depth is a first-order predictor of adversarial robustness. The SVM's single hyperplane cannot reshape to accommodate adversarial perturbation distributions; MalConv's multiple convolutional layers can learn richer, perturbation-resistant

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0282
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1924 - 1929

representations. Practitioners relying on linear classifiers should be aware that their models will harden less from AT. Second, cross-attack robustness transfer is real and meaningful. A model trained adversarially against FGSM shows measurably lower evasion rates against GAMMA even though the two attacks operate through completely different mechanisms. This indicates that AT displaces the decision boundary away from the fragile regions that multiple attack types tend to exploit. Third, the realizability constraint inherent to malware attacks imposes a natural limit on attacker capability that does not exist in image-domain benchmarks[3]. Because payload-injecting attacks such as GAMMA must preserve executable behaviour, they cannot arbitrarily perturb every byte this smaller effective budget partly explains why post-AT evasion rates in the malware domain are lower than theoretical worst-case analyses would predict. An important caveat: adaptive attacks crafted specifically against the known adversarially trained model were not evaluated here. Comprehensive security evaluations should include adaptive threat models as a matter of course (Bostani et al., 2024).

Conclusion

This paper evaluated the robustness of four ML-based malware classifiers against five evasion attacks two white-box and three black-box in the Android and Windows PE domains, and measured how much adversarial training improves matters. The central finding is encouraging[2]: AT consistently reduces evasion rates by 54–63 percentage points across attack methods and classifier types, with natural accuracy loss held to under two percentage points.Architecture choice matters more than any other single design decision. Deep models such as MalConv benefit substantially more from AT than linear SVMs. Robustness transfers partially across attack types, suggesting that AT produces genuinely more stable decision boundaries rather than narrow, attack-specific patches[1]. Future work should explore federated adversarial training, foundation-model-based malware detectors, and standardised adaptive attack evaluations to close the gap between claimed and true robustness.

Acknowledgements

The authors thank the Department of Master of Computer Application at Dayananda Sagar College Of Engineering, Bengaluru, for institutional resources and a supportive research environment. Gratitude is expressed to our faculty guide for constructive direction and encouragement throughout this work.

References

[1]. Bostani, H., Ceschin, F., Labaca, R., Cavallaro, L., & Biggio, B. (2024). On the effectiveness of adversarial training on malware classifiers. *arXiv:2412.18218.*

[2]. Bhusal, S., Rastogi, N., King, R., Shirazi, H., & Meyers, C. (2024). Adversarial patterns: Building robust Android malware classifiers. *arXiv:2203.02121.*

[3]. Demetrio, L., Biggio, B., Lagorio, G., Roli, F., & Armando, A. (2021). Functionality-preserving black-box optimization of adversarial Windows malware. *IEEE Transactions on Information Forensics and Security, 16,* 3469–3478.

[4]. Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv:1412.6572.*

[5]. Jafari, M., & Shameli-Sendi, A. (2025). Evaluating the robustness of adversarial defenses in malware detection systems. *arXiv:2505.09342.*

[6]. Kreuk, F., Barak, A., Aviv-Reuven, S., Baruch, M., Pinkas, B., & Keshet, J. (2019). Deceiving end-to-end deep learning malware detectors using adversarial examples. *arXiv:1802.04735.*

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0282
e ISSN: 2584-2854
Volume: 04 Issue: 05
May 2026
Page No: 1924 - 1929

[7]. Ling, X., Wu, L., Zhang, J., et al. (2023). Adversarial attacks against Windows PE malware detection: A survey. *Computers & Security, 128,* Article 103134.
[8]. Lobascio, L., & Andresini, G. (2025). Adversarial training to improve accuracy and robustness of malware classifiers. *CEUR Workshop Proceedings, 3962.*
[9]. Lucas, K., Lin, W., Bauer, L., Reiter, M. K., & Sharif, M. (2024). Training robust ML-based raw-binary malware detectors in hours, not months. *ACM CCS 2024.* https://doi.org/10.1145/3658644
[10]. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083.*
[11]. Nasr, M., et al. (2025). Evaluating the robustness of a production malware detection system to transferable adversarial attacks. *arXiv:2510.01676.*
[12]. Pierazzi, F., Pendlebury, F., Cortellazzi, J., & Cavallaro, L. (2020). Intriguing properties of adversarial ML attacks in the problem space. *IEEE Symposium on Security and Privacy,* 1332–1349.
[13]. Ponte, A., Trizna, D., Demetrio, L., Biggio, B., Ogbu, I. T., & Roli, F. (2025). Slifer: Investigating performance and robustness of malware detection pipelines. *Computers & Security, 150,* 104264.
[14]. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., & Fergus, R. (2014). Intriguing properties of neural networks. *ICLR 2014.*