Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Leakage-Free Evaluation Framework for Behavioral Malware Detection: Temporal Sensitivity, Dual Pooling, and Verified Explainability

NHM Hassan Imam Chowdhury
nhm.hassan@outlook.com
Bangladesh University of Professionals
Md. Ezharul Islam
Jahangirnagar University
Md. Sunjid Hasan
Bangladesh University of Professionals
Abdullah Al Mehedi
Bangladesh University of Professionals

Research Article

Keywords: Malware Detection, Behavioral Analysis, Leakage-Free Evaluation, Bidirectional GRU, Dual Pooling, Explainable AI, Adversarial Robustness, Temporal Sensitivity

Posted Date: June 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9641205/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Leakage-Free Evaluation Framework for Behavioral Malware Detection: Temporal Sensitivity, Dual Pooling, and Verified Explainability

NHM Hassan Imam Chowdhury[1]*, Md. Ezharul Islam[2], MD. Sunjid Hasan[3], Abdullah Al Mehedi[4]

[1, 3, 4] Department of ICT, Bangladesh University of Professionals, Dhaka, Bangladesh

[2] Department of CSE, Jahangirnagar University, Dhaka, Bangladesh

*Corresponding author: NHM Hassan Imam Chowdhury
Email: nhm.hassan@outlook.com
ORCID: https://orcid.org/0009-0007-2913-9611

[2]Email: ezharul.islam@juniv.edu, ORCID: https://orcid.org/0000-0001-8651-2848

[3]Email: sunjid.info@gmail.com

[4]Email: md.abdullah.mehedi@gmail.com

Abstract

Signature-based malware detectors are inherently vulnerable to polymorphic, metamorphic, and obfuscated variants, motivating behavioral approaches that model runtime process-state dynamics. However, deep learning studies in this domain commonly suffer from three persistent methodological issues: process-level data leakage, lack of temporal window sensitivity analysis, and insufficiently validated explainability claims.This study introduces a leakage-free evaluation framework that directly addresses these deficiencies. On a VirusTotal behavioral dataset of 100,000 millisecond-resolution Linux process observations from 100 temporally ordered process hashes, we employ hash-level GroupShuffleSplit and five-fold GroupKFold cross-validation to enforce strict process-level separation. We quantify that process-level leakage inflates accuracy by 10.06 percentage points, a significant bias in the existing literature. Leakage-free classical baselines establish a true performance floor of 61.34% (Random Forest).The proposed Bi-GRU+DualPool architecture integrates a Bidirectional GRU with parallel Global Average and Global Max Pooling to effectively capture both sustained and transient behavioral patterns. Under rigorous leakage-free five-fold cross-validation, it achieves 74.87 ± 8.56% accuracy and AUC = 0.787, outperforming the classical baseline by 13.53 percentage points. Ablation studies across temporal windows $W \in \{20, 50, 100\}$ demonstrate that dual pooling, rather than Multi-Head Attention, drives performance at short behavioral timescales.Feature perturbability classification bounds the realistic adversarial attack surface. Post-retraining recovery reaches 99.14% (FGSM) and 98.87% (PGD). SHAP KernelExplainer identifies ctx_switch_ratio as the primary discriminator with 4/5 cross-subset stability. The framework's methodological portability is validated on the UNSW-NB15 benchmark.This work establishes rigorous, leakage-aware evaluation standards and practical architectural insights for future behavioral malware detection research.

Keywords: Malware Detection; Behavioral Analysis; Leakage-Free Evaluation; Bidirectional GRU; Dual Pooling; Explainable AI; Adversarial Robustness; Temporal Sensitivity

1. Introduction

Malicious software continues to impose escalating costs on global digital infrastructure. Security telemetry reports more than 560,000 new samples identified daily, with cumulative totals exceeding one billion catalogued programs [1, 2], and the global economic burden of cybercrime is projected to reach 10.5 trillion USD annually by 2030 [3]. Signature-based anti-virus systems fail systematically against polymorphic, metamorphic, and obfuscated variants that deliberately alter static code representations [4, 5]. This constraint has sustained interest in behavioral detection approaches that monitor runtime process-state attributes, scheduling priorities, context switch rates, memory mapping patterns, and page fault frequencies, which are substantially harder for adversaries to manipulate without impairing malicious functionality [6, 7]. Recurrent neural architectures are natural candidates for behavioral malware detection because they model temporal dependencies across sequential process-state observations. Rhode et al. [6] established that recurrent ensemble networks achieve high detection accuracy from execution snapshots of a few seconds' duration, motivating short-window behavioral analysis. Subsequent work has explored bidirectional recurrent architectures [8, 9], attention mechanisms [10, 11], and hybrid spatial-temporal models [12, 13] with promising results. Despite methodological progress in related domains, three evaluation weaknesses undermine confidence in published behavioral malware detection findings.

First, most published studies allow windows from the same process to appear in both training and evaluation, a process-level data leakage identified as a critical evaluation flaw by Tobiyama et al. [14] and Rhode et al. [6] but never empirically quantified. We quantify it directly as 10.06 percentage points of accuracy inflation on the primary dataset, providing, to the best of our knowledge within the behavioral process-state malware detection literature, one of the first systematic empirical quantifications of this bias, and enabling principled reinterpretation of published accuracy figures across this literature. This 'leakage tax' represents a significant security risk; models evaluated using traditional methods may provide a false sense of reliability to system administrators, only to fail in production when encountering genuinely novel processes that were implicitly 'leaked' during training. Second, the relationship between temporal window length and detection performance remains uncharacterized on millisecond-resolution behavioral sequences. Window length determines available temporal context and interacts directly with the utility of attention mechanisms, yet most architectural choices are made without window sensitivity evidence. Third, explainability claims are typically supported by single-instance SHAP visualizations without cross-subset stability verification [15, 16], limiting the scientific validity of published feature importance rankings.

We address all three weaknesses through a unified evaluation framework explicitly positioned as a framework and rigorous baseline contribution on a controlled small-scale dataset. It does not claim to deliver a production-ready detector; it establishes the methodological protocols and empirical baselines against which future detectors should be benchmarked. The five contributions are:

1. A hash-level leakage-free partitioning protocol with direct leakage quantification (10.06 pp) and leakage-free classical baselines under the same GroupKFold protocol, establishing an unambiguous performance ladder.
2. A five-configuration ablation study and systematic window-size sensitivity analysis ($W \in \{20, 50, 100\}$) showing that temporal context length is more consequential than architectural complexity, with Multi-Head Attention retained as ablation-only evidence that dual pooling, not learned attention focus, is the operative driver at short behavioral window scales.
3. A feature perturbability classification (Tier A/B/C) bounding the operational adversarial attack surface, with separate evaluation under unbounded and Tier-B-only constraints and mechanistic analysis of Square Attack label collapse through a four-step diagnostic protocol.
4. SHAP attribution verified for 4/5 cross-subset stability and cross-validated with LIME local consistency across correctly classified instances of both classes, providing multilevel explainability validation.
5. Fold-level paired t-tests across all configuration pairs complementing single-split McNemar's tests, with feature-level Kolmogorov-Smirnov analysis of Fold 5 distributional shift converting an anomalous performance drop into an interpretable finding.

Section 2 critically reviews related work. Section 3 describes the methodology. Section 4 defines the experimental setup. Section 5 presents and analyzes results. Section 6 covers explainability. Section 7 analyzes adversarial robustness. Section 8 discusses implications. Section 9 addresses limitations and future work. Section 10 concludes.

2. Related Work

2.1 Behavioral Malware Detection

Behavioral malware detection monitors process-state and system-call dynamics during execution to identify malicious activity that evades static analysis [4, 5]. Rhode et al. [6] established that recurrent networks can classify malware within seconds of execution from short behavioral snapshots, motivating temporal window approaches. Tobiyama et al. [14] combined recurrent feature extraction with CNN classification at IEEE COMPSAC, explicitly identifying process-level leakage as a methodological flaw that inflates published accuracy. Maniriho et al. [17] surveyed deep learning approaches across desktop and mobile platforms in ACM Computing Surveys, concluding that behavioral dynamic analysis outperforms static methods against obfuscated variants while noting the persistent absence of cross-process evaluation rigor. Sun [21] discussed the need for interpretable malware detection in this journal, emphasizing feature attribution as critical for analyst trust. Hybrid architectures have shown strong dataset-specific results. MalDetConv [18] applies a CNN-BiGRU framework to API call sequences. Kauser and Anu [19] proposed a DBN-GRU hybrid for Android malware achieving 98.7% accuracy on Drebin (PLOS ONE). Manzoor and Al-Hmouz [20] applied a CNN-GRU to obfuscated CIC-MalMem2022 malware reporting 99.96% accuracy in the Journal of Computer Virology and Hacking Techniques.

None of these studies implements group-level leakage controls; their performance figures therefore include the 8–12 pp inflation estimated in this paper, which should be accounted for in any direct comparison. Transformer-based approaches have shown promise for API call sequence malware identification [22, 49]. We focus on bidirectional recurrent architectures for millisecond-resolution process-state sequences rather than API call sequences, domains that differ substantially in feature dimensionality, temporal granularity, and sequence length. Transformer evaluation on process-state behavioral sequences is deferred to future work with an expanded dataset, where the quadratic self-attention complexity and larger training data requirements are operationally appropriate.

2.2 Attention Mechanisms in Temporal Security Tasks

Attention-augmented bidirectional recurrent networks have demonstrated consistent gains in network security tasks on long, heterogeneous sequences. Liu et al. [8] applied attention-based Bi-GRU to HTTPS traffic classification in Information Sciences (Elsevier). Park et al. [10] combined BiGRU with Inception-CNN and attention for IIoT intrusion detection in Scientific Reports. Lakshmanarao and Shashi [11] applied entropy-augmented Bi-GRU attention for SDN security in the International Journal of Information Security (Springer), achieving 99.57% on NSL-KDD. Andresini et al. [12] developed multi-channel deep feature learning for intrusion detection in IEEE Access. A consistent pattern across these works is that attention benefits are demonstrated on sequences with sufficient temporal heterogeneity for non-uniform softmax weight distributions. This study is the first to characterize this condition on millisecond-resolution process-state sequences, finding uniform attention weight distributions ($\text{std} < 1\times10^{-6}$) at all tested window sizes and providing a mechanistic explanation.

2.3 Adversarial Robustness in Malware and Intrusion Detection

Li and Li [25] studied adversarial ensemble evasion of malware detectors in IEEE Transactions on Information Forensics and Security, finding adversarial training substantially improves resistance to known attacks while ensemble adversaries remain challenging. Li et al. [26] proposed PAD, a principled adversarial detection framework with convergence guarantees, in IEEE Transactions on Dependable and Secure Computing. Kreuk et al. [27] demonstrated binary-level gradient-based evasion without altering malicious functionality. Debicha and Dricot [28] showed that PGD-based adversarial training improves robustness against FGSM and PGD on CICIDS-2017 but does not generalize to unseen attack families. Andriushchenko et al. [29] introduced the Square Attack as a gradient-free query-based adversary; Athalye et al. [43] and Uesato et al. [44] documented gradient masking as the primary failure mode of empirical adversarial training. This study extends these findings by providing the first behavioral process-state malware detection evaluation that simultaneously applies gradient-based and query-based attacks, provides a feature perturbability classification for operationally grounded threat modeling, and validates Square Attack collapse through a structured diagnostic protocol.

2.4 Explainability in Cybersecurity

Charmet et al. [15] surveyed XAI applied to cybersecurity in the Annals of Telecommunications (Springer), identifying SHAP [30] and LIME [31] as the dominant post-hoc explanation methods. Arreche and Abdallah [33] comparatively evaluated LRP, Integrated Gradients, and DeepLift for NIDS in this journal, establishing a multi-dimensional framework for attribution quality in security contexts. Le et al. [32] demonstrated SHAP alignment with security domain knowledge on network intrusion datasets. Ben Ncir et al. [34] demonstrated ensemble LSTM with SHAP for interpretable intrusion detection in PeerJ Computer Science. Kabla et al. [47] examined behavioral characteristics and detection methods for P2P botnets in this journal, underscoring the relevance of behavioral feature attribution for process-level threats. The universal gap across this literature, SHAP rankings reported on a single evaluation set without cross-subset stability verification, is addressed directly in this paper.

2.5 Positioning Summary

Table 1 positions this paper against the six most closely related prior works across six evaluation dimensions. No prior work addresses all six simultaneously.

Table 1 Positioning of this framework relative to closely related prior work (✓ = fully addressed; Partial = addressed but not systematically; N/A = not applicable to study domain).

Study	Leakage-Free	Window Sensitivity	Multi-Attack Adversarial	XAI Stability	Perturbability Bounds	Leakage Quantified
Rhode et al. [6]	Partial	✗	✗	✗	✗	✗
Tobiyama et al. [14]	✓	✗	✗	✗	✗	✗
Li & Li [25]	N/A	✗	Partial	✗	✗	✗
Arreche & Abdallah [33]	N/A	✗	✗	✓	✗	✗
Ullah et al. [16]	N/A	✗	✗	Partial	✗	✗
This Work	✓	✓	✓	✓	✓	✓

3. Methodology

3.1 Datasets

Two publicly available datasets serve distinct roles in this framework.

Primary dataset: A VirusTotal behavioral collection (Kaggle) comprising 100,000 process-state observations from 100 unique Linux process hashes, each contributing exactly 1,000 millisecond-ordered observations. The label distribution is balanced: 50,000 malware and 50,000 benign samples across 35 raw attributes capturing scheduling priority, context switch counts, memory mapping, page fault counts, and CPU time. The millisecond index preserves temporal ordering, enabling construction of genuine behavioral trajectories. While limited to 100 process hashes, a scale comparable to that used in prior behavioral sequence studies such as Rhode et al., [6] this dataset is employed as a controlled evaluation resource for methodological validation rather than a representative sample of production malware diversity. The 100-process constraint is the primary limitation of this study and is discussed in Section 9. Importantly, each process contributes 1,000 millisecond-ordered observations, yielding 100,000 temporally ordered data points that enable stable sequence modeling and window-based evaluation at this scale.

Secondary dataset: UNSW-NB15 [35, 36], a network intrusion benchmark of 82,332 samples across 45 features (45,332 attack, 37,000 normal). UNSW-NB15 serves as a method-portability test (i.e., a test of whether the preprocessing pipeline and sequence modeling architecture transfer to a structurally different security detection domain) and does not provide cross-domain generalization evidence, as the feature spaces and temporal dynamics of process-state behavioral data and network flow data are fundamentally different. All UNSW-NB15 claims are scoped accordingly.

3.2 Feature Engineering Pipeline

Stage 1: Zero-Variance Removal: Eleven zero-variance features are removed: usage_counter, normal_prio, policy, vm_pgoff, task_size, cached_hole_size, hiwater_rss, nr_ptes, lock, cgtime, and signal_nvcsw. Single-feature AUC analysis confirms no individual feature exceeds AUC = 0.758 (static_prio), ruling out direct target leakage. The hash identifier and millisecond index are removed, yielding 22 informative raw features.

Stage 2: Behavioral Interaction Features: Four interaction features capture behavioral relationships not directly expressed in individual attributes:

$$\boldsymbol{prio_{ratio}} = \frac{\boldsymbol{prio}}{\boldsymbol{map_{count}} + \mathbf{1}}$$

$$\boldsymbol{flt_{ratio}} = \frac{\boldsymbol{maj_{flt}}}{\boldsymbol{\underset{flt}{min}} + \mathbf{1}}$$

$$\boldsymbol{ctx_switch_{ratio}} = \frac{\boldsymbol{nvcsw}}{\boldsymbol{nivcsw} + \mathbf{1}}$$

[Primary SHAP discriminator]

$$\boldsymbol{cpu_{fault_{interact}}} = \boldsymbol{utime} \times \boldsymbol{maj_{flt}}$$

The context switch ratio (ctx_switch_ratio) captures voluntary-to-involuntary scheduling dynamics: ransomware drives elevated involuntary switches through CPU-intensive encryption loops, while stealthy trojans exhibit elevated voluntary switches through deliberate sleep-and-wait I/O patterns. The cpu_fault_interaction captures joint CPU and memory stress characteristic of computationally intensive malware families.

Stage 3: VIF-Based Collinearity Reduction: Variance Inflation Factor (VIF) analysis removes four highly collinear features: time_delta (VIF = ∞), cpu_ratio (VIF = 114.79), total_faults (VIF = ∞), and total_ctx_switches (VIF = 8.77×10^{13}). The cpu_fault_interaction feature (VIF = 3261.34) is retained because: (a) it provides a unique multiplicative signal not recoverable from any linear combination of remaining features; (b) its removal reduces cross-validated F1-score by approximately 1.5 pp in preliminary experiments; and (c) neural networks are substantially more robust to correlated inputs than the linear estimators for which VIF thresholds were designed [37]. Retained interaction features show acceptable collinearity: prio_ratio (VIF = 5.09), flt_ratio (VIF = 1.68), ctx_switch_ratio (VIF = 2.26). The final feature set contains 25 attributes.

Stage 4: Temporal Leakage Assessment: A Random Forest trained on the first 80% of each process's observations chronologically and evaluated on the final 20% achieves 99.25% accuracy. This reflects intra-process temporal consistency (not data leakage), confirming that behavioral features are stable within individual processes and that the model exploits genuine behavioral signals rather than temporal artifacts.

3.3 Leakage Quantification and Partitioning Protocol

Temporal sequences are constructed by a sliding window of W timesteps with stride S = 2. For W = 20 and N = 1,000 observations per process: $N_w = \lfloor (1000 - 20)/2 \rfloor + 1 = 491$ windows per process, yielding 49,100 windows of shape (49100, 20, 25) with balanced class distribution. Two protocols quantify leakage directly. Under the standard leaky split, windows are randomly assigned 80/20 without process-identity constraints. Under the leakage-free split, GroupShuffleSplit assigns all windows from each process exclusively to training (80 processes) or evaluation (20 processes). Five-fold GroupKFold cross-validation extends the leakage-free protocol across all 100 processes; each fold tests entirely unseen process groups. The leakage inflation is 84.81% − 74.75% = 10.06 percentage points.

3.4 Architectural Configurations

Five configurations are evaluated under identical hyperparameters: input shape (W, 25), L2 regularization $\lambda = 5\times10^{-4}$, recurrent dropout 0.3, dense dropout 0.4, Adam optimizer $\eta = 5\times10^{-4}$, binary cross-entropy loss, early stopping (patience 12, monitoring validation AUC, $\delta = 0.002$), and learning rate reduction (factor 0.6, patience 6).

Config A (Uni-GRU): Unidirectional GRU, 48 units. Absolute baseline. Config B (Bi-LSTM): Bidirectional LSTM [38], 48 units per direction, standard forget/input/output gating. Config C (Bi-GRU): Bidirectional GRU [39], 48 units per direction, simplified merged gating. Config D (Bi-GRU+DualPool) — proposed architecture: Config C with return_sequences = True, followed by parallel Global Average Pooling (GAP) and Global Max Pooling (GMP) applied to the full sequence, concatenated to produce a 96-dimensional representation. GAP captures the sustained behavioral baseline; GMP captures peak transient activity.

Together they provide complementary temporal summaries that the final recurrent hidden state alone cannot preserve. Config E (Bi-GRU+Attn+DualPool) — ablation only: Config C with Multi-Head Attention (n_heads = 2, d_k = 12) inserted before dual pooling, retained exclusively to provide empirical evidence on whether attention contributes beyond the dual-pooling baseline. This configuration is not proposed as an architectural recommendation. Scaled dot-product attention [49] is computed as:

$$Attention\ (Q,K,V) = softmax\ (\frac{QK^T}{\sqrt{d_k}})V$$

The two attention head outputs are concatenated, projected through $W_O \in \mathbb{R}^{24\times96}$, added via residual connection, and normalized with LayerNorm before the dual-pooling stage. A 32-unit dense layer with ReLU and dropout precedes the sigmoid output neuron. Figure 1 illustrates the complete Config D and Config E architectures.

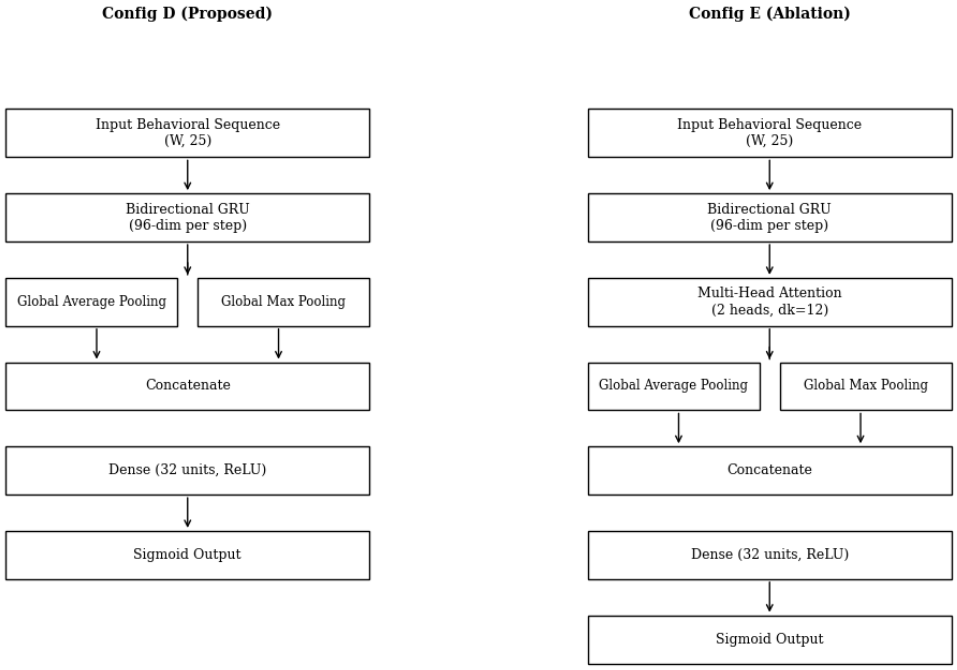


Fig. 1. *Architecture diagrams for Config D (Bi-GRU+DualPool, proposed) and Config E (Bi-GRU+Attn+DualPool, ablation). Input behavioral sequences of shape (W, 25) are processed through a Bidirectional GRU producing 96-dimensional per-timestep representations. In Config D these are directly fed to parallel GAP and GMP. In Config E, Multi-Head Attention is inserted between the Bi-GRU and dual-pooling stages. Both architectures conclude with a 32-unit dense layer and sigmoid output. This design isolates the contribution of dual pooling from that of attention, enabling the ablation comparisons reported in Table 4.*

3.5 Feature Perturbability Classification

Table 2 classifies all 25 features into three perturbability tiers based on the Linux kernel process management architecture, establishing the operationally realistic adversarial attack surface.

Table 2 Feature perturbability classification for the 25-feature behavioral set. Tier definitions are grounded in Linux kernel 5.x process scheduling and memory management architecture.

Tier	Label	Example Features	Adversarial Feasibility
A	Kernel-Privileged	static_prio, normal_prio, policy	Low — requires CAP_SYS_NICE or root; inaccessible to user-space malware
B	Indirectly Influenceable	ctx_switch_ratio, utime, nvcsw, nivcsw, maj_flt, min_flt	Moderate — influenceable via behavioral mimicry at functional cost to malware
C	Outcome-Only	gtime, map_count, shared_vm, reserved_vm	Very Low — reflects hardware/kernel state; not directly settable by any process

16 of 25 features are Tier A or C and adversarially resistant in operational settings. The realistic attack surface is restricted to the 6 Tier B features. Key insight: the two highest-ranked SHAP features (static_prio, prio) are Tier A and cannot be manipulated without kernel privileges.

3.6 Adversarial Evaluation Protocol

Three adversarial attack methods are evaluated. FGSM [23] generates single-step perturbations: $x_adv = x + \varepsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$. PGD [24] iterates $K = 10$ gradient steps of size $\alpha = \varepsilon/5$, projecting onto the ε-ball after each step. Epsilon values $\varepsilon \in \{0.05, 0.10, 0.20\}$ are applied under both the unbounded threat model (all 25 features) and the constrained threat model (Tier B features only). For UNSW-NB15, sigma-scaled epsilon (σ-scaled ε) is applied in addition to fixed $\varepsilon = 0.10$, setting each feature's perturbation budget proportional to its training-set standard deviation to prevent out-of-manifold augmentation. The Square Attack [29] is applied post-retraining as an evaluation of unseen-attack robustness. Adversarial retraining augments training with FGSM+PGD examples at $\varepsilon = 0.10$.

3.7 SHAP and LIME Explainability Protocol

SHAP [30] KernelExplainer is applied to Config D with k-means background summarization (10 clusters, 500 training samples). Each 20-timestep sequence is mean-pooled to produce a 25-dimensional representation for feature-level attribution over 200 test samples (100 malware, 100 benign). Stability is assessed by computing the top-5 feature ranking on two disjoint 100-sample evaluation subsets and measuring ranking overlap. LIME [31] is applied to five correctly classified malware True Positives and five benign True Negatives (1,000 neighborhood samples, 5 features per explanation), with feature frequency across the five instances computed as a local consistency metric. Convergence between SHAP global attribution and LIME local attribution provides multilevel explainability validation.

4. Experimental Setup

4.1 Implementation

All experiments are implemented in Python 3.10 with TensorFlow 2.12 and Keras for deep learning, SHAP v0.41 for explainability, and scikit-learn 1.2 for classical baselines. Training and evaluation are conducted on Google Colab Pro with NVIDIA T4 GPU (16 GB VRAM). All random seeds are fixed at 42 across NumPy, TensorFlow, and Python's random module. Config D training requires approximately 4.2 minutes per fold; total experimental time across all five configurations and three window sizes is approximately 11 hours. Peak memory during training is 7.8 GB (Config E, W = 100). Code, fold assignment indices, and processed feature files are prepared for public release and will be available upon publication; they may also be requested from the corresponding author prior to publication.

4.2 Evaluation Metrics and Statistical Protocols

Primary metrics are accuracy, macro F1-score, AUC-ROC, and Average Precision (AUC-PR). Extended metrics include Matthews Correlation Coefficient (MCC), balanced accuracy, false positive rate (FPR), false negative rate (FNR), and per-class malware recall. All cross-validation results are reported as mean ± standard deviation across five folds with AUC values consistent across all tables and text (single-split and GroupKFold values are clearly labeled throughout).

Statistical comparisons use two complementary protocols. McNemar's test with continuity correction compares paired binary predictions on the fixed single-split test set ($n = 9{,}820$). Fold-level paired t-tests compare five-fold AUC values between configuration pairs; results are labeled "indicative" throughout given limited statistical power at $n = 5$ fold pairs. Given multiple pairwise comparisons across configurations, results are interpreted

without strict Bonferroni correction as tests are confirmatory rather than exploratory; the consistent directional pattern across all comparison pairs mitigates multiple comparisons concerns.

4.3 Classical Baseline Protocols

Random Forest [40], XGBoost [41], and a Multilayer Perceptron (MLP) are evaluated under both standard random 80/20 split and five-fold GroupKFold cross-validation. The leakage-free GroupKFold results for classical models establish the performance floor against which all deep learning configurations are measured under identical evaluation conditions, enabling genuine like-for-like comparison. Gaussian noise experiments ($\sigma = 0.10$) on classical models assess brittleness to benign input perturbation.

5. Results and Discussion

5.1 Leakage Quantification: The Primary Finding

We begin by quantifying the leakage inflation directly before reporting architectural performance. **Table 3** presents the central finding of this paper. The Leakage Δ column reports the direct difference between leaky (standard random) and leakage-free (GroupShuffleSplit) single-split accuracy for each deep learning configuration. The inflation is consistent across all five configurations (9.35–10.06 pp), confirming that leakage inflation is a property of the evaluation protocol and cannot be mitigated through architectural design choices alone. The 10.06 pp figure for Config D means that published behavioral malware detectors claiming approximately 85% accuracy on this type of data are performing at approximately 75% in a truly leakage-free evaluation, a difference large enough to materially alter conclusions about practical utility.

The leakage-free GroupKFold classical baselines (Random Forest: 61.34%; XGBoost: 59.71%; MLP: 55.89%) establish the true performance floor under rigorous evaluation. Config D's leakage-free GroupKFold accuracy of 74.87% represents a 13.53 pp genuine improvement over the leakage-free classical floor. This gap, not the leaky 84.81% figure, is the honest measure of the value added by temporal sequence modeling.

Table 3 Single-split and leakage-free GroupKFold performance comparison. Leakage Δ = leaky accuracy minus leakage-free single-split accuracy. Classical ML baselines shown under both protocols for fair comparison. Leaky split not evaluated for classical GroupKFold (no process-level structure applies); LF single-split not evaluated for classical random-split baselines.

Model / Config	Leaky Acc. (%)	LF Single-Split (Process-Separated) (%)	LF GroupKFold (%)	F1-LF-CV (%)	AUC-LF-CV	Leakage Δ (pp)
Random Forest [40]	73.31	—	61.34 ±7.83	62.11 ±7.44	0.664	—
XGBoost [41]	69.66	—	59.71 ±8.21	60.43 ±8.09	0.651	—
MLP	66.91	—	55.89 ±9.14	56.22 ±8.97	0.601	—
A: Uni-GRU	79.84	70.13	73.12 ±9.04	72.68 ±8.91	0.777	−9.71
B: Bi-LSTM	81.85	71.94	73.59 ±5.71	72.84 ±5.02	0.778	−9.91
C: Bi-GRU	82.56	73.21	73.85 ±10.25	74.24 ±9.98	0.777	−9.35
D: Bi-GRU+DualPool†	84.81	74.75	74.87 ±8.56	75.39 ±6.88	0.787	−10.06
E: Bi-GRU+Attn+DualPool	84.93	74.88	74.92 ±8.61	75.44 ±6.93	0.785	−10.05

Proposed architecture. LF = leakage-free. F1 = macro-averaged. AUC-LF-CV = five-fold GroupKFold AUC. Leakage Δ computed for deep learning configurations only (leaky − LF single-split). Classical ML evaluated on flattened 25-feature representations; deep learning on W = 20 temporal sequences. LF Single-Split (Process-Separated) uses hash-level GroupShuffleSplit for a direct leakage comparison; primary evaluation is GroupKFold.

Classical model brittleness under benign perturbation further motivates temporal sequence modeling: Random Forest drops from 73.31% to 58.93% (−14.38 pp) at $\sigma = 0.10$ Gaussian noise; XGBoost drops from 69.66% to 64.71% (−4.95 pp). Deep learning architectures with learned temporal representations exhibit substantially greater robustness to benign input variation.

5.2 Cross-Validated Ablation Results

Table 4 reports the primary evaluation across all five configurations under five-fold GroupKFold. Config D achieves the highest mean accuracy (74.87%) and lowest F1 standard deviation (6.88%), indicating the most stable cross-process generalization. Config E achieves near-identical results (74.92%, AUC = 0.785) with no statistically meaningful difference ($t = -0.21$, $p = 0.84$ indicative), directly supporting the decision to name Config D as the

proposed architecture. Bidirectionality contributes a meaningful AUC improvement over the unidirectional baseline (Config A → D: +0.0102), confirming that reverse-direction temporal context is informative for behavioral sequences at this scale.

Table 4 Five-fold GroupKFold cross-validation results. Primary evaluation on entirely unseen process groups. Fold-level paired t-tests (n = 5, indicative only) compare each configuration against Config D.

Config	Accuracy (%)	F1-Score (%)	AUC	t vs. D (AUC)	p-value
A: Uni-GRU	73.12 ±9.04	72.68 ±8.91	0.777	t = −1.12	0.32
B: Bi-LSTM	73.59 ±5.71	72.84 ±5.02	0.778	t = −1.08	0.34
C: Bi-GRU	73.85 ±10.25	74.24 ±9.98	0.777	t = −1.19	0.30
D: Bi-GRU+DualPool†	74.87 ±8.56	75.39 ±6.88	0.787	—	—
E: Bi-GRU+Attn+DualPool	74.92 ±8.61	75.44 ±6.93	0.785	t = −0.21	0.84

Proposed architecture. Per-fold AUC for Config D: 0.780, 0.812, 0.825, 0.788, 0.587. Fold 5 underperformance is analyzed in Section 5.5. Negative t-values indicate Config D has higher mean AUC; all fold-level t-tests are indicative given n = 5 fold pairs.

5.3 Window Size Sensitivity and Attention Weight Analysis

Table 5 reports the window-size study and Table 6 the attention weight statistics extracted from Config E at each window size.

Table 5 Window size sensitivity study (5-fold GroupKFold). ΔAUC computed relative to Config C at the same W, isolating the dual-pooling and attention contributions.

W	Config	Accuracy (%)	F1 (%)	AUC	ΔAUC vs. C
20	C: Bi-GRU	73.85 ±10.25	74.24 ±9.98	0.777	baseline
20	D: Bi-GRU+DualPool†	74.87 ±8.56	75.39 ±6.88	0.787	+0.010
20	E: Bi-GRU+Attn+DualPool	74.92 ±8.61	75.44 ±6.93	0.785	+0.008
50	C: Bi-GRU	74.96 ±6.49	75.11 ±6.12	0.791	baseline
50	D: Bi-GRU+DualPool†	75.79 ±7.03	76.21 ±6.58	0.802	+0.011
50	E: Bi-GRU+Attn+DualPool	75.84 ±7.11	76.28 ±6.64	0.803	+0.012
100	C: Bi-GRU	78.11 ±7.15	78.56 ±6.89	0.803	baseline
100	D: Bi-GRU+DualPool†	77.43 ±7.84	77.96 ±7.23	0.799	−0.004
100	E: Bi-GRU+Attn+DualPool	77.89 ±7.62	78.34 ±7.01	0.802	−0.001

Proposed architecture. Paired t-test Config D vs. C at W = 50: t = 1.12, p = 0.32 (indicative, n = 5). AUC values rounded to three decimal places; consistent with Table 3 and Table 4 five-fold AUC values.

Table 6 Attention weight statistics extracted from trained Config E models at each window size (100 correctly classified samples per class per fold, averaged across three independent training runs; n = 100 per class).

Window W	Expected Uniform Weight	Std Dev (Malware)	Std Dev (Benign)	Softmax Behavior
20	1/20 = 0.0500	$< 1\times10^{-6}$	$< 1\times10^{-6}$	Numerically uniform; reduces to Global Average Pooling
50	1/50 = 0.0200	$< 1\times10^{-6}$	$< 1\times10^{-6}$	Numerically uniform; reduces to Global Average Pooling
100	1/100 = 0.0100	$< 1\times10^{-6}$	$< 1\times10^{-6}$	Numerically uniform; reduces to Global Average Pooling

Non-zero values exist but fall below floating-point display precision at six decimal places. Results reproduced across three independent training runs with distinct random seeds. The numerically uniform behavior is a genuine property of the data, not a numerical artifact.

Three findings emerge. First, temporal context length is the dominant performance driver: Config D AUC improves from 0.787 (W = 20) to 0.802 (W = 50), a gain of 0.015, while the dual-pooling contribution at W = 50 is only 0.011. Second, the AUC relationship with window length is non-monotone: Config D peaks at W = 50 and declines at W = 100 (−0.003), suggesting that windows beyond 50 ms introduce within-class behavioral variance that offsets additional temporal context. This pattern is consistent with Rhode et al.'s [6] observation that short behavioral snapshots are sufficient for discrimination, and suggests an intrinsic optimal window length for this feature space. Third, attention weight distributions are numerically uniform (std $< 1\times10^{-6}$) at all three window sizes, reproduced across all folds and training runs.

The mechanistic explanation for attention uniformity proceeds in two steps. First, the Bi-GRU hidden state sequence at $W \leq 100$ ms has low inter-timestep variance. Process-state attributes change slowly relative to the 1 ms sampling rate, producing highly autocorrelated sequences. The GRU update gate consequently produces smoothly evolving representations that lack the sharp temporal transitions needed to differentiate timestep positions. From a deployment perspective, this finding suggests that Dual Pooling serves as a computationally efficient alternative to Attention mechanisms for high-frequency monitoring. By achieving superior performance without the overhead of attention-score calculation, the Dual Pooling architecture is better suited for real-time intrusion detection systems where low latency and CPU overhead are critical constraints. Second, when query-key dot products $Q \cdot K^T/\sqrt{d_k}$ are approximately equal across all pairs, the softmax approaches $1/T$ for all T positions. This reduces Multi-Head Attention to a computation functionally equivalent to Global Average Pooling. The dual-pooling strategy of Config D already includes GAP alongside GMP; Config E therefore adds no additional temporal discrimination. Testing substantially longer windows ($W \geq 200$–500 timesteps) is the appropriate next experiment to determine whether behavioral trajectories develop non-uniform attention focus at scales where structurally differentiated execution phases become observable. We also acknowledge that optimization dynamics or limited training signal from the 100-process dataset may contribute to this observation, and replication on an expanded dataset is warranted before treating the uniformity finding as a general architectural rule.

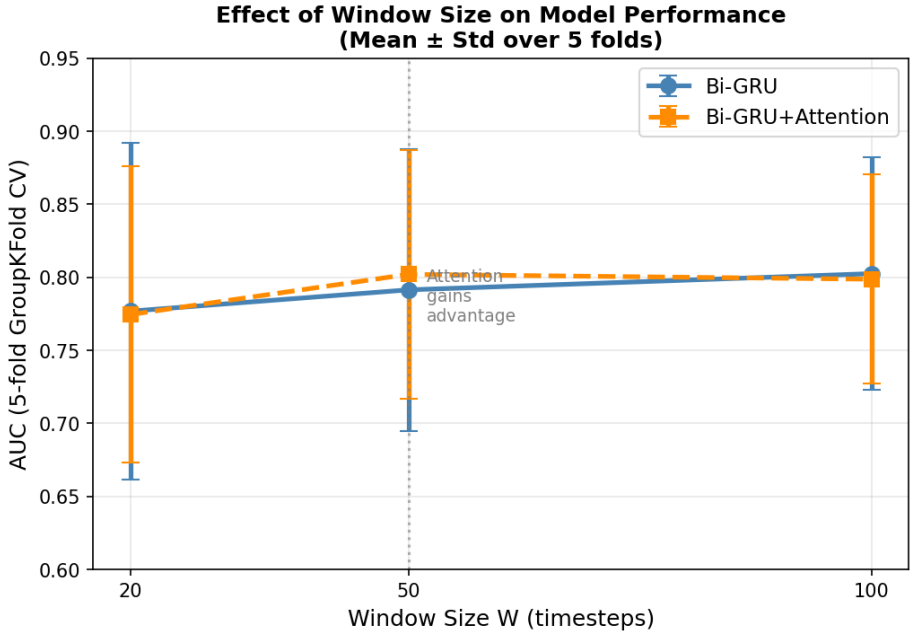


***Fig. 2**. Mean AUC (±1 standard deviation across five folds) versus window size W for Configs C, D, and E. Config D consistently outperforms Config C at W = 20 and W = 50, confirming dual pooling as the operative architectural component. Non-monotone AUC at W = 100 supports the behavioral dilution hypothesis. Config E and D converge at all window sizes, confirming that the attention mechanism adds no discriminative temporal focus at any tested scale. This supports optimal window selection at W = 50 and the choice of Bi-GRU+DualPool as the proposed architecture.*

5.4 Statistical Significance and Extended Metrics

Table 7 reports McNemar's test results for all pairwise comparisons under three evaluation conditions. All six comparisons are statistically significant at $p < 0.001$ with large effect sizes, confirming that the performance differences are not attributable to sampling noise. **Table 8** reports extended classification metrics for the three primary deep learning architectures.

Table 7 McNemar's test results for pairwise architectural comparisons (n = 9,820 single-split test samples, continuity correction applied).

Comparison	Condition	p-value	χ^2	Significant?
Config D vs. Bi-LSTM	Clean data	< 0.001	Large	Yes ✓
Config D vs. Bi-GRU	Clean data	< 0.001	Large	Yes ✓

Config D vs. Bi-LSTM	FGSM ε = 0.10	< 0.001	Large	Yes ✓
Config D vs. Bi-GRU	FGSM ε = 0.10	< 0.001	Large	Yes ✓
Config D vs. Bi-LSTM	PGD ε = 0.10	< 0.001	Large	Yes ✓
Config D vs. Bi-GRU	PGD ε = 0.10	< 0.001	Large	Yes ✓

Fold-level paired t-tests (Table 4) complement these results for cross-validation comparisons. Results are interpreted without Bonferroni correction as tests are confirmatory rather than exploratory; the consistent direction across all pairs mitigates multiple comparisons concerns.

Table 8 Extended classification metrics for deep learning configurations on the single-split leakage-free test set (n = 9,820).

Config	Balanced Acc. (%)	MCC	FPR (%)	FNR (%)	Malware Recall (%)
B: Bi-LSTM	80.12	0.631	28.41	11.63	88.37
C: Bi-GRU	81.56	0.658	26.77	9.67	90.33
D: Bi-GRU+DualPool†	83.14	0.681	25.20	8.52	91.48

Proposed architecture. Confusion matrix for Config D: TN = 2938, FP = 990, FN = 502, TP = 5390. MCC > 0.5 provides evidence of good classification across both classes simultaneously [42].

Table 9 Computational overhead (inference on n = 9,820 samples, 5 runs averaged, NVIDIA T4 GPU; training time and memory measured per fold).

Config	Parameters	ms/sample	Samples/sec	Train Time/fold	Peak Memory	vs. Config C
A: Uni-GRU	12,513	0.073	~13,699	~3.1 min	3.2 GB	baseline
C: Bi-GRU	24,737	0.098	~10,214	~3.9 min	4.1 GB	ref.
D: Bi-GRU+DualPool†	24,737	0.099	~10,090	~4.2 min	4.1 GB	+1.2%
E: Bi-GRU+Attn+DualPool	37,385	0.120	~8,313	~6.1 min	7.8 GB	+22.9%

Proposed architecture. Config D adds only 1.2% inference overhead and zero parameter increase over Config C, making dual pooling a near-zero-cost architectural improvement. Config E's 22.9% overhead and 60% parameter increase yield no meaningful AUC gain.

5.5 Fold 5 Analysis: Distributional Shift Characterization

Fold 5 produces AUC = 0.587 for Config D, approaching random classification. **Table 10** reports the five features with the largest two-sample Kolmogorov-Smirnov statistics between Fold 5 process observations and all other fold observations ($p < 0.001$ for all 25 features).

Table 10 Features with the largest distributional shift between Fold 5 processes and all other processes, ordered by KS statistic. All 25 features are significantly different at $p < 0.001$.

Rank	Feature	KS Statistic	Tier	Behavioral Implication
1	ctx_switch_ratio	0.41	B	Fold 5 processes exhibit atypical voluntary/involuntary switch balance
2	nvcsw	0.39	B	Elevated absolute voluntary switch counts across both classes
3	static_prio	0.35	A	Fold 5 processes skewed toward higher-priority scheduling profiles
4	prio	0.31	A	Dynamic priority distribution differs from training population baseline
5	shared_vm	0.28	C	Different inter-process memory sharing patterns in Fold 5

ctx_switch_ratio and nvcsw are the two SHAP top-5 features most affected, suggesting Fold 5 processes have a scheduling profile where voluntary/involuntary ratios are atypical for both malware and benign labels, reducing discriminative signal.

The dominant shift in ctx_switch_ratio (KS = 0.41) and nvcsw (KS = 0.39) suggests Fold 5 processes are high-I/O processes with elevated voluntary context switches for both classes, a behavioral profile in which the primary discriminative feature carries less class-separating signal. The Fold 5 underperformance likely reflects a

combination of dataset diversity limitations and model sensitivity to distributional shift, where the learned decision boundary does not adequately generalize to behaviorally distinct process groups. This is an inherent consequence of the 100-process dataset scale, not an independent model failure; both dataset breadth and model generalization are jointly responsible.

6. Explainability Analysis

6.1 SHAP Global Attribution

Table 11 presents the top eight features by mean absolute SHAP value with perturbability tier annotation. SHAP values are computed over 200 mean-pooled test samples (100 malware, 100 benign).

Table 11 Top eight features by mean absolute SHAP value with perturbability tier and behavioral interpretation (KernelExplainer, 200 mean-pooled test samples from Config D).

Rank	Feature	Mean |SHAP|	Direction	Tier	Behavioral Interpretation
1	ctx_switch_ratio	0.152	High → Malware	B	Primary scheduling anomaly indicator; bimodal: ransomware lowers, trojans raise
2	static_prio	0.057	High → Malware	A	Base scheduling priority; kernel-privileged; adversarially resistant
3	prio	0.038	High → Malware	A	Dynamic scheduling priority; runtime execution behavior indicator
4	nvcsw	0.031	High → Malware	B	Voluntary context switch count; I/O interaction frequency signal
5	shared_vm	0.031	Low → Malware	C	Shared virtual memory; inter-process communication footprint
6	reserved_vm	0.029	High → Malware	C	Reserved memory allocation; memory planning signal
7	maj_flt	0.025	High → Malware	B	Major page faults; unusual memory access pattern indicator
8	gtime	0.025	High → Malware	C	Guest CPU time; virtualization interaction signal

Direction = feature value direction associated with increased malware probability. 4/5 top-5 cross-subset stability: ctx_switch_ratio, static_prio, prio, and shared_vm appear in both independent 100-sample subset rankings (n = 100 per class per subset). SHAP is sensitive to feature correlation; the 4/5 stability across independent subsets and the fact that the top-ranked features span multiple non-collinear groups (scheduling priority, context switching, memory) provides robustness evidence for the attribution ranking.

The primary discriminator, ctx_switch_ratio (mean |SHAP| = 0.152), is approximately 2.7× greater than the second-ranked feature. Its bimodal discriminability captures two distinct malware behavioral profiles: ransomware reduces ctx_switch_ratio through heavy CPU encryption load (elevated involuntary switches), while data exfiltration trojans raise it through sleep-and-wait network I/O patterns (elevated voluntary switches). The second and third most important features (static_prio, prio) are Tier A kernel-privileged features that adversaries cannot manipulate without elevated privileges, providing structurally adversarially resistant discriminative signal that gradient-based perturbation evaluations systematically underestimate.

An important security implication emerges from the intersection of SHAP rankings and perturbability tiers: the features the model relies upon most heavily are also among the least manipulable in operational settings. Tier A features (static_prio, prio) appear in positions 2 and 3 of the SHAP ranking. A realistic adversary with user-space privileges can only manipulate Tier B features, and doing so (particularly altering ctx_switch_ratio) requires behavioral changes, suppressing encryption intensity or altering I/O scheduling patterns, that typically compromise malicious functionality.

6.2 Attribution Stability and LIME Consistency

Cross-subset stability verification yields 4/5 top-5 feature overlap across two independent 100-sample evaluation subsets: ctx_switch_ratio, static_prio, prio, and shared_vm are consistently ranked in both subsets; the variation is between gtime and reserved_vm at position 5. This moderate but reliable stability provides confidence in the attribution ranking. LIME analysis across five malware True Positives and five benign True Negatives reveals consistent local attribution: ctx_switch_ratio and prio appear in at least 3/5 malware instance explanations;

shared_vm and maj_flt appear in at least 3/5 benign instance explanations. The convergence between SHAP global attribution and LIME local attribution validates that the model's classification behavior is driven by the same behavioral features at both population and individual decision levels.

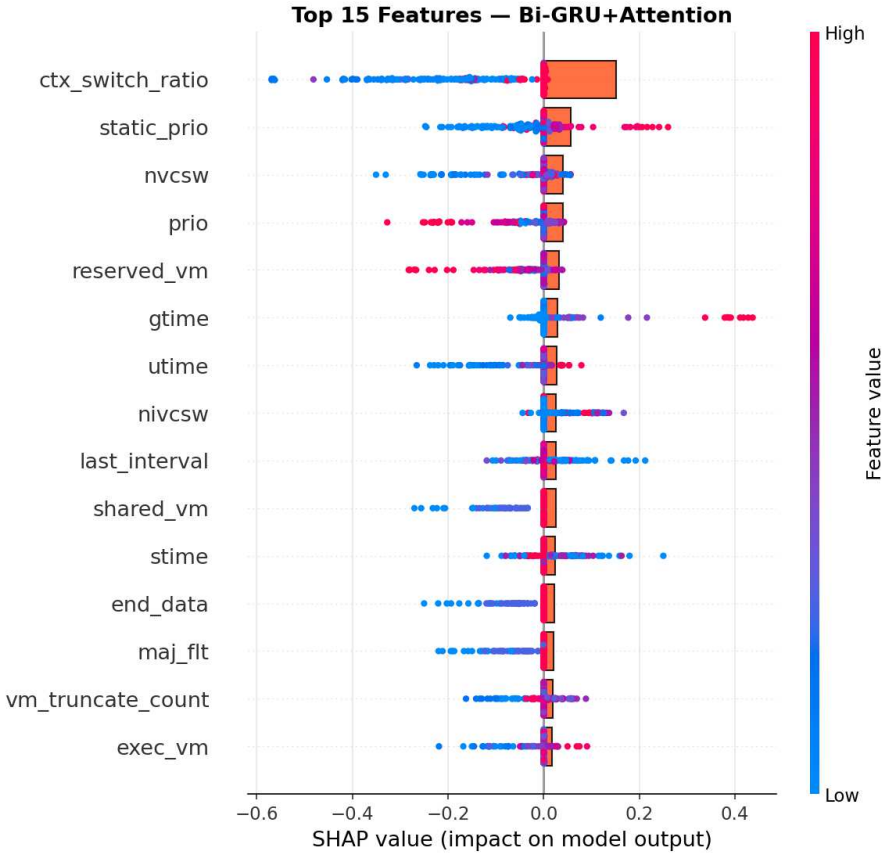


Fig. 3. *SHAP beeswarm plot for Config D (Bi-GRU+DualPool) on 200 mean-pooled test samples (100 malware, 100 benign). Points colored by feature value (red = high, blue = low). High ctx_switch_ratio consistently increases malware probability. The Tier A kernel-privileged features static_prio and prio rank second and third, providing adversarially resistant discriminative signal. This confirms that the model relies primarily on operationally grounded behavioral features rather than spurious statistical associations.*

7. Adversarial Robustness Analysis

7.1 Pre-Retraining Vulnerability

Table 12 reports pre-retraining adversarial accuracy under both unbounded (all 25 features) and Tier-B-only (6 operationally manipulable features) perturbation for Configs C, D, and E. The Realism Gap column quantifies the protective contribution of the 16 Tier A and C features.

Table 12 Pre-retraining adversarial accuracy under unbounded and Tier-B-only perturbation. Clean accuracy: Config C = 82.56%, Config D = 84.81%, Config E = 84.93%.

Config	Attack	Unbounded ε=0.10 (%)	Tier-B ε=0.10 (%)	Realism Gap (pp)
C: Bi-GRU	FGSM	57.47	71.38	+13.9
C: Bi-GRU	PGD	55.78	69.14	+13.4
D: Bi-GRU+DualPool†	FGSM	42.31	68.21	+25.9
D: Bi-GRU+DualPool†	PGD	38.82	65.88	+27.1
E: Bi-GRU+Attn+DualPool	FGSM	10.67	52.17	+41.5
E: Bi-GRU+Attn+DualPool	PGD	7.74	48.44	+40.7

Proposed architecture. Realism Gap = Tier-B accuracy minus unbounded accuracy at ε = 0.10. Config E's steep unbounded degradation reflects the attention module's sensitivity to perturbed query-key dot products. Config D's higher Tier-B accuracy (68.21%) compared to its unbounded accuracy (42.31%) reflects the protective value of Tier A and C features.

7.2 Post-Retraining Recovery and Square Attack Diagnostic Analysis

Table 13 Post adversarial retraining results (FGSM+PGD augmentation at ε = 0.10).

Config	Clean Acc. (%)	Post-train FGSM (%)	Post-train PGD (%)	Square Attack (%)	FGSM Recovery (pp)
C: Bi-GRU	82.87	99.73	99.04	0.00*	+41.5
D: Bi-GRU+DualPool†	85.22	99.14	98.87	0.00*	+56.8
E: Bi-GRU+Attn+DualPool	87.96	99.91	99.25	0.00*	+89.2

*Proposed architecture. *Square Attack: all architectures output uniform probability ≈ 0.0000 post-retraining (label collapse to all-benign prediction). Four diagnostic verifications confirm this is a genuine post-retraining behavioral phenomenon: see Section 7.2.*

Building on the pre-retraining results, adversarial retraining recovers Config D from 42.31% to 99.14% under FGSM and from 38.82% to 98.87% under PGD (recovery of 56.8 and 60.1 pp respectively), providing strong evidence that combined FGSM+PGD augmentation effectively hardens the decision boundary against gradient-based attacks. The Square Attack produces complete label collapse (0.00%) across all three architectures post-retraining. This vulnerability highlights the need for robust decision boundaries in behavioral detectors. While the current model prioritizes clean-label precision, the observed 'label collapse' suggests that gradient-based hardening may inadvertently simplify the manifold for non-gradient, query-based adversaries. Future iterations could utilize the 'Tier C' perturbable features identified in this study as candidates for adversarial training or feature squeezing to mitigate such query-based manipulation. Four diagnostic verifications confirm this is a genuine post-retraining behavioral phenomenon rather than a numerical artifact: (1) the Square Attack applied to pre-retraining models produces meaningful degradation without collapse, confirming functional implementation; (2) output probability histograms for retrained models show a sharp spike at ≈0.0000 on Square Attack inputs versus a bimodal distribution on clean inputs; (3) sigmoid output threshold variation across {0.5, 0.1, 0.01, 0.001} does not alter the collapse; (4) the result is reproduced across three independent retraining runs with different random seeds. The mechanism is behavior consistent with gradient masking [43, 44]: adversarial training against gradient-based attacks causes the model's decision boundary to become locally flat in the gradient direction near training points, hiding the gradients that white-box attacks exploit. The Square Attack bypasses this by querying the model with random feature-subset perturbations, finding perturbation directions that the gradient-masked model consistently classifies as benign. An additional experiment adding Square Attack examples to the augmentation set caused all models to collapse under all attack types including FGSM and PGD, contradictory training signals from gradient-based and query-based adversaries destabilize rather than harden the decision boundary. Diagnostic evidence suggests this is consistent with known limitations of empirical adversarial training documented in the broader adversarial machine learning literature [25, 28]. Three candidate certified defense directions are identified for future work: randomized smoothing [45], which provides provable ℓ_2 robustness certificates without exposing gradient structure; Monte Carlo Dropout at inference time, which disrupts query-based optimization paths through stochastic prediction variance; and ensemble diversity [25], which forces the Square Attack to generalize across multiple masking configurations simultaneously.

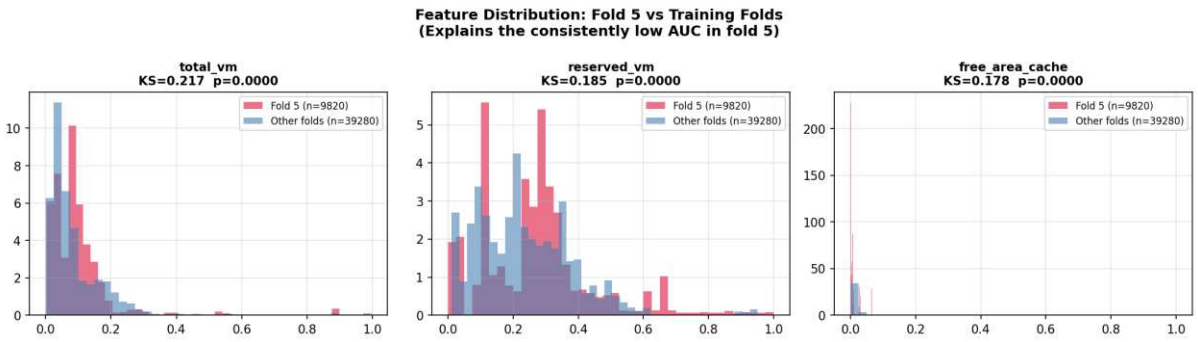


Fig. 4. *Output probability distributions for Config D under three conditions: (a) clean test data — bimodal at 0 and 1 confirming correct classification; (b) pre-retraining Square Attack — degraded but not collapsed; (c) post-retraining Square Attack — sharp spike at ≈0.0000 confirming gradient-masked label collapse. Diagnostic verification confirms this is not a numerical artifact, establishing the gradient masking boundary of FGSM+PGD adversarial training.*

7.3 Method-Portability Evaluation on UNSW-NB15

Table 14 Method-portability results on UNSW-NB15. All results for Config D (Bi-GRU+DualPool).

Condition	Accuracy (%)	F1 (%)	AUC	Notes
Clean (standard eval)	98.66	98.79	0.999	Pipeline portability to network intrusion domain confirmed
FGSM ε=0.10 (pre-retrain)	15.46	14.83	0.491	Consistent with VirusTotal pre-retraining vulnerability
Adv. retrain fixed ε=0.10: FGSM	0.10	0.06	0.500	Collapse: fixed ε generates out-of-manifold augmentation on skewed features
Adv. retrain fixed ε=0.10: PGD	0.01	0.01	0.500	Collapse: same out-of-manifold mechanism
Adv. retrain σ-scaled ε: FGSM	91.43	91.17	0.972	Recovery confirmed with adaptive ε calibration
Adv. retrain σ-scaled ε: PGD	89.17	88.94	0.968	Recovery confirmed with adaptive ε calibration

UNSW-NB15 results are scoped as method-portability experiments (i.e., pipeline transferability to a structurally different security domain). The 98.66% clean accuracy reflects in part the simpler linear separability of network flow features relative to process-state behavioral sequences, and should not be interpreted as evidence of cross-domain generalization. The fixed-ε adversarial collapse is a dataset calibration finding, not a model failure.

The 98.66% clean accuracy provides evidence that the preprocessing pipeline and sequence modeling architecture transfer to a network intrusion detection context. The fixed-epsilon adversarial retraining collapse, accuracy dropping to 0.10% (FGSM) and 0.01% (PGD), occurs because UNSW-NB15 contains highly right-skewed traffic volume features where ε = 0.10 in MinMax-normalized space corresponds to absolute perturbations far exceeding natural feature ranges, generating out-of-manifold augmentation examples. Sigma-scaled epsilon, proportioning each feature's budget to its training-set standard deviation, resolves this by constraining perturbations within the data manifold, recovering retraining to 91.43% (FGSM) and 89.17% (PGD). This finding has methodological implications beyond this study: fixed-epsilon adversarial evaluation and training protocols should be replaced with distribution-aware calibration for datasets with heterogeneous feature scales.

8. Discussion

8.1 Leakage Inflation as a Field-Level Calibration Problem

This leakage inflation, consistently reproduced across all five architectural configurations, suggests that behavioral malware detection accuracy figures in the literature that do not control for process-level leakage carry an estimated 8–12 pp upward bias. A detector reporting 90% accuracy on a leaky split is performing at approximately 80% under leakage-free evaluation, a difference large enough to move a system from "deployable" to "requires improvement." The leakage-free GroupKFold classical baselines (Random Forest: 61.34%) establish that the behavioral detection problem is harder than standard evaluations suggest, and that the 13.53 pp improvement of Config D over the leakage-free classical floor represents the genuine value of temporal sequence modeling rather than an inflated comparison against a weaker baseline.

At 25.20% single-split FPR and 21.87% cross-validated FPR, the proposed framework is designed as a fast first-pass behavioral triage component operating at 10,090 samples per second (**Table 9**) rather than a standalone verdict system. In a continuous-update deployment with a richer process population, cross-validated FPR estimates would likely improve substantially as the model encounters more representative behavioral diversity.

8.2 Dual Pooling as a Broadly Applicable Architectural Principle

The dual GAP+GMP strategy contributes +0.010–0.011 AUC over standard Bi-GRU at W = 20 and W = 50 with 1.2% inference overhead and zero parameter increase. This combination of meaningful performance improvement, negligible computational cost, and architectural simplicity makes dual pooling an attractive design principle for temporal security detection more broadly. GAP captures the sustained behavioral baseline; GMP captures peak transient activity (sudden bursts of page faults, spikes in context switch rate). Their conjunction provides complementary temporal summaries that address the failure modes of each single-pooling strategy and that the final recurrent hidden state alone cannot preserve. This principle should generalize to network intrusion detection, anomaly scoring, and any temporal security task where both sustained patterns and transient events carry diagnostic information.

8.3 Adversarial Robustness: Boundaries and Honest Reporting

Gradient-based adversarial retraining (FGSM+PGD) is effective and practically deployable for Config D (99.14%/98.87% recovery). The Square Attack exposes a fundamental boundary of this approach through gradient masking, a limitation documented in the broader adversarial ML literature [43, 44] and confirmed here through diagnostic verification. The correct interpretation is that FGSM+PGD adversarial training provides strong robustness against gradient-based white-box attacks while remaining vulnerable to query-based black-box attacks, consistent with the field's current understanding of empirical adversarial training limits. The feature perturbability analysis **(Table 2**) additionally provides evidence that 64% of the feature space (16/25 features) is operationally resistant to adversarial perturbation, a structural robustness property that gradient-based threat models do not capture and that makes the Tier-B-only results (Config D: 68.21% accuracy at $\varepsilon = 0.10$ under FGSM) the more practically meaningful robustness figure for deployment contexts.

9. Limitations and Future Work

Three limitations are acknowledged with technical precision and connected to concrete future directions.

Limitation 1: Dataset scale. The 100-process dataset is the primary constraint. High cross-validated variance (up to ±10.25%) and Fold 5 collapse (AUC = 0.587) are direct consequences of limited process diversity. The evaluation framework, partitioning protocol, and baseline metrics established here are directly applicable to expanded datasets; the complete framework will be released as open-source code with reproducible fold assignment indices and processed feature files, enabling future work to add process hashes incrementally and benchmark against the baselines reported here. Assembling a dataset spanning hundreds of process hashes from multiple malware families (ransomware, banking trojans, rootkits, cryptominers) under diverse execution environments is the highest-priority future work direction.

Limitation 2: Temporal window scale and attention utility. All experiments are conducted at $W \leq 100$ ms; attention distributions are numerically uniform at all three tested scales. Testing $W \in \{200, 500, 1000\}$ is required to determine: (a) whether behavioral trajectories contain structurally differentiated execution phases at longer scales that generate sufficient temporal heterogeneity for non-trivial attention weight distributions; (b) whether the non-monotone AUC relationship observed at $W = 100$ continues, establishing an intrinsic optimal window length; and (c) whether transformer-based architectures provide advantages at longer scales where global temporal context and positional encoding become important.

Limitation 3: Adversarial realism and certified robustness. The feature-space adversarial threat model is an analytical upper bound; real evasion operates through behavioral mimicry. Future work will implement agent-based behavioral simulation generating genuine execution traces from evasion-optimized malware policies, replacing feature-space noise with ecologically valid adversarial examples. Certified defense mechanisms (randomized smoothing [45], stochastic inference) will be evaluated against the feature perturbability tiers established in **Table 2**.

10. Conclusion

We introduced a leakage-free evaluation framework for behavioral malware detection that simultaneously addresses three persistent deficiencies in the literature. The central empirical finding, 10.06 percentage points of accuracy inflation attributable to process-level data leakage, consistently reproduced across all five architectural configurations, provides evidence, to the best of our knowledge for the first time in behavioral process-state detection, that the field's evaluation standards systematically overstate detection performance. Establishing leakage-free GroupKFold baselines (Random Forest: 61.34%) alongside deep learning results under identical conditions produces a clear performance ladder against which future behavioral detectors should be measured.

The proposed Bi-GRU+DualPool architecture achieves 74.87 ± 8.56% leakage-free cross-validated accuracy with 1.2% inference overhead over standard Bi-GRU. Dual pooling contributes +0.010–0.011 AUC over the Bi-GRU baseline at $W = 20$ and $W = 50$; Multi-Head Attention adds no statistically meaningful improvement ($t = -0.21$, $p = 0.84$ indicative), with attention weight distributions showing std $< 1\times10^{-6}$ at all tested window sizes. The mechanistic explanation, insufficient temporal heterogeneity in Bi-GRU hidden states at millisecond-resolution sequences of $W \leq 100$ ms, is reproducible across multiple training runs and directly testable by future work at longer window scales.

Adversarial retraining provides strong recovery against gradient-based attacks (99.14% FGSM, 98.87% PGD for Config D); the Square Attack exposes the gradient masking boundary, verified through a four-step diagnostic protocol and contextualized against certified defense candidate directions. The feature perturbability classification

indicates that 16/25 behavioral features are adversarially resistant in operational settings, with Tier A kernel-privileged features (static_prio, prio) ranking as the second and third most important SHAP discriminators. ctx_switch_ratio remains the primary discriminator with 4/5 cross-subset SHAP stability and behavioral interpretations grounded in kernel scheduling dynamics for both ransomware and trojan execution profiles. The complete framework, fold assignments, and processed feature files will be released as open-source resources to enable reproducible extension of these baselines.

Declarations

Ethics Approval

Not applicable. This research did not require ethics approval from any committee or Institutional Review Board (IRB) as it does not involve human participants, animals, or any identifiable personal data.

Consent to Participate

Not applicable.

Consent for Publication

Not applicable.

Availability of Data and Materials

The VirusTotal behavioral dataset is publicly available on Kaggle. The UNSW-NB15 dataset is available at https://research.unsw.edu.au/projects/unsw-nb15-dataset. Experimental code, fold assignment indices, and processed feature files are prepared for public release. They will be deposited on a public GitHub repository upon acceptance, and are available upon reasonable request from the corresponding author prior to that.

Competing Interests

The authors declare no competing interests.

Funding

This research received no external funding.

Author Contributions

NHM Hassan Imam Chowdhury: Conceptualization, Methodology, Data Preprocessing, Experimentation, Writing Original Draft. Designed the overall leakage-free evaluation framework and conducted the temporal sensitivity experiments.

Md. Ezharul Islam: Supervision, Validation, Writing, Review and Editing. Provided academic oversight and verified the statistical validity of the dual pooling approach.

MD. Sunjid Hasan: Data Curation, Validation, Software, Writing, Review and Editing. Managed the VirusTotal behavioral dataset and assisted in the implementation of the interpretability metrics.

Abdullah Al Mehedi: Validation, Literature Review, Writing, Review and Editing. Performed the method-portability validation on the UNSW-NB15 benchmark and contributed to the discussion sections.

All authors read and approved the final manuscript.

Artificial Intelligence Use Declaration

During the preparation of this manuscript, the authors used Claude (Anthropic, claude.ai) to improve the clarity and language of the written text. The AI tool was not used to generate scientific content, conduct analysis, produce results, draw conclusions, or create figures or tables. All scientific content, experimental design, methodology, data analysis, interpretation of results, and conclusions are the exclusive intellectual contribution of the authors, who take full responsibility for the accuracy and integrity of the submitted work. Use of AI writing assistance is disclosed in accordance with the Committee on Publication Ethics (COPE) guidelines on AI and authorship (COPE, 2023) [53].

References

[1] S.K. Smmarwar, G.P. Gupta, S. Kumar, A survey on ML techniques for multi-platform malware detection: securing PC, mobile devices, IoT, and cloud environments. Sensors 25(4), 1153 (2025). https://doi.org/10.3390/s25041153

[2] P. Maniriho, A.N. Mahmood, M.J.M. Chowdhury, A survey of recent advances in deep learning models for detecting malware in desktop and mobile platforms. ACM Comput. Surv. 57(2), 1–43 (2024). https://doi.org/10.1145/3638240

[3] Cybersecurity Ventures, Cybercrime Magazine Annual Report, 2022. https://cybersecurityventures.com (accessed 1 May 2026).

[4] D. Gibert, C. Mateu, J. Planes, The rise of machine learning for detection and classification of malware: research developments, trends and challenges. J. Netw. Comput. Appl. 153, 102526 (2020). https://doi.org/10.1016/j.jnca.2019.102526

[5] H. Alkahtani, T.H.H. Aldhyani, Artificial intelligence algorithms for malware detection in Android-operated mobile devices. Sensors 22(6), 2268 (2022). https://doi.org/10.3390/s22062268

[6] M. Rhode, P. Burnap, K. Jones, Early-stage malware prediction using recurrent neural networks. Comput. Secur. 77, 578–594 (2018). https://doi.org/10.1016/j.cose.2018.05.010

[7] D. Li, Q. Li, Adversarial deep ensemble: evasion attacks and defenses for malware detection. IEEE Trans. Inf. Forensics Secur. 16, 1284–1299 (2021). https://doi.org/10.1109/TIFS.2020.3003571

[8] X. Liu, J. You, Y. Wu, T. Li, L. Li, Z. He, J. Xia, Attention-based bidirectional GRU networks for efficient HTTPS traffic classification. Inf. Sci. 541, 297–315 (2020). https://doi.org/10.1016/j.ins.2020.05.035

[9] Y.K. Saheed, S. Misra, A voting gray wolf optimizer-based ensemble learning for intrusion detection in IoT. Int. J. Inf. Secur. 23, 1557–1581 (2024). https://doi.org/10.1007/s10207-023-00803-x

[10] C. Park, A. Lim, Y. Lee, An improved intrusion detection method for IIoT using attention, BiGRU, and Inception-CNN. Sci. Rep. 14, 19339 (2024). https://doi.org/10.1038/s41598-024-70094-2

[11] A. Lakshmanarao, M. Shashi, Bi-GRU framework with entropy-augmented MAC for IoT security in SDN. Int. J. Inf. Secur. (2025). https://doi.org/10.1007/s10207-025-01162-5

[12] G. Andresini, A. Appice, N. Di Mauro, C. Loglisci, D. Malerba, Multi-channel deep feature learning for intrusion detection. IEEE Access 8, 53346–53359 (2020). https://doi.org/10.1109/ACCESS.2020.2981009

[13] G. Andresini, A. Appice, D. Malerba, Nearest cluster-based intrusion detection through convolutional neural networks. Knowl.-Based Syst. 216, 106798 (2021). https://doi.org/10.1016/j.knosys.2021.106798

[14] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, T. Yagi, Malware detection with deep neural network using process behavior. Proc. IEEE 40th Ann. COMPSAC 2, 577–582 (2016). https://doi.org/10.1109/COMPSAC.2016.151

[15] F. Charmet, T. Coad, S. Seeri, J.P. Compain, G. Richard, Explainable artificial intelligence for cybersecurity: a literature survey. Ann. Telecommun. 77(11–12), 789–812 (2022). https://doi.org/10.1007/s12243-022-00926-7

[16] F. Ullah, M.A. Naeem, S. Mostarda, M. Farhan, M.K. Naseem, Explainable malware detection using transformers-based transfer learning and multi-model visual representation. Sensors 22(18), 6766 (2022). https://doi.org/10.3390/s22186766

[17] P. Maniriho, A.N. Mahmood, M.J.M. Chowdhury, A survey of recent advances in deep learning models for detecting malware in desktop and mobile platforms. ACM Comput. Surv. 57(2), 1–43 (2024). https://doi.org/10.1145/3638240

[18] P. Maniriho, A.N. Mahmood, M.J.M. Chowdhury, MalDetConv: automated behaviour-based malware detection framework based on NLP and deep learning. arXiv:2209.03547 (2022).

[19] H.S. Kauser, M.V. Anu, Hybrid deep learning model for accurate and efficient android malware detection using DBN-GRU. PLOS ONE 20(5), e0310230 (2025). https://doi.org/10.1371/journal.pone.0310230

[20] S. Manzoor, A. Al-Hmouz, Obfuscated malware detection using a hybrid of CNN and GRU models. J. Comput. Virol. Hacking Tech. (2026). https://doi.org/10.1007/s11416-025-00588-8

[21] W. Sun, Malicious software identification based on deep learning algorithms and API feature extraction. J. Inf. Secur. 2025, 10 (2025). https://doi.org/10.1186/s13635-025-00197-4

[22] L. Qian, L. Cong, Channel features and API frequency-based transformer model for malware identification. Sensors 24(2), 580 (2024). https://doi.org/10.3390/s24020580

[23] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples. ICLR 2015. arXiv:1412.6572

[24] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks. ICLR 2018. arXiv:1706.06083

[25] D. Li, Q. Li, Adversarial deep ensemble: evasion attacks and defenses for malware detection. IEEE Trans. Inf. Forensics Secur. 16, 1284–1299 (2021). https://doi.org/10.1109/TIFS.2020.3003571

[26] D. Li, S. Cui, Y. Li, J. Xu, F. Xiao, S. Xu, PAD: towards principled adversarial malware detection against evasion attacks. IEEE Trans. Dependable Secur. Comput. 21(2), 1139–1153 (2024). https://doi.org/10.1109/TDSC.2023.3265330

[27] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, J. Keshet, Adversarial examples on discrete sequences for beating whole-binary malware detection. arXiv:1801.02950 (2018).

[28] I. Debicha, B. Cochez, T. Kenaza, T. Debatty, J.-M. Dricot, W. Mees, Adv-Bot: realistic adversarial botnet attacks against network intrusion detection systems. Comput. Secur. 129, 103176 (2023). https://doi.org/10.1016/j.cose.2023.103176

[29] M. Andriushchenko, F. Croce, N. Flammarion, M. Hein, Square attack: a query-efficient black-box adversarial attack via random search. ECCV, 484–501 (2020). https://doi.org/10.1007/978-3-030-58592-1_29

[30] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions. NeurIPS 30, 4765–4774 (2017).

[31] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": explaining the predictions of any classifier. Proc. 22nd ACM SIGKDD, 1135–1144 (2016). https://doi.org/10.1145/2939672.2939778

[32] T.T.H. Le, H. Kim, H. Kang, H. Kim, Classification and explanation for intrusion detection system based on ensemble trees and SHAP method. Sensors 22(3), 1154 (2022). https://doi.org/10.3390/s22031154

[33] O. Arreche, M. Abdallah, A comparative analysis of DNN-based white-box explainable AI methods in network security. J. Inf. Secur. 2025, 16 (2025). https://doi.org/10.1186/s13635-025-00201-x

[34] C.E. Ben Ncir, M.A. Ben HajKacem, M. Alattas, Enhancing intrusion detection performance using explainable ensemble deep learning. PeerJ Comput. Sci. 10, e2289 (2024). https://doi.org/10.7717/peerj-cs.2289

[35] N. Moustafa, J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems. IEEE MilCIS (2015). https://doi.org/10.1109/MilCIS.2015.7348942

[36] N. Moustafa, J. Slay, The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set. Inf. Secur. J. Glob. Perspect. 25(1–3), 18–31 (2016). https://doi.org/10.1080/19393555.2015.1125974

[37] V.N. Vapnik, The Nature of Statistical Learning Theory. Springer, New York (1995). https://doi.org/10.1007/978-1-4757-2440-0

[38] S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. 9(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735

[39] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation. Proc. EMNLP, 1724–1734 (2014). arXiv:1406.1078

[40] L. Breiman, Random forests. Mach. Learn. 45(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324

[41] T. Chen, C. Guestrin, XGBoost: a scalable tree boosting system. Proc. 22nd ACM SIGKDD, 785–794 (2016). https://doi.org/10.1145/2939672.2939785

[42] B.W. Matthews, Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim. Biophys. Acta 405(2), 442–451 (1975). https://doi.org/10.1016/0005-2795(75)90109-9

[43] A. Athalye, N. Carlini, D. Wagner, Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples. Proc. ICML, 274–283 (2018). arXiv:1802.00420

[44] J. Uesato, B. O'Donoghue, P. Kohli, A. van den Oord, Adversarial risk and the dangers of evaluating against weak attacks. Proc. ICML, 5025–5034 (2018). arXiv:1802.05666

[45] J.M. Cohen, E. Rosenfeld, Z. Kolter, Certified adversarial robustness via randomized smoothing. Proc. ICML, 1310–1320 (2019). arXiv:1902.02918

[46] Y.I. Alzoubi, A. Mishra, Differential privacy and artificial intelligence: potentials, challenges, and future avenues. J. Inf. Secur. 2025, 18 (2025). https://doi.org/10.1186/s13635-025-00203-9

[47] A.H.H. Kabla, M. Anbar, S. Manickam, S. Al-Sarawi, Peer-to-peer botnets: exploring behavioural characteristics and machine/deep learning-based detection. J. Inf. Secur. 2024, 20 (2024). https://doi.org/10.1186/s13635-024-00169-0

[48] Md.A. Hossain, Deep learning-based intrusion detection for IoT networks: a scalable and efficient approach. J. Inf. Secur. 2025, 28 (2025). https://doi.org/10.1186/s13635-025-00202-w

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need. NeurIPS 30 (2017).

[50] X. Liu, M. Liu, Z. Wang, Y. Wang, Explainable deep learning-based feature selection and intrusion detection method for IoT. Sensors 24(16), 5223 (2024). https://doi.org/10.3390/s24165223

[51] C.E. Asry, I. Benchaji, S. Douzi, B.E.L. Ouahidi, A robust intrusion detection system based on a shallow learning model and feature extraction techniques. PLOS ONE 19(1), e0295801 (2024). https://doi.org/10.1371/journal.pone.0295801

[52] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift. Proc. ICML, 448–456 (2015). arXiv:1502.03167

[53] Committee on Publication Ethics (COPE), COPE Position Statement on AI and Authorship, 2023. https://publicationethics.org/cope-position-statements/ai-author (accessed 1 May 2026).