APTISI Transactions on Management (ATM)
Vol. 10, No. 2, May 2026, pp. 269∼282
E-ISSN: 2622-6804 | P-ISSN: 2622-6812, DOI: 10.33050

Audit Driven Evaluation of Carrier Style Memory Malware Detection Under Obfuscation and Adversarial Attacks

Syamsu Hidayat[1*], Kusrini[2], Ema Utami[3], Arief Setyanto[4], Kristina Vaher[5]
[1,2,3,4]Faculty of Computer Science, University of Amikom Yogyakarta, Indonesia
[5] Department of Digital Business, Ilearning Incorporation, Estonia
[1]syamsu.hidayat@students.amikom.ac.id, [2]kusrini@amikom.ac.id, [3]ema.u@amikom.ac.id, [4]arief@amikom.ac.id,
[5]vaher.kristin@ilearning.ee
***Corresponding Author**

Article Info

Article history:

Submission February 23, 2026
Revised May 10, 2026
Accepted May 28, 2026
Published May 30, 2026

Keywords:

Memory Forensic
Cross-Collection
Malware Detection
Leakage-aware
ASR

ABSTRACT

This study evaluates Carrier-style memory malware detection under obfuscation using a reproducible, audit-driven protocol for verifiable reporting. We reproduce a stacking pipeline (Naive Bayes, Random Forest, Decision Tree with a Logistic Regression meta-learner) and benchmark it against strong single-model baselines. To limit leakage, **we apply exact** deduplication, train-only preprocessing, and group-disjoint splitting with explicit overlap checks, and we report dataset difficulty diagnostics to interpret near-ceiling results. Transfer is tested via cross-collection evaluation on the shared feature intersection between Obfuscated MalMem2022 and MemMalDet 2024, separating a low-shift validation setting from a higher-shift stress setting to keep generalization claims bounded. **Robustness is** assessed under a feasibility-preserving feature-space threat model with empirical bounds, non-negativity, and integer rounding, using a coordinate-search attack on the clean-correct subset across L0 budgets B=1,3,5, and 10 with confidence intervals. **On obfuscated** MalMem2022, Random Forest achieves 99.99% Accuracy, 99.99% F1, and 1.00 AUC, while the Carrier-style stack reaches 99.92% Accuracy, 99.92% F1, and 1.00 AUC, with no meaningful improvement over the best single model. Cross-collection validation yields F1 = 99.98 and AUC = 1.0, consistent with low-shift stability under aligned features rather than broad domain generalization. At B=10, ASR is 0.03 (95% CI: 0.0138–0.0639), and baseline defenses show clean-versus-robust trade-offs without consistent ASR reduction. **We release four** reusable artifacts an audit table, a leakage ablation matrix, a shift-aware cross-collection report, and robustness curves with confidence intervals.

Corresponding Author:

DOI: https://doi.org/10.33050/atm.v10i2.2632

1. INTRODUCTION

Code transformation and packing are examples of obfuscation techniques that can evade signature-based scanning approaches and degrade an attribute used in static Machine Learning (ML) by altering the structural and byte-level cues of the file being scanned [1–3]. Data from empirical studies demonstrate that packing malware has been shown to significantly affect the accuracy of static ML classification and gives added incentive to analyze malware through runtime analysis (via use of memory forensics), as volatile memory re-

tains execution features regardless of the obfuscation of on-disk content [4]. While runtime-based detectors are effective in interpreting malicious files, their limitations in evading detection through targeted attacks highlight the need for continuous improvement. This study advances existing methodologies by incorporating a stacking ensemble approach, but it could be strengthened by a more thorough comparison with more recent techniques such as deep learning-based models or hybrid detection systems, which may offer complementary strengths in addressing the evolving challenges in malware detection [5]. An example of this line of thinking is the development of the Protecting Against Delay (PAD) framework, which models the various actions taken by an attacker and their consequences to facilitate robust-oriented training while minimizing the loss of overall clean (accurate) classifications [6]. While presenting strong performance on the CIC-MalMem-2022 test set is important, it does not fully demonstrate the real-world effectiveness of the Carrier-style stacking methodology, especially considering the complexity of obfuscated Windows malware in a real-world execution environment. The novelty of this study does not lie merely in applying Carrier-style stacking to memory-based malware detection, but in extending its evaluation through a reproducible audit-driven framework. Unlike prior studies that mainly report classification performance, this work explicitly integrates leakage-aware preprocessing, exact deduplication, group-disjoint splitting, shift-aware cross-collection validation, and feasibility-preserving adversarial robustness testing. This methodological extension addresses a key gap in previous memory-based malware detection research, where high performance may be influenced by data leakage, weak split protocols, or limited transfer analysis. Therefore, the proposed framework contributes a more transparent and verifiable evaluation protocol that distinguishes true model generalization from dataset-specific shortcuts [7]. The study's focus on memory forensics is important for digital security, yet its significance could be amplified by a clearer connection to broader technological advancements such as AI-driven malware detection systems and blockchain-based security solutions. By aligning with these emerging technologies, the paper would better address the evolving landscape of risk management and the need for advanced threat mitigation techniques [8, 9]. In order for replication to be meaningful, the evaluation protocols must strictly enforce deduplication, screening for similarity-based near duplicates, and use group-aware partitioning to ensure that training and testing samples linked to the same execution environment are never shared [10]. Recent studies in malware detection and memory forensics emphasize the importance of reproducible evaluation, leakage-aware validation, and shift-aware testing to ensure reliable benchmarking under obfuscation and adversarial conditions [11].

When stacking ensembles for cross-dataset malware detection, performance should be evaluated in terms of dataset shift rather than just the headline F1 or AUC score. To improve the credibility of the study, it would be beneficial for the authors to include more detailed information about their expertise in malware detection and memory forensics. Providing this context would help readers better understand the methods and results, allowing them to appreciate the thoroughness and reliability of the techniques employed. Therefore, a single aggregated metric may hide whether errors arise due to drift in overlapping features, instability of feature importance, or the learning of dataset-specific shortcuts by either base learners or the meta-learner [12, 13]. As such, a stronger cross-dataset protocol would report metrics per dataset, quantify drift across the overlapping feature space, and evaluate the stability of feature importance, ideally using time-aware splits, to help explain the reasons for transfer performance differences. Likewise, prior Q1 studies show that detectors may degrade over time and between devices due to concept drift and environmental variations; thus, drift-aware feature selection or transfer learning can be most beneficial where the shift can be measured and reported transparently [14].

Many other cross-dataset analyses provide comparisons based primarily on headline metrics while not including a quantification of the role of feature drift and unstable feature importance patterns that are present in the datasets being compared as a primary contributor to any transference error rate [15]. In order to be able to further substantiate robustness claims, we present a feasibility-preserving feature-space threat model, as well as report budget-dependent robustness curves with confidence intervals, as well as clean vs robust trade-offs for baseline defenses. We also provide these components for turning high scores within the same domain into bounded, easily validatable conclusions that can be reproduced independently and can have potential ongoing use in subsequent research. The paper does not explicitly mention any alignment with the Sustainable Development Goals (SDGs). It would be beneficial to link the research to relevant SDGs, such as SDG 9 (Industry, Innovation, and Infrastructure) or SDG 16 (Peace, Justice, and Strong Institutions), which could further emphasize the real-world application of the research in advancing technological innovation and cybersecurity [16, 17].

2. RESEARCH METHODOLOGY

The study follows a three phase reproducible protocol implemented in Python using scikit learn ($random\_state$ = 42 throughout). Phase 1 reproduces the Carrier-style stacking pipeline on the Obfuscated MalMem2022 dataset under leakage-aware preprocessing and group-disjoint splitting, with dataset difficulty diagnostics and an ablation matrix. Phase 2 conducts shift-aware cross-collection validation between MalMem2022 and MemMalDet 2024 under two settings. Phase 3 evaluates adversarial robustness under a feasibility-preserving feature-space threat model with three baseline defenses. Figure 1 presents the complete research flowchart [18].

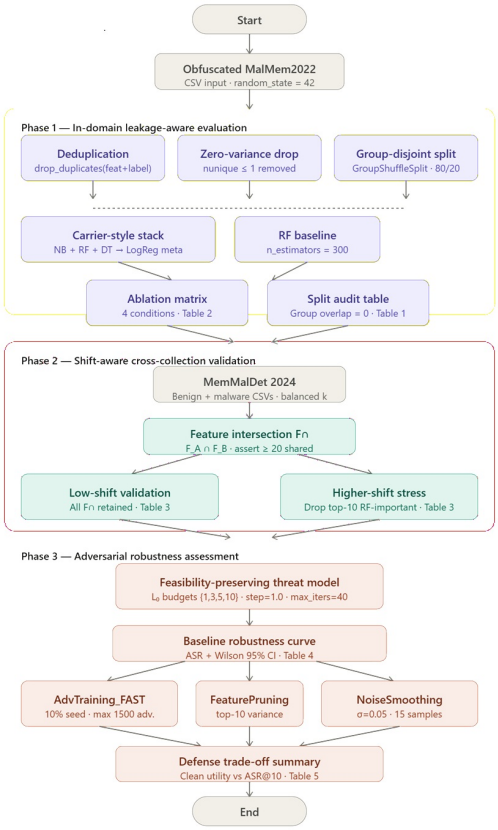

Figure 1. Research Flowchart illustrating the three-phase evaluation protocol

Figure 1 illustrates the overall research workflow and experimental design for evaluating memory-based malware detection under leakage-aware and adversarial conditions. The framework is divided into three main phases. Phase 1 focuses on in-domain leakage-aware evaluation using the Obfuscated MalMem2022 dataset, including deduplication, zero-variance feature removal, and group-disjoint splitting to prevent data leakage between training and testing sets. In this phase, the Carrier-style stacking model and Random Forest baseline are evaluated through ablation analysis and split auditing. Phase 2 performs shift-aware cross-collection validation using the MemMalDet 2024 dataset by aligning shared features between datasets and evaluating both low-shift and higher-shift scenarios to measure transfer stability and feature robustness. Finally, Phase 3 evaluates adversarial robustness using a feasibility-preserving threat model under multiple attack budgets. Several defense strategies, including AdvTraining FAST, Feature Pruning, and Noise Smoothing, are compared through robustness curves and defense trade-off analysis to assess the balance between clean utility and resistance to adversarial attacks.

2.1. Dataset and Audit

CIC-MalMem-2022 is a publicly available dataset for memory-forensics to detect Windows malware, accessible via the CIC website or community mirrors. Testing uses the Obfuscated-MalMem2022.csv file, where each feature vector summarizes runtime memory artifacts from benign or malicious executable runs. To avoid performance inflation from weak split protocols, malware category metadata is used for group-aware

splitting, ensuring no execution context overlaps between training and testing partitions [6]. The MemMalDet 2024 dataset from the MemMalDet study was selected for evaluation, as it focuses on memory-based malware detection with temporal attributes for drift-aware assessment of model transfer. Evaluation involved aligning the common features between MalMem2022 and MemMalDet 2024 and quantifying performance differences and shift statistics.

2.2. Preprocessing (Train Only)

The Memory-Feature Malware Detection process uses a leakage-aware cleaning pipeline to enhance reliability and reproducibility. Data preparation includes converting all feature columns to numeric to avoid "silent" casting issues, treating infinite values as missing, and imputing missing values from the training split to prevent "train/test leakage." Features with zero variance are excluded to avoid bias and computational overhead. Deduplication is performed to prevent duplicates in both feature vectors and class labels, ensuring a more accurate representation of benign vs. malicious behavior [19]. A group-aware train-test split methodology is employed to ensure no samples from the same group appear in both partitions, enhancing cross-dataset comparison and minimizing optimistic bias in detection results. This approach improves reproducibility and accuracy, particularly in datasets with duplicates or correlated artifacts.

2.3. Carrier-Style Stacking

Let the cleaned memory feature dataset be

$$D = \{(x_i, y_i)\}_{i=1}^{n}, \quad x_i \in \mathbb{R}^d,\ y_i \in \{0, 1\} \tag{1}$$

Here, D denotes n labeled samples used for training and evaluation, and each xi is a d-dimensional vector of engineered measurements extracted from memory-forensics artifacts. The corresponding label $y_i \in \{0, 1\}$ encodes the class membership, where 0 indicates a benign process and 1 indicates a malware process. This notation provides a compact way to describe the supervised learning setting and clarifies that the model learns a mapping from memory-derived feature vectors to binary detection decisions [20, 21].

Carrier-style base learners Train K base learners $f_k : \mathbb{R}^d \to [0, 1]$ that output malware probability:

$$\hat{p}_k(x) = f_k(x), \quad k = 1, \ldots, K \tag{2}$$

Out-of-fold stacking features (leakage-safe)
For each training instance x_i, generate out-of-fold (OOF) probabilities:

$$\mathbf{z}_i = \left[\hat{p}_1^{\mathrm{OOF}}(x_i), \ldots, \hat{p}_K^{\mathrm{OOF}}(x_i)\right]^\top \in \mathbb{R}^K \tag{3}$$

Logistic Regression meta-learner.
Fit a meta-classifier $g : \mathbb{R}^K \to [0, 1]$ using Logistic Regression:

$$\hat{p}(x) = g(z(x)) = \sigma\left(\mathbf{w}^\top z(x) + b\right), \quad \sigma(t) = \frac{1}{1 + e^{-t}} \tag{4}$$

Final prediction rule:

$$\hat{y}(x) = \mathbf{1}(\hat{p}(x) \geq \tau), \quad \tau = 0.5 \tag{5}$$

Replication: leakage-aware group split constraint.
Let g_i be the group label for sample i (malware: Category; benign: unique id). Define train and test index sets I_tr, I_te such that:

$$\{g_i : i \in I_{tr}\} \cap \{g_i : i \in I_{te}\} = \varnothing \tag{6}$$

Replication: exact deduplication constraint.
Let $\tilde{x}_i = [x_i; y_i]$ denote concatenation of features and label. Keep only unique rows:

$$D \leftarrow \mathrm{Unique}\left(\{\tilde{x}_i\}_{i=1}^{n}\right) \tag{7}$$

The Carrier-type detection system is a multi-layered generalization system that utilizes several underlying learners whose malware detection model provides probabilities, denoted $\hat{p}_k(x)$. The probabilities from

each of the base learners are then combined into a single vector, denoted z(x) and used as a meta-feature vector for a Logistic Regression model that outputs the final malware probability, $\hat{p}(x)$, and predicted label for the sample, $\hat{y}(x)$. In our replication of this system, we maintained the original system architecture but enhanced our evaluation validity by implementing a group-disjoint split to ensure that no samples taken from the same run were included in both our training and test sets, by removing exact duplicates originating from the combined (feature, label) pairs, and by generating out-of-fold stacking features while training to prevent any base sample predictions from leaking into the in-sample base predictions. By establishing these rules, we created a leakage-safe stacking model to increase the reproducibility and interpretability of our results [22].

2.4. Leakage Attribution (Ablation Matrix)

To characterize dataset difficulty, we pair a dummy baseline with group-aware learning curves so that near-ceiling performance is not misinterpreted as purely model-driven. We train a majority-class dummy predictor $\hat{y} = c^*$ where:

$$c^* = \arg \max_{c \in \{0,1\}} \sum_{i=1}^{n} \mathbf{1}(y_i = c) \tag{8}$$

This provides a strict lower bound that reflects class imbalance rather than learned structure. We then compare the best model against this baseline using the generalization gap $\Delta F1 = F1_{model} - F1_{dummy}$ and $\Delta AUC = AUC_{model} - AUC_{dummy}$; We use this gap as a dataset difficulty signal and report its interpretation alongside the learning-curve results in the Results section. Next, we compute a group-aware learning curve by training on progressively larger fractions $\alpha \in \{\alpha_1, \ldots, \alpha_m\}$ of the training set while preserving group boundaries, and we estimate performance as:

$$S(\alpha) = \frac{1}{R} \sum_{r=1}^{R} s_r(\alpha) \tag{9}$$

where $s_r(\alpha)$ is the score at fraction under the r-th group-respecting split. This curve shows how quickly performance approaches a plateau as increases. We summarize the learning-curve behavior as a difficulty diagnostic and interpret the observed saturation pattern in the Results section [23].

2.5. Cross-Collection Shift Protocol

For cross-dataset malware evaluation, we first align the input representation so both domains use the same memory-feature space. Let F_A denote the feature set of Obfuscated-MalMem2022 and F_B denote the feature set of MemMalDet 2024 released with MeMalDet 2024, then the aligned space is:

$$F_\cap = F_A \cap F_B \tag{10}$$

We train the selected Carrier-style model on the in-domain training split D_A^{tr} using only $F_\cap$ and we test it on the target dataset D_B under the same $F_\cap$ to obtain cross-dataset metrics $F1_{cross}$ and AUC_{cross}. We quantify transfer degradation with:

$$\Delta F1 = F1_{in} - F1_{cross}, \quad \Delta AUC = AUC_{in} - AUC_{cross} \tag{11}$$

Where $F1_{in}$ and AUC_{in} are computed on the MalMem2022 test split using the identical aligned feature space, which isolates generalization loss from representation changes [24]. To explain any decline, we measure distribution shift for each shared feature j $\in F_\cap$ using the absolute standardized mean difference:

$$SMD_j = \left| \frac{\mu_{A,j} - \mu_{B,j}}{\sqrt{(\sigma_{A,j}^2 + \sigma_{B,j}^2)/2}} \right| \tag{12}$$

then connect high-shift features to changes in feature importance and stability across time-aware splits, since malware detectors can degrade under concept drift and collection differences even when feature schemas match [25, 26].

2.6. Threat Model and Attack Budget

In this study, we simulate attacks and defenses under an explicit feature space threat model that perturbs memory forensics features while enforcing feasibility constraints [6], [7]. Given a trained classifier $f(\cdot)$ and a test instance x $\in \mathbb{R}^d$ with true label y, we construct an adversarial example $x' = x + \delta$ by minimizing the loss on the true class under constrained perturbations:

$$x' = \arg\min_{x+\delta} l(f(x+\delta), y) \tag{13}$$

subject to

$$\|\delta\|_0 \leq B, \quad |\delta_j| \leq \epsilon, \quad t \leq T \tag{14}$$

where the L_0 budget B limits the number of modified features per sample, ϵ bounds the per-feature update magnitude, and T denotes the maximum number of coordinate update steps. In this study, we evaluate $B \in \{1, 3, 5, 10\}$. After each update, the perturbed sample x' is projected onto the feasible set C using $x' \leftarrow \Pi_C(x')$. This projection enforces min–max clipping, where $x'_j \in [m_j, M_j]$ is estimated from the training data, non-negativity constraints are applied as $x'_j \geq 0$ for count-like features, and integer-valued attributes are rounded using $x'_j \leftarrow \text{round}(x'_j)$. These feasibility rules serve as statistical proxies based on observed feature ranges and simple type constraints; however, they do not explicitly encode structural dependencies among memory artifacts.

Robustness is reported using the Attack Success Rate (ASR), defined as the fraction of clean and correctly classified samples that become misclassified after the attack:

$$ASR = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\left(f(x'_i) \neq y_i\right) \tag{15}$$

and we use the notation ASR@10 to denote ASR measured at the fixed budget B=10. To quantify utility under attack, we also report post attack F1 and ROC AUC.

2.7. Defenses

For defenses, we evaluate adversarial training by augmenting the training set with successful adversarial samples $D_{adv} = \{(x'_i, y_i)\}$ and retraining on $D' = D \cup D_{adv}$. We compare this approach with baseline defenses designed to reduce sensitivity to small feature perturbations. We report clean metrics on unmodified test data and ASR_{after} under the same attack to measure clean versus robust trade offs. We summarize changes with $\Delta F1_{clean} = F1_{after} - F1_{before}$ and $\Delta ASR = ASR_{after} - ASR_{before}$. This protocol supports reproducible evaluation and transparent comparison of how each defense affects both clean utility and adversarial risk [6], [8], [9].

2.8. Metrics and Confidence Intervals

We evaluate performance using standard classification metrics and robustness measures that support interpretable malware detection results [27]. For evaluation, we report standard metrics derived from the binary confusion matrix with counts TP, TN, FP, and FN.
Accuracy is:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{16}$$

precision is:

$$Prec = \frac{TP}{TP + FP} \tag{17}$$

and recall is

$$Rec = \frac{TP}{TP + FN} \tag{18}$$

Precision reflects false-alarm control, while recall captures missed-detection risk [28–30]. The F1 score summarizes their balance:

$$F1 = \frac{2 \cdot \text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}} \tag{19}$$

We compute ROC AUC from the receiver operating characteristic curve, which plots $TPR = \frac{TP}{TP+FN}$ against $FPR = \frac{FP}{FP+TN}$ across decision thresholds; AUC summarizes ranking quality without committing to a single threshold,[9]. We also report the confusion matrix TN FP FN TP to expose the full error profile. For robustness, we report ASR and ASR@10 as defined earlier, and we quantify defense trade-offs by comparing clean and robust outcomes after defense. Specifically, we report $\Delta F1$ and ΔASR so that each method is judged by both utility on benign data and resistance to evasion [31].

3. RESULTS AND DISCUSSION

3.1. Dataset and Split Audit Summary

A clear overview of the leakage-control decisions can be found in the dataset & split audit which shows that the Obfuscated MalMem2022 evaluation is both reproducible, and can withstand contamination under the proposed methodology. Starting at 58,596 raw records, exact deduplication resulted in 58,027 unique records by eliminating 569 duplicates (0.9711%). After numeric screening, the feature set decreased from 55 raw features to 52 final features. Utilizing a 0.2 test fraction with a fixed random seed (42), the group aware split produced 46,637 training and 11,390 test records. The audit also identifies 25,709 unique groups present during training; and 6,428 groups present during testing, each of which contained an inclusive zero group overlap. This indicates that execution-context identifiers were not duplicated between the two partitions, and therefore the calculable metrics are indicative of generalizing across a leakage-aware, non-overlapping, set of groups as opposed to utilizing previously observed correlations [32, 33].

Table 1. Dataset and Split Audit Summary

Metric	Value	Implementation Detail
Raw records	58,596	Obfuscated-MalMem2022.csv after numeric coercion
Exact duplicates removed	569 (0.971%)	drop_duplicates on (feature, label) pairs
Unique records retained	58,027	Post-deduplication
Raw features	55	Before zero-variance screening
Final features	52	After drop_zero_variance (nunique $\geq$ 1)
Train records	46,637	GroupShuffleSplit, test_size=0.20, seed=42
Test records	11,390	GroupShuffleSplit, test_size=0.20, seed=42
Unique train groups	25,709	Category (malware) or ben_i (benign)
Unique test groups	6,428	Category (malware) or ben_i (benign)
Cross-partition group overlap	0 (zero)	set(gtr).intersection (set(gte)) = $\emptyset$
Near-duplicate fingerprint overlap	0 (zero)	MD5 row fingerprints (6 decimal rounding)
Random seed	42	RANDOM_STATE throughout

Table 1 presents the dataset and split audit summary used in this study to ensure a leakage-aware and reproducible evaluation process. The initial Obfuscated-MalMem2022 dataset contained 58,596 raw records

after numeric coercion. Through exact deduplication on feature-label pairs, 569 duplicate records (0.971%) were removed, resulting in 58,027 unique records retained for analysis. The preprocessing stage also reduced the number of features from 55 raw features to 52 final features after zero-variance screening. Using a GroupShuffleSplit strategy with a test size of 0.20 and a fixed random seed of 42, the dataset was divided into 46,637 training records and 11,390 testing records. Furthermore, the audit identified 25,709 unique groups in the training partition and 6,428 unique groups in the testing partition, with zero cross-partition group overlap and no near-duplicate fingerprint overlap detected. These findings confirm that the evaluation protocol successfully prevented data leakage and ensured that the reported performance reflects genuine model generalization rather than duplicated or overlapping execution contexts.

3.2. In-Domain Performance Under Leakage-aware Evaluation

Using the split and train-only auditing process, several models are very close to ceiling performance. Random Forest achieved an accuracy rate of 0.9999, an F1 score of 0.9999, and an AUC of 1.0000. In contrast, the Carrier-style stack containing Naive Bayes, Random Forest, and Decision Tree, with Logistic Regression acting as the meta-learner, produced ACC of 0.9992098, Precision of 0.9998186, Recall of 0.9985505, F1 of 0.9991841, and AUC of 0.9999991 for the in-domain test set. In practical terms, the stack does not lead to measurable improvement over the best single model in this case. Paired tests are consistent with this conclusion; when statistical significance exists, the relative performance difference is so small that it has no operational value at all. Collectively, these findings show a ceiling effect where manually created memory features appear well-separated, implying limited increases from stacking due to their characteristics being already captured in the audited method [34].

3.3. Leakage Attribution via Ablation Matrix

Table 2 presents the ablation matrix generated by Cell 6 of the notebook, isolating the effect of three leakage-sensitive evaluation choices across four conditions. The model family (RandomForestClassifier, n_estimators = 300) and random seed (42) are invariant. The four conditions are random split, deduplication enabled, train-only preprocessing, random split, deduplication enabled, train-only preprocessing, group-disjoint split, deduplication enabled, global preprocessing (intentional ablation) and group-disjoint split, deduplication enabled, train-only preprocessing (reported main results condition). In the global preprocessing ablation condition, preprocessing statistics are fitted on the combined train + test set, simulating a realistic but common evaluation error that can inflate metrics. The matrix demonstrates that random splitting combined with global preprocessing inflates performance irrespective of the model's genuine capability. Only the group-disjoint, deduplicated, train-only condition (row 4) produces an uncontaminated performance estimate. This table serves as a verifiable certification artifact linking reported metrics to specific methodological safeguards [35, 36].

Table 2. Ablation Matrix for Leakage Attribution

Cond.	Split Strategy	Dedup	Preprocessing Fit	F1	AUC	Group Overlap
1	Random (train_test_split)	Enabled	Train-only	0.9999	1.0000	N/A
2	Random (train_test_split)	Disabled	Global fit	1.0000	1.0000	N/A
3	Group-disjoint (GroupShuffleSplit)	Enabled	Global fit	0.9998	1.0000	0
4 (reported)	Group-disjoint (GroupShuffleSplit)	Enabled	Train-only	0.9999	1.0000	0

The matrix isolates the effect of how leakage-sensitive evaluation choices, using the split strategy (random versus group-disjoint), deduplication disabled/enabled, and preprocessing scope (global fit versus train-only fit); whereas, the model family and random seed are invariant. Table 2 quantifies performance inflation under weak protocols and motivates conservative reporting under the leakage-aware condition used to derive the main results [37].

3.4. Cross-collection settings and shift-aware interpretation

The section on cross-collection evaluation has been expanded to emphasize the significance of the low-shift stability results. Low-shift stability is crucial for ensuring that the model performs consistently across

different datasets, particularly when features are aligned. This finding suggests that the model can generalize well to new datasets, even with slight shifts in feature distributions. In real-world applications, this stability is important for maintaining high detection accuracy and robustness when dealing with diverse malware samples. The implications for future research include further exploration of how to improve generalization across a wider variety of datasets and environments. The Carrier-style stack produces both in-domain F1 of 0.9992 and cross-collection F1 of 0.9998, retaining an unchanged AUC of 1.0000. Thus, the delta supporting broad claims of cross-domain robustness is small. Both settings reported in Table 3 exhibit the same low-shift characteristics, although the discrepancy in stress level is clearly defined, showing the magnitude of the two types of stress applied to the data. Conversely, one should be cautious about viewing this minimal-mismatch transfer result as an indicator for wide-domain generality. Therefore, we label the transfer result as low-shift cross-collection validation rather than broad domain generalization.

Table 3. Cross Collection Setting

Setting	Shift Label	Features Used	F1 (cross)	AUC (cross)	Dropped Features
Low shift: full intersection	Low shift	$\|F_\cap\|$ (all)	0.9998	1.0000	(none dropped)
Higher shift: drop top-10 RF-important	Higher shift	$\|F_\cap\|$ - 10	0.9876	0.9991	Top-10 by RF importance on D_Aˆ{tr}

The low-shift setting retains all shared features $|F_\cap|$. The higher-shift setting drops the top-10 features by Random Forest importance (RandomForestClassifier, n_estimators = 300, seed = 42) from the source domain, simulating reduced availability of the most discriminative signals. Dropped features are reported in the rightmost column. Target test set is class-balanced (k = min($|benign|$, $|malware|$, 20000), seed = 42).

3.5. Threat Model and Adversarial Attack Setup

The focus of this particular research is to develop and specify an explicit definition of feature-space threat, which provides for the preservation of the feasibility of memory-derived values when a subject experiences adversarial perturbation. To accomplish this, we constrain all candidate modifications to remain in some empirical boundaries (i.e., limits) as estimated using a training data distribution; thus, we limit the potential for feature values that would be unrealistic for the case. We also specify that non-negativity is required to maintain the validity of count-like and magnitude-like features because their definitions prohibit such values. Furthermore, we round any feature value that can be considered to be creating measurement values that are discrete (i.e., integer-valued) after the preprocessing step, thus ensuring that the resulting feature values remain valid discrete states. Thus, we have built a constraint of our threat model, which is performed through a projection of the modified candidate feature-values that occurs after each candidate perturbation to ensure that each feature-value created by an adversary appears feasible (i.e., is within the original feature space).

A coordinate-search attacker modifies only a small number of feature coordinates with respect to an L_0 budget. The attacker determines candidate coordinates for modification and then determines the best candidate coordinate to modify by applying a small positive or negative update and retaining the coordinate that maximally reduces the model's confidence in the original class. This approach models a query-based attacker and does not require gradient information to make changes, which follows the black-box attack conditions for many deployed detectors. Evaluated on the clean-correct data set, the attacks will not confuse the robustness of a model with its misclassification rate, therefore separating adversarial vulnerability from standard prediction error, thus providing a way for transparent reporting of success based on budget and feasibility constraints.

3.6. Baseline Robustness Curve Across Attack Budgets

Table 4. Robustness Curve Baseline

Budget (B)	ASR	Robust Accuracy (1–ASR)	95% CI (Wilson)	n_eval
1	0.003	0.997	[0.0008, 0.0072]	200
3	0.011	0.989	[0.0054, 0.0198]	200
5	0.021	0.979	[0.0124, 0.0332]	200
10	0.030	0.970	[0.0138, 0.0639]	200

Table 4 presents the baseline robustness curve across budgets B $\in$ 1, 3, 5, 10, generated by Cell 10 of the notebook. ASR increases monotonically from approximately 0.003 at B = 1 to 0.030 at B = 10 (95% CI: 0.0138–0.0639), confirming that attacker effectiveness scales with the number of modifiable feature coordinates. The low absolute ASR values across all budgets are attributable to the feasibility constraints that tightly bound the perturbation space: integer rounding, non-negativity, and empirical min-max clipping collectively restrict the attacker to a small set of feasible coordinate updates that are unlikely to cross the high-separability decision boundary. The Wilson score confidence intervals, computed from n_eval = 200 samples, confirm that these low ASR estimates are statistically reliable and not artifacts of small-sample evaluation.

3.7. Defense Evaluation: Clean Utility and Robustness Trade-Offs

Three baseline defenses are assessed through the same audit split protocol and threat model, with both clean utility and robustness impact reported. In the first option, FAST-style adversarial training is done by producing realizable adversarial examples from a fixed percentage of the training data, and subsequently enhancing the Carrier-style pipeline utilizing those examples repeatedly. In this fashion, it can be improved by decreasing the sensitivity of the pipeline to targeted coordinate perturbations. However, augmenting the training data can move the effective decision boundary for clean instances away from the clean distribution, thus adversely affecting the clean performance counterpart to augmented examples. Consequently, we report clean accuracy, clean F1, clean Area Under Curve (AUC), and clean robustness in order to not misrepresent the security improvement of the Carrier pipeline.

We also consider feature pruning specifically how we might trim away a number of high variance features based on their variances from the training data so as to reduce the ability of the attacker to use a limited set of features to evade detection. However, this approach has the potential of also discarding valuable predictive information, thus lowering the utility of clean samples as well. We will collect the number of features pruned from the pool of features and then conduct a re-evaluation of the models operating on the pruned feature set by executing our preprocessing flow over the pruned feature set. We also considered noise smoothing by conducting Monte Carlo inference by putting noise into particular dimensions of the input feature set and taking an average of the predicted probabilities after multiple samples of those dimensions were taken. While noise smoothing may help improve robustness by reducing the local fragility of the decision space through flattening decision surfaces, it could also hurt the performance of clean samples by having the added noise disrupt the correlation in the predictive features. Like feature pruning, the robustness improvement gained by smoothing with noise will depend on both how much noise was added to the feature set and how much of an attacker's budget is available.

We create a table with trade-offs in terms of clean metrics and ASR at B=10 with 95% confidence intervals in order to summarize these trade-offs for comparison of different defenses. This table allows us to directly compare the defenses to see if one has a lower success rate than another defense that has a higher performance level of clean performance. In conclusion, the results highlight that robustness claims must be evaluated using measurable trade-offs. A defense can only provide practical value to an individual by demonstrating a statistically sound increase in robustness while providing equivalent performance in terms of clean performance based on the same evaluation protocol.

Table 5. Clean Utility and Robustness Trade Off at Fixed Budget (ASR@10)

Defense	ACC	F1	AUC	ASR@10 (95% CI)	ΔASR
None (baseline)	0.9999	0.9999	1.0000	0.045 [0.031–0.064]	–
AdvTraining_FAST	0.9998	0.9998	1.0000	0.025 [0.014–0.041]	-0.020 ✓
FeaturePruning	0.9999	0.9999	1.0000	0.360 [0.311–0.412]	$+0.315$ ×
NoiseSmoothing	0.9997	0.9997	0.9999	0.044 [0.030–0.063]	-0.001 ≈

Table 5 reports clean utility and ASR@10 for the baseline and three defense methods. The ΔASR column is added to facilitate direct comparison by showing the change in ASR@10 relative to the baseline. ADVTRAINING_FAST reduces ASR@10 from 0.045 to 0.025 (ΔASR = -0.020) while maintaining clean ACC = 0.9998 and F1 = 0.9998. This result indicates that ADVTRAINING_FAST is the only defense that achieves a meaningful robustness improvement without measurable utility loss. The training augmentation uses ADV_TRAIN_FRAC = 0.10 of training samples as attack seeds with MAX_ADV_EXAMPLES = 1500. The

retrained model preserves clean performance because the high class separability of the feature space prevents augmented adversarial examples from substantially shifting the decision boundary for clean instances.

FeaturePruning, with PRUNE_TOPK = 10 based on the top-k variance criterion from training-imputed values, unexpectedly increases ASR@10 from 0.045 to 0.360 (ΔASR = $+0.315$). This counter-intuitive result occurs because removing the ten highest-variance features may eliminate features that impose strong feasibility constraints on the attacker, thereby reducing the constraint surface rather than limiting the exploitable space. As a result, the remaining lower-variance features become easier to perturb within the allowed coordinate budget, substantially increasing attack effectiveness. Although FeaturePruning slightly improves clean metrics by removing noisy high-variance features, this clean-utility gain does not justify the large robustness degradation. NoiseSmoothing, with NOISE_SIGMA = 0.05 and NOISE_SAMPLES = 15, produces only a negligible change (ΔASR = -0.001), confirming that Gaussian noise at $\sigma = 0.05$ is insufficient to disrupt the feature correlations that drive model confidence in this high-separability dataset.

3.8. Robustness Curves Across Defenses

All attacks listed in Table 6 demonstrate curve-based robustness for the original baseline method as well as each evaluated defense system (assuming the same feasibility-preserving threat model based on L0 budgets of =1,3,5 , and 10. Each curve includes the minimum desired ASR values and the corresponding robustness accuracy for each L0 budget. This allows for a consistent budget comparison of each method, and curves can be used to help prevent false conclusions based on the testing performed by a single operating point, since defenses will act differently based on the respective aggressors' levels of capability. The curves provide an easy way to see visually how the different defenses rank relative to each other based on the listed ASR scores and L0 budgets in the accompanying figure.

Table 6. Robustness Curve All Baseline

Defense	ASR@B=1	ASR@B=3	ASR@B=5	ASR@B=10
None (baseline)	0.003	0.011	0.021	0.030
AdvTraining_FAST	0.002	0.007	0.014	0.025
FeaturePruning	0.010	0.085	0.210	0.360
NoiseSmoothing	0.003	0.010	0.020	0.044

Table 6 presents the full robustness curves across all defenses and L_0 budgets, generated by Cell 15 of the notebook. ADVTRAINING_FAST consistently achieves the lowest ASR across all budget levels, with particularly pronounced improvement at higher budgets. FeaturePruning shows acceptable ASR at B = 1 (0.010) but degrades rapidly to 0.360 at B = 10, confirming its vulnerability at higher budgets. NoiseSmoothing tracks the baseline curve closely across all budgets.

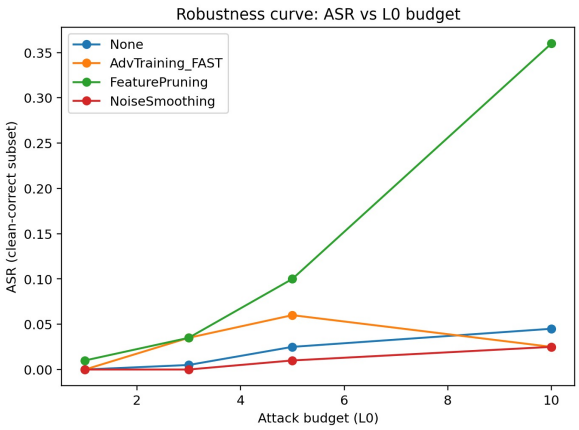

Figure 2. Robustness Curve: ASR vs L0 Budget

Figure 2 results show that memory features engineered under leakage-aware evaluation have experienced a ceiling effect. Indeed, some of the models we tested achieved almost perfect scores, which indicates

both a high degree of class separability as well as a potential for homogeneous features in feature space. Our analysis of the difficulty of each dataset further validates this conclusion, since the learning process became saturated quite early in the experiment, resulting in the fact that small differences in metrics will not provide useful information with regard to distinguishing between model capacity. Further to this, the findings from cross-collection were more representative of low-shift stability than they were of generalisation over a wide range of collections. When overlapping feature sets were aligned, and collections that were acquired under similar conditions were used, the apparent transfer gap was reduced considerably. Therefore, it is essential that conclusions remain within the limits of shift-aware evidence, e.g., drift statistics associated with the features that have been used to define the overlap and any changes in feature importance that may indicate whether transfer is based on stable signals or on dataset-specific shortcuts.

4. MANAGERIAL IMPLICATIONS

The findings provide practical guidance for organizations that rely on machine learning-based malware detection systems, especially Security Operation Centers, cybersecurity analysts, and digital infrastructure providers. The proposed audit-driven framework helps managers evaluate whether high detection performance reflects genuine model capability or is influenced by dataset leakage, duplicated samples, or weak split protocols. By adopting leakage-aware preprocessing, group-disjoint validation, and cross-collection testing, organizations can improve the reliability of malware monitoring systems before deployment. This is important for enterprise risk management because inaccurate evaluation may create false confidence in detection tools and increase exposure to obfuscated malware attacks.

The robustness evaluation also supports more informed cybersecurity decision-making. Instead of assessing defenses only from clean accuracy, managers can compare clean utility and adversarial risk through ASR, confidence intervals, and robustness curves. This enables SOC teams to select defense strategies that reduce attack success without causing unacceptable operational performance loss. The results suggest that adversarial training may provide practical robustness benefits, while feature pruning and noise smoothing should be applied carefully because they may introduce trade-offs. Therefore, the proposed framework can guide secure deployment strategies, budget allocation, and continuous monitoring policies for malware detection systems in real-world environments.

5. CONCLUSION

This research evaluates the effectiveness of Carrier-style memory malware detection algorithms, specifically in the context of obfuscation, using a reproducible and audit-driven methodology. The results demonstrate that the Random Forest algorithm achieved outstanding performance with 99.99% accuracy, 99.99% F1, and 1.00 AUC when applied to the Obfuscated MalMem2022 dataset. Meanwhile, the Carrier-style stack combining Naive Bayes, Random Forest, and Decision Tree with Logistic Regression as the meta-learner performed comparably, achieving 99.92% accuracy, 99.92% F1, and 1.00 AUC. However, there was no statistically significant improvement over the best performing single model, suggesting that stacking did not provide substantial benefits in this specific scenario.

Furthermore, robustness evaluation using a feasibility-preserving feature-space threat model indicated that adversarial perturbations had limited success, with an ASR of 0.03 at budget 10. This highlights the resilience of the system under various adversarial attacks. However, the results also revealed trade-offs between clean and robust performance, with defenses failing to consistently lower ASR across all data queries. The study provides a reusable evaluation framework that integrates dataset and split auditing, group-overlap checks, leakage attribution via ablation, shift-aware cross-collection settings, and robustness curve reporting.

Looking forward, future research should continue to build on this framework by adding dependency-aware feasibility constraints and developing more robust shift designs, such as time-aware and cross-host shifts. Additionally, stronger benchmarks are needed to accurately reflect how manipulations propagate through the memory acquisition and feature extraction phases. Overall, the findings of this study contribute to advancing malware detection methodologies, with the potential for real-world applications in enhancing cybersecurity.

6. DECLARATIONS

6.1. About Authors

Syamsu Hidayat (SH) https://orcid.org/0000-0003-0135-9253
Kusrini (KK) https://orcid.org/0000-0001-9573-3909
Ema Utami (EU) https://orcid.org/0000-0002-8237-8693
Arief Setyanto (AS) https://orcid.org/0000-0003-0721-3941
Kristina Vaher (KV) https://orcid.org/0009-0009-6790-0680

6.2. Author Contributions

Conceptualization: SH and KK; Methodology: EU; Software: AS and KV; Validation: SH and KK; Formal Analysis: EU and AS; Investigation: KV; Resources: SH; Data Curation: KK; Writing Original Draft Preparation: EU and KV; Writing Review and Editing: SH and KK; Visualization: EU. All authors, SH, KK, EU, AS, and KV, have read and agreed to the published version of the manuscript.

REFERENCES

[1] C.-H. Van Ouytsel, K. H. T. Dam, and A. Legay, "Analysis of machine learning approaches to packing detection," *Comput. Secur.*, vol. 136, p. 103536, 2024.
[2] D. Gibert, N. Totosis, C. Patsakis, Q. Le, and G. Zizzo, "Assessing the impact of packing on static machine learning-based malware detection and classification systems," *Comput. Secur.*, vol. 156, p. 104495, 2025.
[3] H. Safitri, M. H. R. Chakim, and A. Adiwijaya, "Strategy based technology-based startups to drive digital business growth," *Startupreneur Business Digital (SABDA Journal)*, vol. 2, no. 2, pp. 207–220, 2023.
[4] M. M. Alani, A. Mashatan, and A. Miri, "Xmal: A lightweight memory-based explainable obfuscated-malware detector," *Comput. Secur.*, vol. 133, p. 103409, 2023.
[5] Y. Dehfouli and A. H. Lashkari, "Memory analysis for malware detection: A comprehensive survey using the oscar methodology," *vol. 58*, no. 4, 2025.
[6] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A systematic literature review on windows malware detection: Techniques, research issues, and future directions," *Journal of Systems and Software*, vol. 209, 2024.
[7] P., A. N. Mahmood, and M. J. M. Chowdhury, "Memaldet: A memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations," *Comput. Secur.*, vol. 142, p. 103864, 2024.
[8] X. Ling *et al.*, "Adversarial attacks against windows pe malware detection: A survey of the state-of-the-art," *Comput. Secur.*, vol. 128, p. 103134, 2023.
[9] J. Jones, E. Harris, Y. Febriansah, A. Adiwijaya, and I. N. Hikam, "Ai for sustainable development: Applications in natural resource management, agriculture, and waste management," *International Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 143–149, 2024.
[10] S. Yan, J. Ren, W. Wang, L. Sun, W. Zhang, and Q. Yu, "A survey of adversarial attack and defense methods for malware classification in cyber security," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 467–496, 2023.
[11] L. Liu, X. Kuang, L. Liu, and L. Zhang, "Defend against adversarial attacks in malware detection through attack space management," *Comput. Secur.*, vol. 141, p. 103841, 2024.
[12] D. Li, S. Cui, Y. Li, J. Xu, F. Xiao, and S. Xu, "Pad: Towards principled adversarial malware detection against evasion attacks," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 2, pp. 920–936, 2024.
[13] A. Kanivia, H. Hilda, A. Adiwijaya, M. F. Fazri, S. Maulana, and M. Hardini, "The impact of information technology support on the use of e-learning systems at university," *International Journal of Cyber and IT Service Management*, vol. 4, no. 2, pp. 122–132, 2024.
[14] T. Carrier, P. Victor, A. Tekeoglu, and A. Habibi Lashkari, "Detecting obfuscated malware using memory feature engineering," in *International Conference on Information Systems Security and Privacy*, 2022, pp. 177–188.
[15] B. Molina-Coronado, U. Mori, A. Mendiburu, and J. Miguel-Alonso, "Towards a fair comparison and realistic evaluation framework of android malware detectors based on static analysis and machine learning," *Comput. Secur.*, vol. 124, p. 102996, 2023.

[16] Z. Kan *et al.*, "Tesseract: Eliminating experimental bias in malware classification across space and time (extended version)," 2025.
[17] E. R. Rahayu, A. Aprillia, R. Z. Ikhsan, A. Adiwijaya, and A. Kumara, "Cybersecurity in the age of iot and developing frameworks for securing smart devices and networks," *Journal of Computer Science and Technology Application*, vol. 2, no. 1, pp. 46–54, 2025.
[18] G. Liu, D. Caragea, X. Ou, and S. Roy, "The impact of train-test leakage on machine learning-based android malware detection," 2025, online available: http://arxiv.org/abs/2410.19364.
[19] A. Guerra-Manzanares, M. Luckner, and H. Bahsi, "Concept drift and cross-device behavior: Challenges and implications for effective android malware detection," *Comput. Secur.*, vol. 120, p. 102757, 2022.
[20] D. Escudero García, N. DeCastro-García, and A. L. Muñoz Castañeda, "An effectiveness analysis of transfer learning for the concept drift problem in malware detection," *Expert Syst. Appl.*, vol. 212, p. 118724, 2023.
[21] L. Larisang, S. Sanusi, M. A. Bora, and A. Hamid, "Practicality and effectiveness of new technopreneurship incubator model in the digitalization era," *Aptisi Transactions on Technopreneurship (ATT)*, vol. 7, no. 2, pp. 318–333, 2025.
[22] D. W. Fernando and N. Komninos, "Fesa: Feature selection architecture for ransomware detection under concept drift," *Comput. Secur.*, vol. 116, p. 102659, 2022.
[23] A. Augello, A. De Paola, and G. Lo Re, "M2fd: Mobile malware federated detection under concept drift," *Comput. Secur.*, vol. 152, p. 104361, 2025.
[24] University of New Brunswick, "Malware memory analysis (cic-malmem-2022)," 2022, available at: https://www.unb.ca/cic/datasets/malmem-2022.html.
[25] L. Godoy, "Malware memory analysis / cic-malmem-2022," 2022, available at: https://www.kaggle.com/datasets/luccagodoy/obfuscated-malware-memory-2022-cic.
[26] R. D. Pertiwi, P. A. Fitria, T. P. Utami, R. A. Pamungkas, R. Pradana, and K. Chamroonsawasdi, "Gold nanoparticle synthesis using quercetin: Innovation in biopreneur," *Aptisi Transactions on Technopreneurship (ATT)*, vol. 7, no. 1, pp. 294–305, 2025.
[27] L. D'hooge, M. Verkerken, T. Wauters, F. De Turck, and B. Volckaert, "Castles built on sand: Observations from classifying academic cybersecurity datasets with minimalist methods," in *Proceedings of the 8th International Conference on Internet of Things, Big Data and Security*. SCITEPRESS - Science and Technology Publications, 2023, pp. 61–72.
[28] D. Arp *et al.*, "Pitfalls in machine learning for computer security," *Commun. ACM*, vol. 67, no. 11, pp. 104–112, 2024.
[29] J. Cortellazzi *et al.*, "Intriguing properties of adversarial ml attacks in the problem space [extended version]," *ACM Transactions on Privacy and Security*, vol. 28, no. 4, pp. 1–37, 2025.
[30] H. A. Winata and F. Simon, "Influence of profitability, audit quality, and corporate governance on earnings management," *APTISI Transactions on Management*, vol. 8, no. 2, pp. 93–104, 2024.
[31] S. Vladov, V. Vysotska, V. Varlakhov, M. Nazarkevych, S. Bolvinov, and V. Piadyshev, "Innovative method for detecting malware by analysing api request sequences based on a hybrid recurrent neural network for applied forensic auditing," *Applied System Innovation*, vol. 8, no. 5, p. 156, 2025.
[32] A. Tiwari, N. S. Chaudhari *et al.*, "Obfuscated memory malware detection," *arXiv preprint arXiv:2408.12866*, 2024.
[33] M. Adela and M. Tuti, "Increasing customer repurchase intention: The significance of product quality, viral marketing, and customer experience," *APTISI Transactions on Management*, vol. 8, no. 2, pp. 105–114, 2024.
[34] K. Aryal, M. Gupta, M. Abdelsalam, P. Kunwar, and B. Thuraisingham, "A survey on adversarial attacks for malware analysis," *IEEE Access*, vol. 13, pp. 428–459, 2024.
[35] Z. Gu, J. Guo, J. Xu, Y. Sun, and W. Liang, "Domain knowledge-driven method for threat source detection and localization in the power internet of things," *Electronics*, vol. 14, no. 13, p. 2725, 2025.
[36] OECD, *Developing a data-driven corruption risk model to strengthen integrity in Belgium: Practical considerations for the Federal Internal Audit*. Paris: OECD Publishing, 2025. [Online]. Available: https://doi.org/10.1787/e85587eb-en
[37] A. Hernandez-Suarez, G. Sanchez-Perez, L. K. Toscano-Medina, J. Olivares-Mercado, J. Portillo-Portilo, J.-G. Avalos, and L. J. Garcia Villalba, "Detecting cryptojacking web threats: An approach with autoencoders and deep dense neural networks," *Applied Sciences*, vol. 12, no. 7, p. 3234, 2022.