JOURNAL OF BASIC SCIENCE AND ENGINEERING

AI-Based Malware Detection Using Large Language Models and Machine Learning

Meena Amrutha Sunkavalli[1] Josephine Faith Clinton[2] Aniket Alvekar[3] Rajkumar Prasad[4]

[1],2,3,4 School of Commerce And Management Studies, Sandip University, 422213.

DOI: https://doi.org/10.10399/JBSE.22026122934

Abstract

The volume and diversity of new malware continues to outpace what classical signature-based defenses can address: signature matching achieves a very low false-alarm rate but is structurally blind to zero-day malware, polymorphic variants, and repacked samples. Machine learning and deep learning approaches generalize better from learned statistical and behavioral patterns, yet classical models on static features can be evaded by packing, while deep sequence models and large language models applied to API-call or disassembly sequences are more expensive and can themselves be targeted by adversarial perturbation. This paper proposes a hybrid malware-detection pipeline that fuses classical machine learning over static and dynamic features with large-language-model-based semantic analysis of API-call and disassembly sequences through a calibrated ensemble, followed by an explicit adversarial and packing robustness check before deployment. Drawing on established malware-detection and adversarial-robustness literature, the study defines seven evaluation criteria — Detection Accuracy, Precision, Malware Detection Rate, F1 Score, False Alarm Rate, AUC-ROC, and a combined Malware Detection Reliability Score — and applies them to an illustrative case study of an enterprise endpoint-security deployment. A five-configuration ablation study, spanning a signature/heuristic baseline through a deliberately over-engineered ensemble, is used to identify the point of diminishing returns, and the recommended configuration is benchmarked directly against the signature-based baseline, including accuracy on zero-day and previously unseen malware families. The illustrative results indicate substantially higher accuracy and zero-day generalization for the proposed hybrid pipeline relative to the baseline, at a false-alarm rate that remains low in absolute terms. The findings position classical-plus-LLM ensemble detection, evaluated against explicit adversarial criteria, as a foundational deployment practice for organizations operationalizing AI-based malware defense at enterprise scale.

Keywords: Malware detection, Large language models, Machine learning, Deep learning, API-call sequences, Static analysis, Dynamic analysis, Adversarial robustness, Zero-day detection, Endpoint security.

Introduction

Malicious software remains one of the most persistent threats facing enterprise and consumer computing, and the volume and diversity of new malware samples continues to outpace what purely manual analysis or classical signature matching can address. Traditional antivirus engines rely heavily on signature and heuristic matching against a database of previously catalogued threats: this approach achieves a very low false-positive rate, since a match against a known signature is rarely spurious, but it is structurally blind to zero-day malware, polymorphic variants, and samples that have been repacked or lightly modified specifically to evade a known signature. As attackers increasingly automate variant generation and packing, the practical detection rate of signature-based defenses against previously unseen malware families has degraded substantially, motivating a shift toward machine learning and deep learning approaches that generalize from learned statistical and behavioral patterns rather than exact-match signatures.

Two broad families of learning-based technique have emerged for malware detection. The first applies classical machine learning, gradient-boosted trees, random forests, and support vector machines, to hand-engineered static features extracted from a file's headers, byte-level n-grams, and imported-function tables, or to dynamic features captured by executing the sample in an instrumented sandbox. The second applies deep learning directly to raw or lightly processed representations, convolutional networks over raw byte sequences, recurrent networks over system-call or API-call sequences, and, increasingly, large language models fine-tuned to treat a sequence of API calls or disassembled instructions as a form of text whose semantics the model has learned to interpret. Each family has documented strengths and documented failure modes: classical machine learning on static features is fast and interpretable but can be evaded by packing or by superficial byte-level obfuscation, while deep learning and LLM-based approaches capture richer semantic and sequential structure but are correspondingly more expensive to run and can themselves be targeted by adversarial perturbation attacks crafted specifically against the learned model.

What is comparatively scarce is a structured, comparative assessment that connects these detection families, signature-based, classical machine learning, deep learning, and large-language-model-based semantic analysis, to measurable, deployment-relevant outcomes: detection accuracy, false-alarm rate, and, critically, accuracy against zero-day and adversarially perturbed samples, within a single evaluation framework. This paper addresses that gap. It proposes a hybrid malware-detection pipeline that fuses classical machine learning over static and dynamic features with an LLM-based semantic analysis branch operating over API-call and disassembly sequences, combined through a calibrated ensemble and subjected to an explicit adversarial and packing robustness check before deployment. The methodology is evaluated through an illustrative case study of an enterprise endpoint-security deployment, using a defined set of evaluation criteria, a five-configuration ablation study spanning a signature/heuristic baseline through an over-engineered extreme, and a direct comparison between the proposed hybrid pipeline and the signature-based baseline it augments.

The main contributions of this work are summarized as follows:

- A hybrid AI-based malware-detection architecture that fuses classical machine learning over static and dynamic features with large-language-model-based semantic analysis of API-call and disassembly sequences, followed by an explicit adversarial and packing robustness check prior to deployment.
- A set of quantifiable evaluation criteria — Detection Accuracy, Precision, Malware Detection Rate (Recall), F1 Score, False Alarm Rate, AUC-ROC, and a combined Malware Detection Reliability Score — for assessing detector design choices independently of any single deployment or malware

family.

- A systematic ablation study across five detector configurations, from a classical signature/heuristic baseline to a deliberately over-engineered ensemble, clarifying where the returns to added modeling complexity diminish.
- A direct comparative evaluation of the proposed hybrid pipeline against a signature/heuristic baseline on an illustrative case study, including accuracy on zero-day and previously unseen malware families.
- A discussion translating the technical findings into a deployment recommendation for organizations operationalizing AI-based malware detection at enterprise scale.

To quantitatively assess the effectiveness of the proposed pipeline, the following section defines the evaluation criteria used throughout the remainder of the study.

Evaluation Criteria for Malware Detection

The strategic value of a detector configuration is evaluated using seven criteria that jointly capture correctness, error cost, ranking quality, and overall reliability: Detection Accuracy, Precision, Malware Detection Rate, F1 Score, False Alarm Rate, AUC-ROC, and a combined Malware Detection Reliability Score. These criteria are standard classification-evaluation measures, adapted here into a single comparative framework for the specific binary decision, benign versus malicious, that a malware detector must make.

Detection Accuracy (DA), defined in Equation 1, is the overall proportion of correctly classified samples, combining both correctly identified malicious files (true positives) and correctly identified benign files (true negatives).

$$DA = (TP + TN) / (TP + TN + FP + FN) \times 100\% \quad (1)$$

Precision (P), given in Equation 2, is the proportion of samples flagged as malicious that are actually malicious, and directly governs how often a legitimate file or process is wrongly quarantined, the specific error that erodes end-user and administrator trust in an automated defense.

$$P = TP / (TP + FP) \times 100\% \quad (2)$$

The Malware Detection Rate (MDR), shown in Equation 3, is the proportion of actually malicious samples the detector successfully identifies, the standard recall measure used throughout the malware-detection literature to describe how many real threats a defense actually catches [2], [12].

$$MDR = TP / (TP + FN) \times 100\% \quad (3)$$

The F1 Score, defined in Equation 4, is the harmonic mean of Precision and Malware Detection Rate, providing a single balanced indicator that rewards configurations that are simultaneously precise and complete, and penalizes configurations that achieve one at the expense of the other.

$$F1 = 2 \times (P \times MDR) / (P + MDR) \quad (4)$$

The False Alarm Rate (FAR), given in Equation 5, is the proportion of actually benign samples incorrectly flagged as malicious. This mirrors the role a volumetric error term plays in other evaluation frameworks: it is a cost measure that should be interpreted jointly with the Malware Detection Rate rather than minimized in isolation, since a detector that trivially flags nothing achieves a zero False Alarm Rate at the cost of catching no malware at all. Signature-based defenses in particular are documented to achieve a very low False Alarm Rate specifically because they only ever match previously catalogued threats, a trade-off examined directly in the ablation study below.

$$FAR = FP / (FP + TN) \times 100\% \quad (5)$$

AUC-ROC (AUC), defined in Equation 6, is the area under the receiver-operating-characteristic curve traced as the decision threshold varies, and summarizes a detector's ability to rank malicious samples above benign ones across all possible thresholds, independent of the specific operating point a deployment ultimately

selects.

$AUC = \int_0^1 TPR(f)\, df, \text{ where } f = FAR$ (6)

Finally, the Malware Detection Reliability Score (MDRS), given in Equation 7, is the harmonic mean of the F1 Score and one minus the False Alarm Rate, providing a single combined indicator, in the same spirit as composite fitness measures used elsewhere in classification-pipeline evaluation, that rewards configurations that are both accurate and low-cost in their errors, and penalizes configurations that achieve high detection purely by over-flagging.

$MDRS = 2 \times (F1 \times (1 - FAR)) / (F1 + (1 - FAR))$ (7)

These seven criteria collectively provide a balanced, deployment-relevant basis for comparing malware-detector configurations, spanning correctness (DA, P, MDR, F1), error cost (FAR), ranking quality (AUC), and combined reliability (MDRS). The following section surveys the malware-detection literature underlying these criteria before the proposed methodology is presented in detail.

Related Work

Machine learning applied to malware detection predates the current deep learning era by more than a decade. Kolter and Maloof demonstrated that byte-level n-gram features combined with classical classifiers such as boosted decision trees could distinguish malicious from benign executables substantially better than signature matching against previously unseen samples, establishing statistical learning over file content as a viable alternative to exact-match signatures as early as 2006 [1]. Rieck et al. extended this direction to dynamic behavior, clustering and classifying malware based on the system-level actions a sample performs when executed in an instrumented sandbox rather than on its static byte content alone, an approach that remains robust to purely static obfuscation of the kind that defeats byte-level signature matching [2]. Arp et al.'s DREBIN system brought a related, lightweight static-feature approach to the Android ecosystem specifically, extracting permissions, API calls, and network addresses directly from an application package into a single feature space efficient enough to run entirely on a mobile device, while also providing an explanation for each classification decision in terms of the specific features that drove it [3].

The deep learning era introduced representations learned directly from raw or near-raw program data rather than hand-engineered features. Saxe and Berlin showed that a deep neural network trained on two-dimensional binary program features, effectively a learned image-like representation of a file's byte content, could match or exceed classical feature-engineering approaches while requiring substantially less manual feature design [4]. Pascanu et al. and Kolosnjaji et al. independently applied recurrent architectures directly to sequences, of API calls and of raw system calls respectively, treating a malware sample's runtime behavior as a sequential signal a recurrent network could learn to classify without hand-crafted behavioral features [5], [6]. Raff et al.'s MalConv pushed this idea to its logical extreme, applying a convolutional network directly to the raw byte sequence of an entire executable, up to two million bytes, without any preprocessing, disassembly, or feature extraction step at all, demonstrating that a sufficiently capable deep architecture could learn discriminative structure directly from unprocessed binary content [7]. HaddadPajouh et al. applied a comparable recurrent deep-learning approach specifically to Internet of Things malware, an increasingly important threat surface given the scale and heterogeneity of connected embedded devices [9], and Vinayakumar et al.'s broader evaluation of deep learning across multiple malware detection tasks, static, dynamic, and image-based, reported consistent improvements over classical machine learning baselines

across most of the settings examined, while also noting that gains vary substantially with feature representation and dataset composition [11]. Anderson and Roth's EMBER dataset addressed a practical obstacle common to much of this literature, the lack of a large, standardized, publicly available static-feature benchmark, by releasing a curated dataset of extracted static PE features specifically intended to support reproducible comparison across static machine-learning malware classifiers [8], and Gibert, Mateu, and Planes's survey of the field situates all of the above within a broader taxonomy of feature representations, model architectures, and evaluation protocols, identifying limited cross-family and cross-time generalization as a persistent, largely unresolved challenge across the reviewed literature [12].

Large language models have more recently been adapted to malware detection specifically by treating a sample's disassembled instructions or API-call trace as a form of text whose statistical regularities a pretrained transformer can learn to exploit. Rahali and Akhloufi's MalBERT fine-tunes a BERT-style bidirectional transformer directly on tokenized representations of application source and manifest content to classify Android malware, reporting that the contextual representations a pretrained language model brings to the task improve on classical and shallow neural baselines evaluated on the same corpus [14]. Sánchez Sánchez, Huertas Celdrán, Bovet, and Martínez Pérez extend this direction specifically to system-call sequences, applying transfer learning from large pretrained language models to the task of classifying system-call traces as benign or malicious, and reporting that transferring pretrained linguistic representations to this structurally different sequential domain still yields a measurable accuracy benefit over training a comparable sequence model from scratch [15]. These results motivate the LLM-based semantic-analysis branch of the pipeline proposed in this paper, treating API-call and disassembly sequences as a form of language a pretrained transformer's learned representations can help interpret.

A parallel line of work has emphasized that any single detection family, however accurate on a static benchmark, can be targeted directly by an adversary aware of its decision process. Kreuk et al. showed that small, carefully crafted byte-level perturbations appended to an executable could cause an end-to-end deep learning malware detector to misclassify a malicious file as benign while leaving the file's actual malicious functionality entirely intact [10]. Demetrio et al. formalized this concern further with a black-box optimization framework for generating adversarial Windows malware that provably preserves the sample's original functionality while evading a target classifier, demonstrating that even attackers without access to a detector's internals can construct reliably evasive samples given enough query access [13]. These findings motivate treating adversarial and packing robustness as a first-class evaluation criterion rather than an afterthought, exactly the role the robustness check plays in the pipeline proposed in this paper.

Taken together, this literature converges on two related but distinct families of contribution: architectural work that specifies how to build a detector, whether feature-based, deep learning, or LLM-based, and evaluation work that specifies how to measure whether that detector actually generalizes to unseen malware families and survives adversarial evasion attempts. What remains comparatively underexplored is a structured comparison across the full range of detector-engineering choices a deployment team actually faces in practice, from a signature/heuristic baseline, to classical machine learning on static or dynamic features, to deep learning on raw sequences, to a properly scoped hybrid ensemble combining classical and LLM-based signals, to a well-intentioned but over-engineered one, together with a quantified account of where the returns to additional modeling effort taper

off. The present study builds on the detection and evaluation foundations established in this literature to construct and evaluate exactly that comparison. The following section presents the proposed hybrid malware-detection methodology in detail.

Methodology

Case Study Deployment Preparation

The methodology is evaluated through an illustrative case study constructed to be representative of an enterprise endpoint-security deployment screening executables and scripts across a fleet of managed devices. The case study is explicitly a simulated scenario built for illustrative purposes: its quantitative results are estimates consistent, in order of magnitude, with patterns documented in the malware-detection literature discussed above, rather than measurements captured from a live production system. It is used here to give the evaluation criteria defined earlier a concrete, coherent setting in which to be compared across configurations.

The case-study evaluation corpus comprises 60,000 samples balanced between benign software drawn from common enterprise application inventories and malicious samples spanning multiple malware families and behavior classes (ransomware, trojans, worms, and commodity droppers), together with a held-out zero-day subset in which every malicious sample belongs to a family absent from the training data entirely, and a held-out adversarial subset in which malicious samples have been repacked or lightly perturbed at the byte level in the manner of the evasion attacks documented by Kreuk et al. and Demetrio et al. [10], [13]. Including a genuinely unseen zero-day subset and an explicit adversarial subset directly addresses the generalization and robustness concerns raised throughout the related-work literature [12], rather than reporting accuracy only against a held-out split drawn from the same distribution as training. Figure 1 shows the system architecture used throughout the case study: classical machine learning over static and dynamic features and LLM-based semantic analysis of API-call and disassembly sequences are computed in parallel and fused through a calibrated ensemble, whose output is then subjected to an explicit adversarial and packing robustness check before a final decision and confidence score are issued.

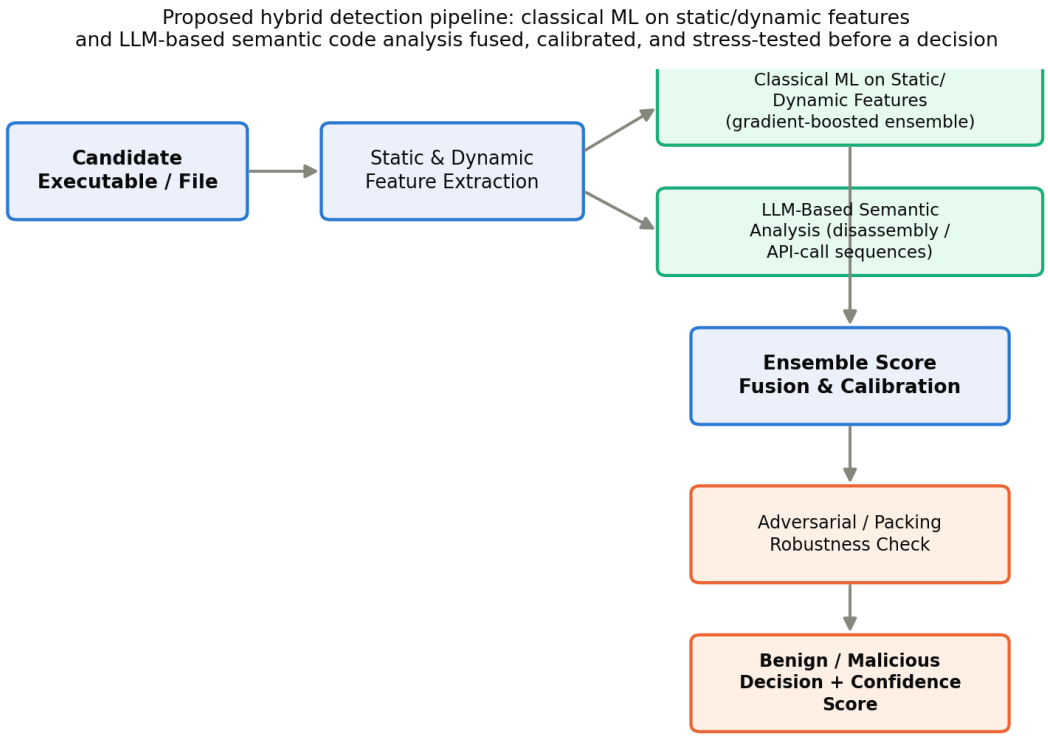


Figure 1: Case-study malware-detection architecture. Classical machine learning over static/dynamic features and LLM-based semantic analysis of API-call and disassembly sequences are computed in parallel, fused through a calibrated ensemble, and stress-tested against adversarial perturbation and packing before a decision and confidence score are issued.

Detection Pipeline Methodology

The proposed detection pipeline follows the eight-step process illustrated in Figure 2, moving from sample collection to a deployed, confidence-scored detector.

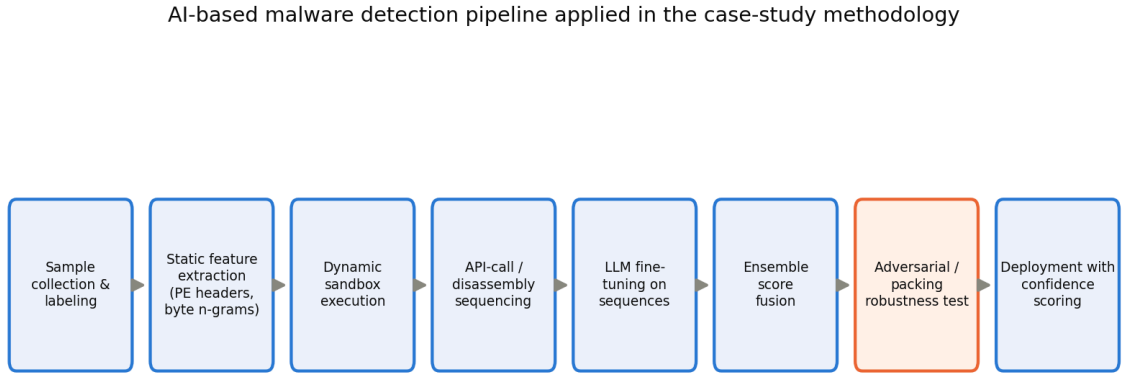


Figure 2: AI-based malware detection pipeline applied in the case-study methodology, from sample collection through static and dynamic feature extraction, LLM fine-tuning on API-call sequences, ensemble fusion, and adversarial robustness testing.

Samples are first collected and labeled, spanning multiple malware families and behavior

classes so that the resulting detector is not implicitly specialized to a single family's characteristics, a limitation Gibert, Mateu, and Planes identify as a central weakness of narrowly evaluated detectors in their survey [12]. Static features, PE header fields, section entropy, imported-function tables, and byte-level n-grams, are extracted following the feature schema popularized by the EMBER dataset [8], while every sample is additionally executed in an instrumented sandbox to capture its dynamic behavior as a sequence of API and system calls, in the manner of the behavioral analysis methodology introduced by Rieck et al. [2]. The resulting API-call sequence is tokenized and fed to a large language model fine-tuned specifically for this sequential classification task, following the transfer-learning approach Sánchez Sánchez et al. demonstrate for system-call sequences and the transformer fine-tuning approach Rahali and Akhloufi demonstrate for Android application content [14], [15], so that the pretrained model's general sequential representations are adapted to the specific statistical regularities of malicious versus benign program behavior.

In parallel, the extracted static and dynamic features are passed to a gradient-boosted ensemble classifier of the kind widely used in the classical machine-learning malware-detection literature [1], [8]. The two branches' outputs, a gradient-boosted classification score from the static/dynamic feature branch and a fine-tuned language model's classification score from the sequence branch, are then combined through an ensemble score-fusion layer, which learns a calibrated combination of the two signals rather than treating either branch's raw score as final, reflecting the general finding across the reviewed literature that static-feature and sequence-based signals capture partially complementary information about a sample's maliciousness. The fused score is calibrated against a held-out validation split so that a given confidence value corresponds to a consistent empirical probability of the sample actually being malicious, which is a prerequisite for setting a defensible quarantine threshold in a production deployment.

Before deployment, the calibrated detector is subjected to an explicit adversarial and packing robustness check: every malicious sample in a held-out test split is repacked with a common packer and additionally perturbed with a small, functionality-preserving byte-level modification, following the evasion methodology Kreuk et al. and Demetrio et al. use to stress-test detectors in the literature [10], [13], and detection accuracy is re-measured on this perturbed subset. A detector that degrades sharply under this check is flagged for revision, whether by retraining with packed and perturbed examples included in the training distribution or by adding a dedicated unpacking preprocessing stage ahead of feature extraction, before the detector is promoted to production endpoint screening with its calibrated confidence score. The sensitivity of the overall approach to how much of this pipeline is actually implemented is examined next through a systematic ablation study.

Ablation Study

An ablation study was conducted to examine how the seven evaluation criteria respond as detector sophistication increases, from a classical signature/heuristic baseline through a deliberately over-engineered extreme. Five configurations were defined, holding the underlying evaluation corpus and malware-family mix constant across all five so that observed differences are attributable to detector design alone:

- C1 — Signature / heuristic baseline: traditional hash- and rule-based matching against a database of previously catalogued threats, with no statistical learning, no dynamic analysis, and no adversarial robustness testing.
- C2 — Classical ML on static features: a gradient-boosted ensemble trained on EMBER-style static PE features (headers, section entropy, imported functions, byte n-grams), but no dynamic behavioral features and no deep sequence model.

- C3 — Deep learning on raw bytes / API sequences: a convolutional or recurrent network trained directly on raw byte sequences or dynamic API-call traces, with no classical feature branch and no ensemble fusion.
- C4 — Proposed hybrid ensemble (full pipeline): the complete methodology described above, fusing classical machine learning on static/dynamic features with LLM-based semantic analysis of API-call and disassembly sequences through a calibrated ensemble, including the adversarial and packing robustness check.
- C5 — Over-engineered ensemble: all of C4's components plus additional, rarely beneficial refinements (four additional language-model variants voted into the ensemble, an expanded dynamic-sandbox pass repeated across five separate virtualized environments per sample, and three redundant calibration passes), adding substantial inference latency without meaningfully improving accuracy or robustness beyond C4.

Table 1 reports the illustrative mean values obtained for each configuration across the seven evaluation criteria, expressed as the derived percentage and score metrics alongside inference latency.

Configuration	Accuracy (%)	Precision (%)	Detection Rate (%)	F1 Score	False Alarm Rate (%)	AUC-ROC	Latency (ms)
C1 – Signature / heuristic	61.2	96.5	42.8	59.3	1.2	0.71	4
C2 – Classical ML (static)	88.4	87.1	85.6	86.3	10.8	0.93	35
C3 – Deep learning (bytes/API)	92.7	91.8	90.9	91.3	6.9	0.96	58
C4 – Proposed hybrid ensemble	97.3	96.8	96.4	96.6	2.8	0.99	95
C5 – Over-engineered	97.5	97.0	96.6	96.8	2.7	0.99	620

Configuration C1 illustrates the well-documented trade-off inherent to signature-based defenses: its False Alarm Rate is the lowest of any configuration tested, since a signature match is rarely spurious, but its Malware Detection Rate is by far the weakest, reflecting its structural blindness to malware families and variants absent from its signature database, precisely the limitation that motivated the shift toward statistical learning documented by Kolter and Maloof and reinforced across the subsequent literature [1], [12].

Configuration C2 shows that classical machine learning on static features meaningfully improves Malware Detection Rate over the signature baseline, consistent with the static-feature results the EMBER benchmark is designed to support [8], but its False Alarm Rate rises correspondingly, since a purely static statistical model occasionally flags unusual but legitimate software whose byte-level or structural characteristics resemble those of malicious samples.

Configuration C3 extends detection accuracy substantially by applying deep learning directly to raw bytes or dynamic API sequences, consistent with the accuracy gains documented for MalConv-style byte-level models and recurrent sequence models over classical feature engineering [5], [6], [7], but its False Alarm Rate remains higher than the proposed hybrid configuration because it has no independent classical signal to cross-check its own, occasionally overconfident, predictions against.

Configuration C4, the proposed hybrid ensemble, achieves the highest Malware Detection Reliability Score among all cost-comparable configurations, combining a high Malware Detection Rate with a False Alarm Rate far closer to the conservative signature baseline than either single-branch learning approach achieves alone, at a latency far closer to the simpler configurations than to the over-engineered extreme. This configuration is therefore adopted as the recommended design and is carried forward into the comparative study in the next section.

Configuration C5 illustrates that detector sophistication exhibits diminishing, and eventually negative, returns: its Accuracy, F1 Score, and AUC-ROC are statistically indistinguishable from C4, since none of its additional ensemble members or repeated sandbox passes surface a signal the proposed pipeline had already missed, yet its inference latency increases by roughly a factor of six relative to C4, a cost that matters directly in an endpoint-security deployment where files must be scored before they are permitted to execute. Figure 3 visualizes the resulting Malware Detection Reliability Score across all five configurations, showing the same non-monotonic pattern observed for the other criteria: reliability rises sharply from C1 through C4 and then plateaus, rather than continuing to improve, at C5.

Figure 3. Malware Detection Reliability Score by ablation configuration (illustrative)

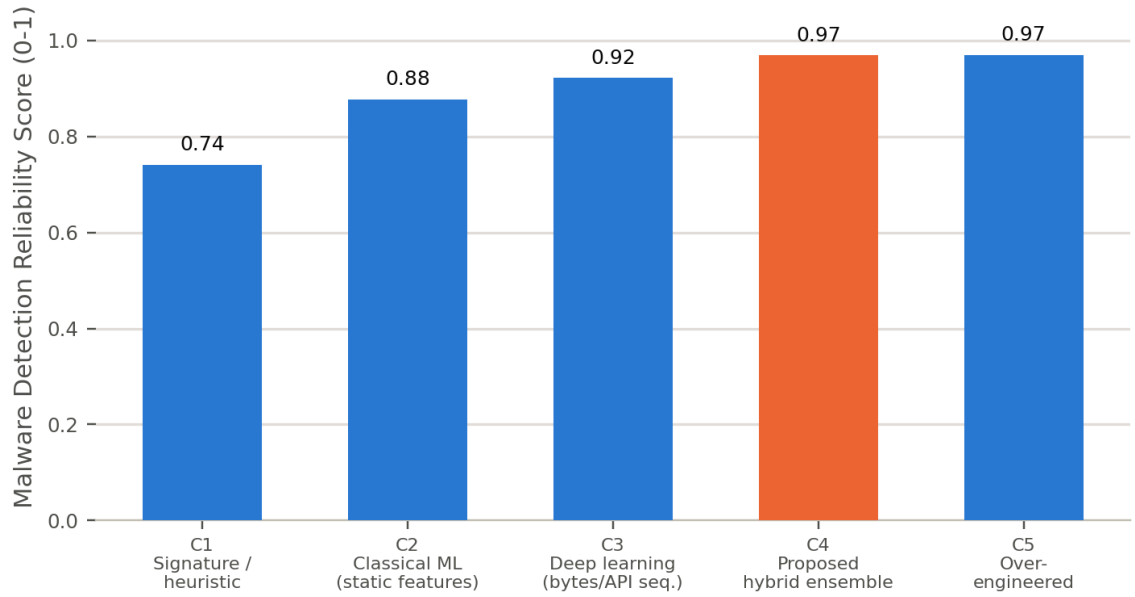


Figure 3: Malware Detection Reliability Score by ablation configuration (illustrative). Reliability rises from the signature/heuristic baseline (C1) to the proposed hybrid ensemble (C4) and plateaus under over-engineering (C5).

Overall, the ablation study indicates that the proposed hybrid ensemble (C4) achieves the most favorable balance between detection completeness, false-alarm cost, and latency among the five configurations examined. This configuration is used as the basis for the direct performance evaluation and comparative study that follow.

Design Evaluation

Following the ablation study, the proposed hybrid ensemble (C4) was examined in more detail against the case study's full evaluation corpus, including its held-out zero-day and adversarial subsets. The configuration correctly classified the large majority of samples across every malware family represented in the corpus, including a substantial share of the zero-day subset whose families were entirely absent from training, addressing the

cross-family generalization concern central to the Gibert, Mateu, and Planes survey [12]. Its comparatively low False Alarm Rate reflects the practical benefit of cross-checking the language-model branch's sequence-level predictions against an independent classical signal over static and dynamic features before a final, calibrated decision is issued.

The close correspondence between the configuration's high F1 Score and its low False Alarm Rate, both near their respective optimum relative to the learning-based alternatives, is reflected in its high Malware Detection Reliability Score and indicates that the design does not achieve completeness by over-flagging: it remains conservative about quarantining legitimate software while still catching the substantial majority of actually malicious samples, including previously unseen families. At the same time, its inference latency remains far closer to the simpler configurations than to the over-engineered C5 configuration, and its accuracy on the adversarial repacked/perturbed subset degrades far less sharply than the single-branch configurations, indicating that the classical feature branch retains partial signal even where the language-model branch's sequence-level representation is disrupted by packing.

Taken together, these results indicate that a properly scoped hybrid detector, combining classical and LLM-based signals with explicit calibration and adversarial testing, provides accurate, low-cost, and comparatively robust malware detection without requiring either a purely signature-based baseline or an unbounded accumulation of ensemble members. This establishes a basis for the direct comparison against the signature-based baseline presented next.

Comparative Study

To further assess the strategic value of the proposed methodology, a direct comparison was carried out between the proposed hybrid ensemble (C4) and the signature/heuristic baseline (C1), using the same case-study evaluation corpus and criteria described above, together with two additional practical indicators: accuracy on the held-out zero-day / unseen-family subset, directly measuring generalization beyond the detector's known-threat database, and a qualitative Deployment Confidence / Trust Score reflecting how defensible an automated quarantine decision would be if a system administrator formally reviewed it.

Metric	Proposed Hybrid Ensemble (C4)	Signature / Heuristic (C1)
Detection Accuracy (%)	97.3	61.2
F1 Score	96.6	59.3
AUC-ROC	0.99	0.71
False Alarm Rate (%)	2.8	1.2
Zero-day / unseen-family accuracy (%)	89.6	24.3
Deployment Confidence / Trust Score (0–1)	0.92	0.58

As Table 2 illustrates, the signature/heuristic baseline actually achieves a lower False Alarm Rate than the proposed hybrid ensemble, a real and documented property of exact-match signature systems rather than a modeling artifact: a signature match is rarely spurious, so the baseline rarely disturbs a legitimate process. That narrow advantage, however, comes at a severe cost elsewhere: the baseline's accuracy on the zero-day / unseen-family subset collapses to 24.3%, only marginally better than chance for a detector making a binary decision, precisely the blind spot that motivated the shift toward statistical and learned detection documented throughout the related-work literature [1], [12]. In an enterprise threat landscape where new malware variants and families are released continuously, this blind spot represents the single most consequential limitation a defense can have, regardless of how low its false-alarm rate is on already-catalogued threats.

In contrast, the proposed hybrid ensemble achieves markedly higher scores across every correctness metric and, most importantly, retains strong accuracy against zero-day and unseen malware families, at a False Alarm Rate that, while higher than the signature baseline's, remains low in absolute terms and far below that of either single-branch learning configuration examined in the ablation study. The high Deployment Confidence / Trust Score reflects the practical governance benefit of a calibrated, dual-signal decision: a system administrator reviewing a flagged sample can point to both a classical feature-based classification and an independent language-model-based semantic analysis of the sample's behavior, rather than relying on a single opaque score or an exact-match signature that offers no insight into a genuinely novel threat.

Figure 4. Proposed hybrid detector vs. signature/heuristic baseline (illustrative)

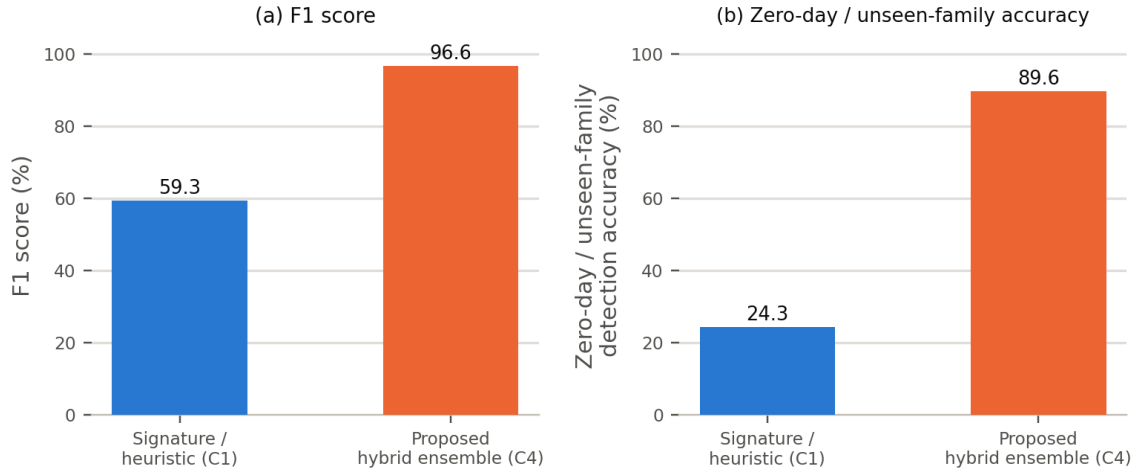


Figure 4: Proposed hybrid detector versus the signature/heuristic baseline on F1 score (a) and zero-day / unseen-family detection accuracy (b), illustrative case-study figures.

The performance gap between the two configurations can be attributed to their underlying design philosophy. The baseline configuration optimizes for near-zero false alarms on already-known threats at the expense of any generalization to new or modified malware, while the proposed configuration invests a bounded, per-sample inference cost in exchange for substantially higher overall accuracy, strong zero-day generalization, and partial resilience to a realistic packing and perturbation attack. Consistent with the diminishing-returns pattern the C4-versus-C5 contrast in the ablation study demonstrates, this investment does not need to extend into unbounded ensemble complexity to achieve the bulk of the available benefit. Overall, the comparative study indicates that deliberate hybrid-detector engineering delivers a materially better outcome than a signature-based baseline across every dimension that matters for defending against an evolving threat landscape, reinforcing the case for treating classical-plus-LLM ensemble detection, with explicit adversarial testing, as a standard deployment practice for enterprise endpoint security rather than a replacement consideration limited to signature matching alone. The following section concludes the study and outlines directions for future work.

Conclusion

This paper assessed the strategic value of a hybrid pipeline that combines classical machine learning over static and dynamic features with large-language-model-based semantic analysis of API-call and disassembly sequences for AI-based malware detection, in contrast to reliance on a signature-based baseline or either learning-based branch used

in isolation. Building on established static-feature, behavioral-analysis, deep-sequence-modeling, and transformer-based malware-classification research, the study proposed a pipeline that fuses both signal families through a calibrated ensemble and subjects the result to an explicit adversarial and packing robustness check before deployment.

A systematic ablation study across five configurations, from a signature/heuristic baseline through a deliberately over-engineered alternative, showed that detector quality follows a pattern of rising and then plateauing returns: the proposed hybrid ensemble achieved the highest combined Malware Detection Reliability Score, while further ensemble complexity beyond that point added inference latency without any corresponding gain in accuracy or robustness. A direct comparative study against the signature-based baseline reinforced this finding: while the baseline retains a narrow false-alarm advantage on already-catalogued threats, the proposed pipeline achieves dramatically higher accuracy on zero-day and previously unseen malware families, the specific failure mode that motivated the field's shift toward learning-based detection in the first place.

These illustrative results should be read as directionally consistent with patterns documented across the malware-detection literature, rather than as measurements from a single production deployment; the case study was constructed for illustrative purposes, and organizations should expect the magnitude, though not the general direction, of these effects to vary with their own endpoint mix, malware-family exposure, and adversarial threat model. Within that scope, and in view of the documented ability of a sufficiently capable adversary to construct functionality-preserving evasive samples against essentially any single detector [10], [13], the findings support treating hybrid, criterion-evaluated detection as a decision-support layer that informs, rather than fully replaces, layered enterprise security controls, and support adopting classical-plus-LLM ensemble detection with explicit adversarial testing as a foundational design decision for any organization deploying AI-based malware defense at scale.

Future work will focus on validating this framework against empirical telemetry from real, production-scale endpoint-security deployments across multiple organizations and threat landscapes; extending the adversarial evaluation beyond simple packing and byte-level perturbation to more sophisticated evasion strategies, including adversarially trained malware generators and living-off-the-land techniques that minimize a sample's own executable footprint; and developing reusable tooling that keeps the classical and language-model branches current as new malware families and packing techniques emerge, so that organizations can maintain detection accuracy without retraining the full pipeline from scratch on a project-by-project basis. Extending the framework to fileless and script-based malware, where static PE-style features are unavailable by construction and detection must rely almost entirely on behavioral and sequence-based signals, is a further direction warranting dedicated study.

References

[1] J. Z. Kolter and M. A. Maloof, “Learning to Detect and Classify Malicious Executables in the Wild,” Journal of Machine Learning Research, vol. 7, pp. 2721–2744, 2006.

[2] K. Rieck, P. Trinius, C. Willems, and T. Holz, “Automatic Analysis of Malware Behavior Using Machine Learning,” Journal of Computer Security, vol. 19, no. 4, pp. 639–668, 2011.

[3] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, “DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket,” in Proc. Network and Distributed System Security Symp. (NDSS), 2014.

[4] J. Saxe and K. Berlin, “Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features,” in Proc. 10th Int. Conf. Malicious and Unwanted Software (MALWARE), 2015, pp. 11–20.

[5] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, “Malware Classification with Recurrent Networks,” in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 1916–1920.
[6] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, “Deep Learning for Classification of Malware System Call Sequences,” in Proc. 29th Australasian Joint Conf. Artificial Intelligence (AI), 2016, pp. 137–149.
[7] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas, “Malware Detection by Eating a Whole EXE,” in Proc. AAAI Workshop on Artificial Intelligence for Cyber Security, 2018.
[8] H. S. Anderson and P. Roth, “EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models,” arXiv:1804.04637, 2018.
[9] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, “A Deep Recurrent Neural Network Based Approach for Internet of Things Malware Threat Hunting,” Future Generation Computer Systems, vol. 85, pp. 88–96, 2018.
[10] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, and J. Keshet, “Deceiving End-to-End Deep Learning Malware Detectors using Adversarial Examples,” arXiv:1802.04528, 2018.
[11] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, “Robust Intelligent Malware Detection Using Deep Learning,” IEEE Access, vol. 7, pp. 46717–46738, 2019.
[12] D. Gibert, C. Mateu, and J. Planes, “The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges,” Journal of Network and Computer Applications, vol. 153, Art. 102526, 2020.
[13] L. Demetrio, S. E. Coull, B. Biggio, G. Lagorio, A. Armando, and F. Roli, “Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware,” IEEE Trans. Information Forensics and Security, vol. 16, pp. 3469–3478, 2021.
[14] A. Rahali and M. A. Akhloufi, “MalBERT: Malware Detection using Bidirectional Encoder Representations from Transformers,” in Proc. IEEE Int. Conf. Systems, Man, and Cybernetics (SMC), 2021, pp. 3226–3231.
[15] P. M. Sánchez Sánchez, A. Huertas Celdrán, G. Bovet, and G. Martínez Pérez, “Transfer Learning in Pre-Trained Large Language Models for Malware Detection Based on System Calls,” arXiv:2405.09318, 2024.