

API Call-Based Windows Malware Detection Using CNN-LSTM-Attention and Explainable SHAP

Febby Viona¹⁾, Puguh Hiskiawan^{2)*}

Data Science, Faculty of Technology and Design, Bunda Mulia University

¹⁾s36230018@student.ubm.ac.id, ²⁾phiskiawan@bundamulia.ac.id

Submitted : Jul 28, 2026 | Accepted : Aug 20, 2026 | Published : Oct 5, 2026

Abstract: The Windows operating system remains a primary target for cyberattacks, with modern malicious software increasingly relying on obfuscation, packing, and polymorphism to evade signature-based detection. This research addresses this challenge by developing a hybrid Convolutional Neural Network, Long Short-Term Memory, and Multi-Head Attention model to classify Windows malicious software from sequences of the first 100 dynamic Application Programming Interface calls. Drawing on the Kaggle "Malware Analysis Datasets: API Call Sequences" refined to 13,206 unique samples with a severe class imbalance of 95.36% malware versus 4.64% benign, the model employs a class-weighting strategy to ensure robust learning. Rigorous evaluation through five-fold cross-validation demonstrates that the proposed architecture achieves a mean Area Under the Receiver Operating Characteristic Curve of 0.9562 (± 0.0186) and an accuracy of 95.78% ($\pm 1.05\%$). Ablation studies confirm that the Multi-Head Attention mechanism is statistically indispensable ($p < 0.05$), as its removal degrades performance to an Area Under the Receiver Operating Characteristic Curve of 0.9073. While competitive with ensemble baselines such as Random Forest and XGBoost, the proposed approach provides superior interpretability by integrating SHapley Additive Explanations. This enables local and global forensic accountability by pinpointing critical temporal execution windows specifically at steps t_1 , t_{50} , and t_{75} that drive malicious predictions. These results indicate that combining hybrid deep learning with post-hoc explainability produces malware detection that is both robust and forensically transparent.

Keywords: API Call Sequence; CNN-LSTM-Attention; Explainable AI; Malware Detection; SHAP

INTRODUCTION

The rapid and exponential growth of information technology has fundamentally transformed operations across business, government, and critical infrastructure sectors worldwide. Within this highly digitalized landscape, the Microsoft Windows operating system has consistently maintained its position as the dominant desktop and enterprise platform globally (Maniriho et al., 2023). However, this dominance simultaneously renders the Windows ecosystem the primary target for malicious cyber actors. Modern threats such as ransomware, infostealers, rootkits, and Advanced Persistent Threats (APT) have evolved far beyond simple static malware, incorporating sophisticated evasion techniques including code obfuscation, packing, payload encryption, and polymorphism, which allow malware to dynamically alter its internal structure upon each infection (Aboaoja et al., 2022). Global cybersecurity agencies consistently report hundreds of millions of new Windows malware samples annually, with potential consequences ranging from large-scale data breaches and operational paralysis to massive financial losses (Yousuf et al., 2023).

Existing malware detection approaches have demonstrated significant limitations in addressing these evolving threats. Signature-based detection, the most widely deployed conventional method, proves ineffective against polymorphic and obfuscated malware that can continuously alter its binary fingerprint (Maniriho et al., 2024). Static analysis, which inspects executable code without running the program, is similarly constrained, as packed or encrypted malware conceals its malicious logic until runtime, causing such threats to evade static inspection entirely (Gyamfi et al., 2023). These shortcomings have driven a paradigm shift toward dynamic behavioral analysis, wherein a program is executed in an isolated sandbox environment to observe its actual runtime

*Puguh Hiskiawan



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

interactions with the operating system (Zhang et al., 2023). In this context, sequences of Windows API (Application Programming Interface) calls have emerged as a critical and highly reliable behavioral indicator, given that no Windows malware can interact with hardware resources or system memory without invoking the Windows API (Maniriho et al., 2023).

In recent years, deep learning architectures have emerged as the primary standard for addressing pattern recognition challenges in sequence-based data. The capability of neural networks to extract high-level temporal features has not only proven effective within the domain of malware analysis but has also demonstrated remarkable performance across diverse sequential domains, such as audio classification and traditional music recognition (Hiskiawan, Wijayanti, et al., 2025). Convolutional Neural Networks (CNN) have shown effectiveness in extracting local n-gram patterns from API call sequences, while Long Short-Term Memory (LSTM) networks excel at capturing long-range temporal dependencies across execution traces (Yao et al., 2024). Hybrid CNN-LSTM models have consistently outperformed single-architecture approaches by combining local pattern extraction with sequential context modeling (Akhtar & Feng, 2022; Hiskiawan, Wijayanti, et al., 2025). However, two critical research gaps remain unresolved. First, the majority of these models operate as black boxes, producing classification decisions without providing any interpretable explanation of which API calls or temporal positions drove the prediction. This opacity is particularly problematic in the digital forensic domain, where results must be independently verifiable and legally defensible in court (Hermosilla et al., 2025; Hiskiawan, Geasela, et al., 2025). Second, the specific temporal positions within a 100-step API call execution window that are most indicative of malicious behavior have not been systematically identified and visualized for forensic and model-interpretability use (Saqib et al., 2024).

To address these gaps, this study proposes a hybrid CNN-LSTM-Attention architecture integrated with SHapley Additive Explanations (SHAP) Kernel Explainer for Windows malware detection (Cannarile et al., 2022). The model incorporates a Multi-Head Attention mechanism designed to contextualize each API call position relative to the full execution sequence, supporting the overall classification process (Wang et al., 2024). The research is conducted on the publicly available "Malware Analysis Datasets: API Call Sequences" dataset from Kaggle, comprising 43,876 labeled program samples, each represented as a sequence of 100 integer-encoded API Call IDs (Catak et al., 2020). The highly imbalanced class distribution (97.54% malware vs. 2.46% benign) is addressed through class weighting during training (Palma Salas & de Geus, 2024). SHAP is applied post-training to generate both local explanations via Waterfall Plots for individual samples and aggregated feature importance maps across a representative subset of test samples (Yu et al., 2026). Specifically, this research aims to: (1) evaluate the capability of the CNN-LSTM-Attention model in achieving high classification accuracy on highly imbalanced sequential API call data, (2) demonstrate the effectiveness of SHAP in identifying specific API call positions that most influence malware classification decisions, and (3) determine the chronological positions within the 100-step execution window that carry the greatest forensic significance for malware detection.

LITERATURE REVIEW

Dynamic Malware Analysis and Windows API Call Sequences

Malware detection approaches are broadly categorized into static and dynamic analysis methods. Static analysis examines the binary structure of an executable file without executing it, relying on signatures, byte n-grams, and control flow graphs to identify malicious patterns. However, modern malware increasingly employs obfuscation, packing, and polymorphism to conceal its true structure at rest, rendering static inspection ineffective against a large proportion of contemporary threats. Dynamic analysis addresses this fundamental limitation by executing a program within a controlled sandbox environment and observing its runtime behavior, thereby capturing the true operational intent of a binary regardless of its surface-level obfuscation (Zhang et al., 2023).

Within the dynamic analysis paradigm, Windows Application Programming Interface (API) call sequences have emerged as the most reliable behavioral indicator for malware classification. Every interaction that a Windows executable performs with the operating system including file I/O, memory allocation, registry access, network socket creation, and process injection must be mediated through the Windows API. Consequently, the sequence and pattern of API calls invoked during program execution constitutes a comprehensive behavioral fingerprint that cannot be concealed through code obfuscation or packing alone (Maniriho et al., 2023). This characteristic makes API call sequences a particularly robust and noise-resistant feature representation for dynamic malware analysis compared to network traffic logs or system call traces.

Convolutional Neural Networks for Sequential Feature Extraction

Convolutional Neural Networks (CNN) were originally developed for spatial feature extraction in image recognition tasks, but have since demonstrated strong applicability to other data modalities, including sequential and one-dimensional data (Hiskiawan et al., 2023). In the context of malware detection, Conv1D filters operate as sliding windows across the API call sequence, extracting localized short-term n-gram patterns for example, identifying characteristic sequences of three to five consecutive API calls that commonly appear in ransomware

*Puguh Hiskiawan



This is anCreative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

initialization or memory injection routines (Bensaoud & Kalita, 2024). By stacking multiple Conv1D layers with increasing filter counts, the network progressively abstracts low-level API co-occurrence patterns into higher-level behavioral motifs. MaxPooling layers further reduce the dimensionality of extracted features while preserving the most prominent local activations, and Batch Normalization stabilizes training by normalizing intermediate layer outputs architectural optimizations that are widely integrated into modern deep learning frameworks to improve feature stability and model performance, as seen in various complex applications (Hiskiawan et al., 2026).

Long Short-Term Memory for Temporal Behavioral Modeling

While CNN effectively captures local short-range API co-occurrence patterns, it lacks the capacity to model long-range sequential dependencies that span large temporal distances within an execution trace. Long Short-Term Memory (LSTM) networks were specifically designed to address this limitation through a gated memory cell architecture that selectively retains, updates, or discards information across arbitrary sequence lengths (Yao et al., 2024). In malware behavioral analysis, LSTM layers process the CNN-extracted feature maps chronologically, maintaining a hidden state that accumulates contextual information about the entire preceding execution history. This enables the model to recognize that a suspicious memory allocation occurring at step t20 may only become definitively malicious when followed by a network connection call at step t75, a relationship that CNN alone cannot capture (Li et al., 2022). Hybrid CNN-LSTM architectures that combine local pattern extraction with temporal context modeling have consistently outperformed single-architecture approaches across multiple malware detection benchmarks (Akhtar & Feng, 2022).

Multi-Head Attention Mechanism

The Multi-Head Attention mechanism, originally introduced in the Transformer architecture, extends the capability of sequential models by enabling the network to simultaneously attend to multiple positions within a sequence with different representational subspaces (Wang et al., 2024). When applied on top of LSTM outputs in a hybrid architecture, Multi-Head Attention dynamically computes a weighted importance score for each API call position within the execution trace, allowing the model to selectively focus on the most behaviorally significant execution steps while suppressing irrelevant background system calls (Li et al., 2022). A residual connection between the LSTM output and the Attention output, followed by Layer Normalization, further stabilizes gradient flow and preserves low-level temporal features learned by the LSTM layer (Liang et al., 2026). This architectural combination ensures that both fine-grained local API patterns and globally significant temporal behavioral transitions are preserved and weighted appropriately in the final classification decision.

Explainable Artificial Intelligence (XAI) and SHAP

Despite the strong classification performance of hybrid deep learning models, their inherent black-box opacity wherein decision rationale is encoded in millions of non-interpretable floating-point weights presents a critical barrier to adoption in high-stakes forensic and legal contexts (Manthena et al., 2023, 2025). Explainable Artificial Intelligence (XAI) encompasses a family of post-hoc and intrinsic methods designed to produce human-interpretable explanations of model decisions. Among these, SHAP (SHapley Additive Explanations) has become the dominant model-agnostic explanation framework due to its solid game-theoretic foundation in cooperative Shapley values, which guarantees consistency, local accuracy, and missingness properties that alternative methods such as LIME do not uniformly satisfy (Vuran Sari & Acı, 2025).

The SHAP Kernel Explainer is a model-agnostic variant that treats the model as a black box and estimates Shapley values through weighted linear regression over a background reference dataset, making it universally applicable to any differentiable or non-differentiable model including hybrid CNN-LSTM-Attention architectures (Cannarile et al., 2022). For each prediction, SHAP assigns a signed contribution score to every input feature in this case, each of the 100 API call positions quantifying the direction and magnitude of each feature's influence on the final malware classification probability. Local Waterfall Plot explanations decompose individual sample predictions into per-feature contributions, while aggregated Mean Absolute SHAP values across a representative subset of test samples produce a temporal importance profile that identifies execution stages associated with malicious behavior (Saqib et al., 2024; Yu et al., 2026). In the present study, this aggregation is computed on a stratified subset of 10 test samples rather than the full test set, as detailed in the Method section. This dual-level explanation capability contributes to model transparency, which is a necessary but not sufficient condition for forensic accountability; full legal admissibility additionally requires elements such as chain-of-custody documentation, reproducibility, model versioning, audit logging, explanation stability analysis, and expert validation, which fall outside the scope of SHAP-based interpretability alone (Galli et al., 2024).

Table 1 Summary of Previous Studies

Author	Method	Dataset	Results	Limitations
Akhtar & Feng (2022)	CNN-LSTM	IoT Malware Dataset	Accuracy $\approx 99\%$	No explainability; focuses on IoT malware
Saqib et al. (2024)	Survey of XAI	Literature Review	Comprehensive review of SHAP, LIME, and XAI methods	No proposed detection model
Aboaoja et al. (2022)	Survey	Literature Review	Reviewed malware detection approaches and research challenges	No proposed model or experimental evaluation.
Vuran Sarı & Acı (2025)	CNN-GRU + SHAP + LIME	Windows Static Malware Dataset	CNN-GRU achieved 99.37% accuracy	Uses static analysis without temporal attention
Najibi & Bidgoly (2024)	Explainable Deep Learning	CICAndMal2017	Improved robustness against unseen Android malware	Focuses on Android malware rather than Windows API call sequences

Building upon previous studies, this research proposes a CNN-LSTM-Attention model integrated with the SHAP Kernel Explainer for Windows malware detection using dynamic API call sequences. Unlike prior explainable deep learning models that primarily evaluate mobile threats such as (Najibi & Bidgoly, 2024) using CICAndMal2017 to improve robustness against unseen Android malware the proposed architecture addresses the complex temporal dynamics of Windows API call sequences while maintaining high forensic transparency. This enables the identification of influential API call positions while maintaining accurate malware classification, thereby providing more transparent decision-making and stronger support for digital forensic investigations, particularly on highly imbalanced datasets.

METHOD

The overall research workflow of the proposed framework is depicted in Figure 1. The methodology consists of four core phases: data preprocessing with class imbalance mitigation, feature learning using a hybrid CNN-LSTM-Attention architecture, model training with loss optimization, and forensic interpretability analysis using SHAP.

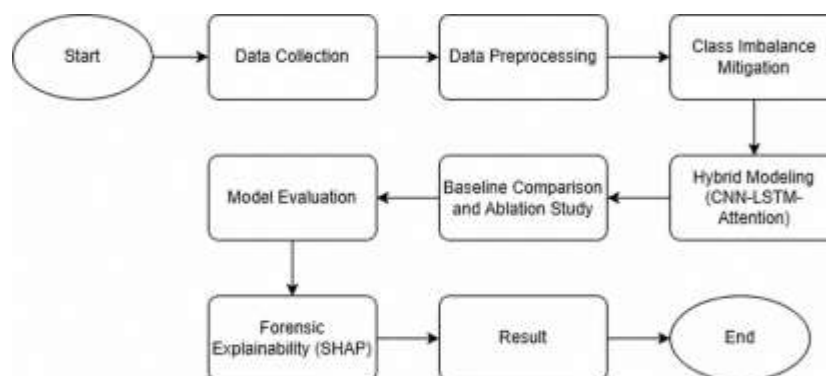


Figure 1 Research Flow

Data Collection

This study utilizes the benchmark dataset `dynamic_api_call_sequence_per_malware_100_0_306.csv` retrieved from Kaggle. The raw dataset initially comprised 43,876 dynamic execution logs of Windows applications captured in a controlled sandbox environment. Following a rigorous data sanitization process to eliminate duplicate sequences and prevent data leakage, the final working dataset was reduced to 13,206 unique operational behaviors. Each record captures a sequence of 100 API calls executed chronologically over time. The detailed statistical characteristics and structural profile of the sanitized dataset are summarized in Table 2.

*Puguh Hiskiawan



This is anCreative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Table 2

Dataset Specifications and Characteristics (Post-Sanitization)

Dataset Parameter	Metric / Value	Description & Operational Details
Total Unique Samples	13,206 executables	Total unique execution traces retained after removing 30,670 duplicates.
Malware Samples	12,593 (95.36%)	Majority class (Label = 1)
Benign Samples	613 (4.64%)	Minority class (Label = 0)
Class Imbalance Ratio	$\approx 20.5 : 1$	Severe imbalance ratio mitigated via augmented class weighting.
Sequence Length	100 steps	Time-series API call steps (t_0 to t_{99})
API Vocabulary Size	307 unique APIs	Encoded as discrete integer IDs ranging from 0 to 306.
Feature Dimensions	102 columns	1 Hash ID + 100 API Sequence Features + 1 Target Label.
Data Partitioning	80 : 20 Split	Stratified split yielding 10,564 train & 2,642 test samples with confirmed zero-overlap.

Data Preprocessing

The initial pipeline phase involves loading and structuring the dynamic API call sequence dataset. Each feature column (t_0 to t_n) represents a specific API Call ID invoked chronologically by a Windows executable during runtime. Prior to feature engineering, irrelevant identifiers, such as the file hash, were removed to isolate the behavioral sequence data, and the target labels (Malware vs. Benign) were separated. To guarantee rigorous model evaluation and address potential data leakage, duplicate API sequence features were explicitly removed prior to dataset splitting. This cleaning process reduced the dataset from 43,876 to 13,206 unique execution sequences (removing 30,670 redundant behaviors). A critical design decision was the handling of the categorical API call integer IDs (ranging from 0 to 306). Rather than applying One-Hot Encoding or continuous scaling, which would either produce a prohibitively sparse 307-dimensional matrix or destroy categorical representations, the integer IDs were preserved strictly as raw sequential inputs. This enables the deep learning model to utilize an Embedding Layer, which automatically learns dense 128-dimensional continuous vector representations for each unique API call during training, capturing implicit semantic relationships between system functions. Both the neural network pipeline and the post-hoc SHAP explainability calculations operate directly on these discrete integer-encoded API sequences. Finally, the preprocessed dataset was partitioned into training and testing sets using an 80:20 ratio. A stratified splitting technique was employed to ensure that both subsets meticulously preserved the original class distribution. To provide empirical proof against sequence leakage, an explicit zero-overlap verification was performed between the training set ($N = 10,564$) and the test set ($N = 2,642$). The intersection yielded absolutely zero overlapping sequences, ensuring that the held-out test evaluation represents 100% unseen execution behavior.

Class Imbalance Mitigation

The dataset exhibits a severe class imbalance, heavily skewed toward the malware class. To prevent training bias without altering the original data distribution (e.g., via SMOTE or undersampling which can disrupt sequence integrity), an algorithm-level cost-sensitive learning strategy was employed. Scikit-Learn's `compute_class_weight` function was initially utilized to calculate balanced heuristic weights. However, to strictly penalize the misclassification of the severely underrepresented minority class (benign), an empirical scaling multiplier of 3.0 was systematically applied to the benign class weight. These heavily penalized dynamic weights (Benign ≈ 61.0 , Malware ≈ 0.51) were integrated directly into the Binary Crossentropy loss function during the hybrid CNN-LSTM-Attention training phase. This cost-sensitive objective forces the model to focus equally on the structural behavioral patterns of both classes, ensuring robust detection without minority class starvation.

Hybrid Modelling CNN-LSTM-Attention

To simultaneously capture localized API call execution patterns and long-range chronological behavioral dependencies, this study proposes a hierarchical hybrid CNN-LSTM-Attention architecture. The model ingests each raw sequence of 100 integer API call IDs and first maps them into a 128-dimensional dense embedding space via an Embedding layer regularized by SpatialDropout1D (rate = 0.2). Local short-term n -gram execution patterns

*Puguh Hiskiawan



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

are then extracted through a CNN block comprising two stacked Conv1D layers (the first with 64 filters and kernel size = 5; the second with 128 filters and kernel size = 3, both using ReLU activation), a MaxPooling1D layer (pool size = 2), and Batch Normalization for stable gradient flow (Hiskawan et al., 2023). The resulting locally-enriched feature maps are subsequently fed into an LSTM layer (128 units, dropout = 0.2, return sequences = True) to model the chronological behavioral dependencies across the execution trace. A 4-head Multi-Head Attention mechanism with a key dimension of 128 and a residual connection, followed by Layer Normalization, is then applied on top of the LSTM outputs. This allows the model to contextualize each API call position relative to the full sequence before classification. The attended feature representations are aggregated through GlobalAveragePooling1D, followed by a fully connected Dense network (64 units, ReLU) that incorporates Batch Normalization and Dropout (rate = 0.3). The architecture concludes with a final Dense layer using a Sigmoid activation for binary classification (Benign = 0, Malware = 1). Model training incorporates EarlyStopping (patience = 10, monitor = 'val_loss') and ReduceLROnPlateau (factor = 0.2, patience = 5) callbacks to prevent overfitting and adaptively adjust the learning rate.

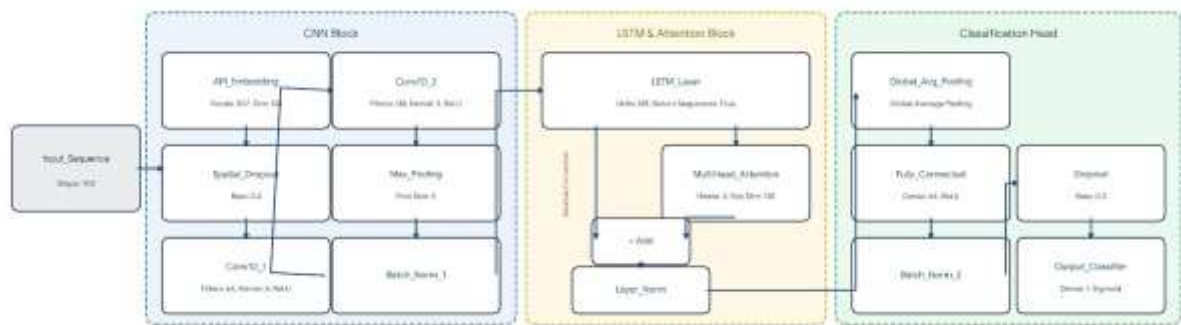


Figure 2 Proposed CNN-LSTM-Attention architecture

The proposed hybrid CNN-LSTM-Attention architecture processes sequences of 100 API call IDs by first mapping them into a 128-dimensional continuous space via an Embedding layer regularized by Spatial Dropout. Local short-term execution patterns are extracted using a CNN block composed of two stacked Conv1D layers, MaxPooling, and Batch Normalization. To capture chronological behavioral dependencies, these locally-enriched features are fed into an LSTM layer. A subsequent 4-head Multi-Head Attention mechanism, reinforced with a residual connection and Layer Normalization, dynamically highlights forensically critical API calls while mitigating vanishing gradients. Finally, the attended sequence is aggregated via Global Average Pooling and processed through a Dense network with Dropout, culminating in a Sigmoid classifier for binary malware detection.

Model Evaluation

Model performance is evaluated using Accuracy, Precision, Recall, F1-Score, and ROC-AUC. Due to severe class imbalance, relying solely on Accuracy can trigger the "accuracy paradox," where predictions are biased toward the majority malware class (Gyamfi et al., 2023). To ensure a balanced evaluation, Precision minimizes false positives (incorrectly blocking benign apps), while Recall measures detected malware to prevent critical false negatives. Ultimately, the F1-Score combines both into a harmonic mean, delivering a robust metric optimized for imbalanced dataset performance.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{(TP+FP)} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (4)$$

Baseline Comparison and Ablation Study

To validate the individual contribution of each architectural component and benchmark the proposed model against both classical machine learning and simpler deep learning baselines, a 5-fold stratified cross-validation was conducted across eight configurations: two classical baselines (Random Forest, XGBoost), four architectural ablations isolating individual components (CNN-only, LSTM-only, CNN+Attention, LSTM+Attention), one

*Puguh Hiskawan



This is anCreative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

ablation removing only the attention mechanism from the full pipeline (CNN-LSTM without Attention), and the proposed full CNN-LSTM-Attention model. All deep learning variants were trained with identical class weighting, batch size (128), and epoch count (5 epochs per fold) to ensure a fair comparison. Model performance was measured using Accuracy and ROC-AUC, reported as mean $\pm$ standard deviation across the five folds. Statistical significance between the proposed model and each baseline was assessed using a paired t-test on the per-fold ROC-AUC scores.

Forensic Explainability (SHAP)

To address the inherent black-box opacity of the hybrid deep learning model and produce forensically accountable decision explanations, the SHAP (SHapley Additive Explanations) Kernel Explainer is integrated as a post-hoc interpretability layer. SHAP is a model-agnostic, game-theoretic method grounded in cooperative Shapley values that assigns each feature (API call position) a quantitative contribution score to the final classification decision (Ali et al., 2023). A background reference dataset of 20 training samples is used to represent the model's baseline expected behavior, and SHAP values are computed for 10 representative test samples, selected via stratified sampling (5 malware, 5 benign) from correctly classified instances to ensure both classes are represented in the aggregated analysis.

Perturbed inputs generated by KernelExplainer are constructed by substituting entire feature values from either the background reference set or the explained instance never through continuous interpolation so every synthesized sequence retains valid, integer-valued API IDs within the original vocabulary (0–306), consistent with the embedding layer's discrete input requirement. However, because this substitution operates independently at each temporal position, synthesized sequences may combine positions from two distinct original execution traces, and are therefore not guaranteed to represent temporally coherent, realistic API call behavior.

Local explanations are generated via Waterfall Plots for two representative, correctly classified instances one malware sample and one benign sample both drawn from the same 10-sample stratified subset, decomposing each prediction probability into signed per-feature contributions and revealing which specific API call time steps pushed the decision toward Malware or Benign. Aggregated temporal significance is mapped by computing the Mean Absolute SHAP value across this same 10-sample subset for each of the 100 API call positions, producing a ranked temporal importance profile that identifies the chronological execution stages most critical to malware detection. To assess whether the attention mechanism provides interpretability signals consistent with SHAP, mean attention weights extracted from the trained Multi-Head Attention layer were computed for the same 10-sample subset and compared against this aggregated SHAP importance profile across temporal positions. Because KernelExplainer is computationally expensive, this analysis is restricted to a small representative subset rather than the full 8,776-sample test set; the resulting importance profile should therefore be read as an indicative, exploratory pattern rather than a statistically generalized characterization of the entire test population. This dual-level explanation framework enables forensic analysts to independently validate model decisions, focus investigation on the highest-risk behavioral windows, and produce transparent, auditable explanatory artifacts to support though not by themselves constitute evidence for legal proceedings (Hermosilla et al., 2025).

RESULT

This section presents the quantitative and qualitative outcomes of training, evaluating, and interpreting the proposed hybrid CNN-LSTM-Attention model on the Windows API call sequence dataset.

Training Convergence

Figure 3 illustrates the training and validation accuracy and loss curves across the 30 training epochs. Both the training and validation accuracy curves rise steadily during the initial epochs, rapidly converging to approximately 95% to 96%. While minor validation fluctuations occur notably around epochs 8 and 16 due to the inherent complexity of sequential Windows API traces and class imbalance constraints the general alignment between the training and validation curves indicates that the model generalizes reasonably well without severe, persistent overfitting. Correspondingly, the training and validation loss curves decline sharply and stabilize at lower values, culminating in a final training loss of approximately 0.11 and validation loss around 0.13. The overall stabilization of the optimization process confirms that the proposed CNN-LSTM-Attention architecture effectively learns the underlying behavioral patterns of the API sequences.

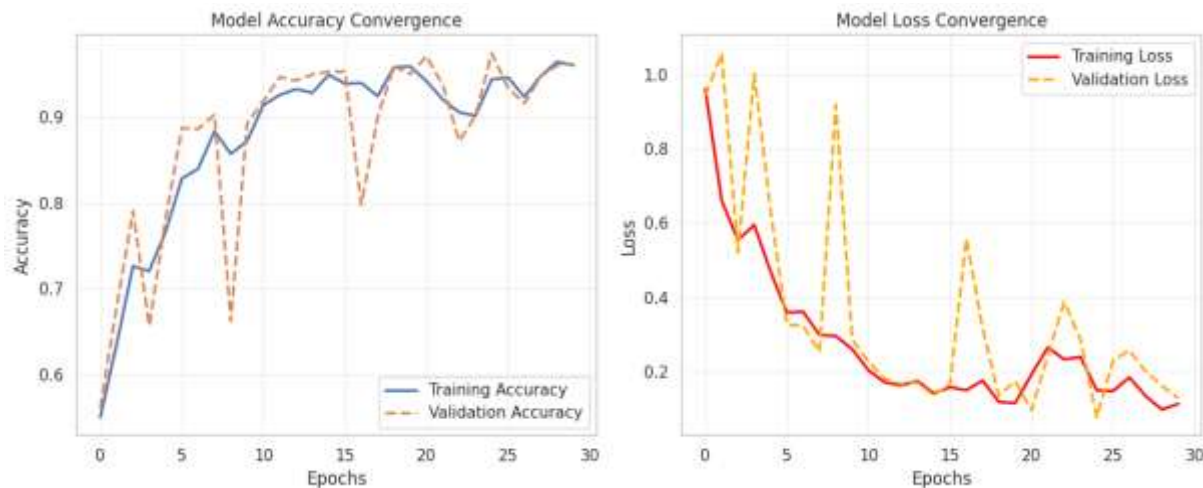


Figure 3 Training Accuracy & Loss Curves

Classification Performance & Evaluation Metrics

Figure 4 illustrates the Receiver Operating Characteristic (ROC) curve for the proposed malware detection model. The model achieved an outstanding Area Under the Curve (AUC) score of 0.9663. This high AUC-ROC value confirms the model's robust discriminative capability, demonstrating its proficiency in accurately distinguishing between malicious and benign executables across varying classification thresholds.

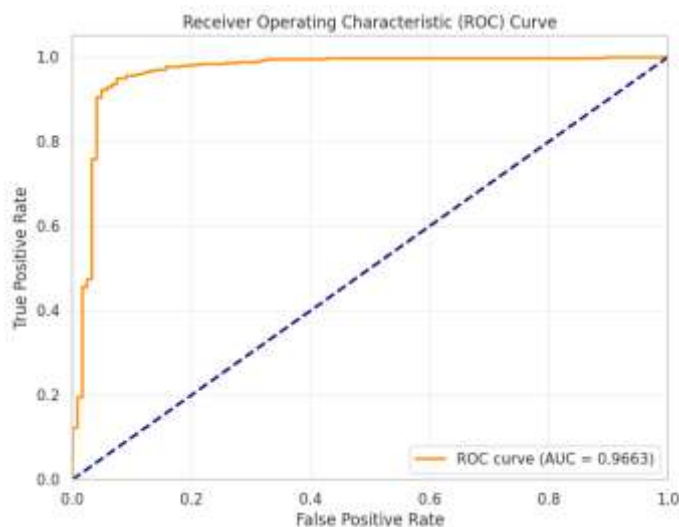


Figure 4 Receiver Operating Characteristic (ROC) Curve

Table 3 Classification Report

Class / Metric	Precision	Recall	F1-Score
Benign (0)	0.77	0.71	0.74
Malware (1)	0.99	0.99	0.99
Accuracy			0.98
Macro Avg	0.88	0.85	0.86
Weighted Avg	0.98	0.98	0.98

Table 3 presents the detailed classification performance of the proposed model on the hold-out test set using the optimal decision threshold of 0.14. The model achieves an overall accuracy of 98%. Specifically, for the Malware (Class 1) category, the model demonstrates exceptional efficacy with a precision, recall, and F1-score of

*Puguh Hiskiawan



This is anCreative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

0.99. Meanwhile, for the minority Benign (Class 0) category, it yields a precision of 0.77, a recall of 0.71, and an F1-score of 0.74, effectively balancing classification performance despite the dataset's class imbalance.

Table 4 Extended Performance Metrics derived from the Confusion Matrix.

Metric	Value	Interpretation & Forensic Relevance
PR-AUC Score	0.9974	Near-perfect precision-recall performance on the minority malware class.
Balanced Accuracy	0.8855	Unbiased overall accuracy, accounting for class distribution balance.
Macro-F1 Score	0.8655	Balanced trade-off between Precision and Recall across all classes.
Matthews Corr. Coeff. (MCC)	0.7323	Strong positive correlation between predicted and actual binary labels.
Specificity (TNR)	0.7869	Proportion of benign software correctly identified as non-malicious.
False Positive Rate (FPR)	0.2131	Rate of benign applications misclassified as malware (false alarms).
False Negative Rate (FNR)	0.0159	Rate of malware bypassing detection (critical security threat).

As summarized in Table 4, a comprehensive evaluation beyond standard accuracy reveals the practical reliability of the proposed model in a real-world cybersecurity context. Notably, the architecture achieves an exceptionally low False Negative Rate (FNR = 0.0159), indicating that only 1.59% of malicious sequences successfully evaded detection. In threat analysis, minimizing false negatives is paramount, as undetected malware poses a significantly higher operational risk than false alarms. Additionally, with a Specificity of 0.7869 and a Balanced Accuracy of 0.8855, the model maintains a reliable baseline for benign identification. Furthermore, the high PR-AUC score (0.9974) and Matthews Correlation Coefficient (MCC = 0.7323) demonstrate that the model maintains robust discrimination capability without suffering from majority-class bias or inflated performance metrics.

5-Fold Cross-Validation, Baseline Comparison, Ablation Study, and Statistical Significance

As synthesized in Table 5 and illustrated in Figure 5, the Proposed Model (Full) achieved superior discriminative performance across 5-fold cross-validation (ROC-AUC: 0.9562 ± 0.0186 , Accuracy: $95.78 \pm 1.05\%$). Ablation analysis confirms that Multi-Head Attention is indispensable; removing it caused the most severe performance drop (ROC-AUC: 0.9073, Accuracy: 86.18%) and high fold variance ($\pm 9.66\%$). Among isolated deep learning components, CNN-only (ROC-AUC 0.9531) outperformed LSTM-only (0.9356), showing that local $\$n\$$ -gram API patterns provide stronger discriminative signals than unweighted temporal sequences. While classical baselines like Random Forest and XGBoost achieved high accuracy and lower variance, the proposed architecture better optimized the ROC-AUC trade-off under class imbalance. Finally, the stability analysis in Figure 6 highlights the full model's consistent high performance (~ 0.9600 median ROC-AUC) across all folds without catastrophic failures.

Table 5 5-Fold Cross-Validation Performance Comparison (Mean $\pm$ SD)

Model Configuration	ROC-AUC	Accuracy (%)
Proposed Model (Full)	0.9562 ± 0.0186	95.78 ± 1.05
CNN Only	0.9531 ± 0.0160	95.65 ± 0.04
CNN + Attention	0.9504 ± 0.0195	92.63 ± 2.71
Random Forest	0.9465 ± 0.0049	95.80 ± 0.15
XGBoost	0.9391 ± 0.0062	96.02 ± 0.14
LSTM + Attention	0.9344 ± 0.0089	89.72 ± 5.14
Ablation (No Attention)	0.9073 ± 0.0220	86.18 ± 9.66
LSTM Only	0.9356 ± 0.0135	95.60 ± 0.43

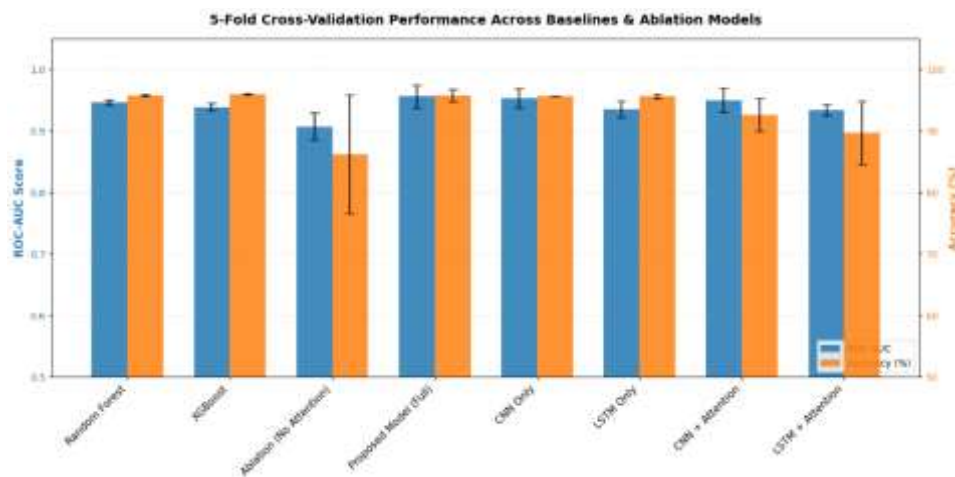


Figure 5 Performance comparison of baseline and ablation models.

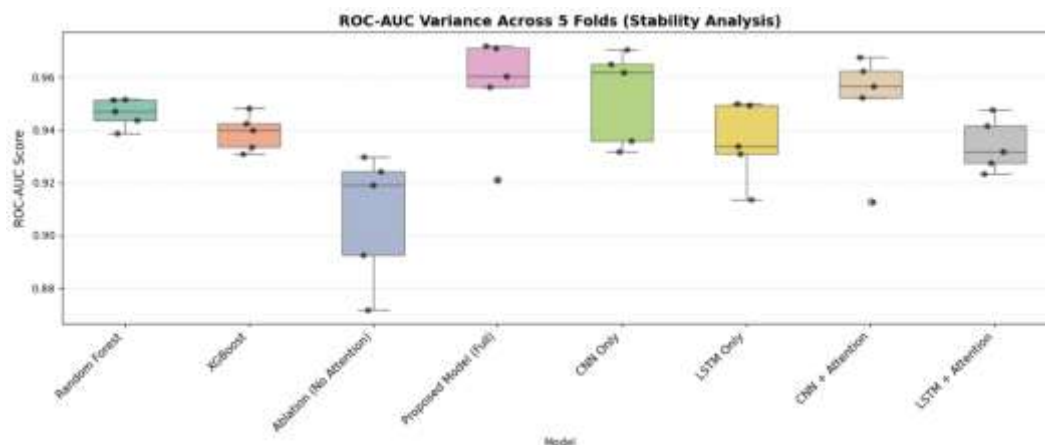


Figure 6 ROC-AUC variance across folds

Significance Analysis

To evaluate whether the performance differences between the proposed architecture and competing strategies are statistically significant, a paired samples t-test was conducted across the five folds. The analysis reveals that the performance advantage of the Proposed Model over the non-attention ablation (Ablation: No Attention) is statistically significant ($t = 5.005$, $p = 0.0075$), confirming the critical role of the attention mechanism. Meanwhile, compared to classical baselines like Random Forest ($t = 1.382$, $p = 0.2390$) and XGBoost ($t = 2.208$, $p = 0.0919$), the differences are not statistically significant ($p \geq 0.05$), indicating that while our deep learning model achieves competitive discriminative capability, it performs on par with strong ensemble baselines in terms of stability across folds.

Local Forensic Explainability (SHAP)

The SHAP waterfall plots illustrate the local feature contributions for individual test predictions, demonstrating how specific API call sequence positions influence the final classification verdict. For the malware sample, the model correctly classifies the instance with a high confidence score of $f(x) = 0.997$, (starting from an expected value $E[f(X)] = 0.997$), influenced by specific API temporal features such as $172 = t2$ and subsequent execution steps. Conversely, for the benign sample, the model successfully classifies the instance as safe, driving the final prediction down to a low probability of $f(x) = 0.012$ from the baseline expectation of 0.997 . This reduction is heavily pulled downward by negative contributions from key temporal positions, most notably $141 = t50$ (-0.19), $257 = t22$ (-0.18), $114 = t24$ (-0.17), and $208 = t17$ (-0.15). Ultimately, these local explanations provide forensic analysts with transparent, auditable evidence to pinpoint the exact temporal execution steps dictating model verdicts.

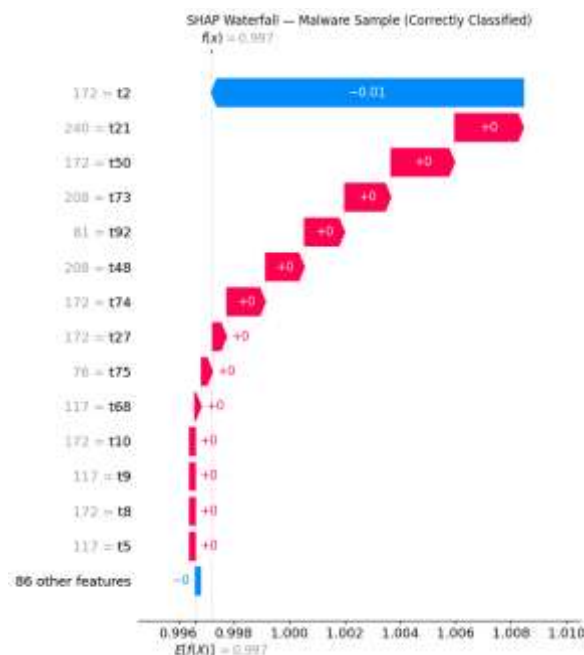


Figure 7 SHAP waterfall plot for a correctly classified malware sample

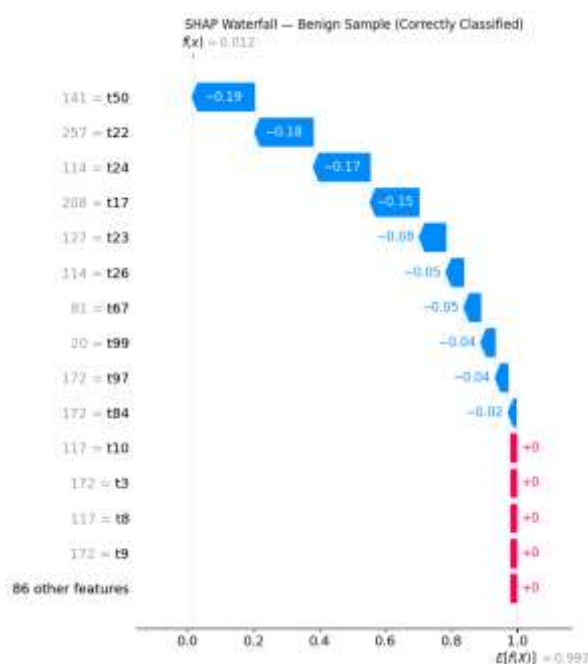


Figure 8 SHAP waterfall plot for a correctly classified benign sample

Aggregated Temporal Importance Analysis (SHAP)

To evaluate the relative importance of each execution step within the sampled subset of test data, aggregated feature attribution was computed by taking the mean absolute SHAP value ($\text{mean}(|\text{SHAP value}|)$) across the 10 representative samples for each of the 100 API call positions. As illustrated in Figure 9, the resulting importance ranking identifies specific temporal windows that drive the model's malware classification logic. Specifically, the execution step t1 emerges by far as the most influential feature within the sampled subset, carrying the highest average impact on model output magnitude (exceeding 0.05). Additional prominent temporal windows include t50, t75, and t21 t23, which also exhibit significant mean absolute SHAP values. This indicates that behavioral

*Puguh Hiskiawan



This is anCreative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

anomalies occurring during these specific chronological stages of program execution particularly at the onset and mid-to-late execution phases are critical indicators for differentiating malicious software from benign applications, providing digital forensic investigators with targeted temporal windows for rigorous code auditing.

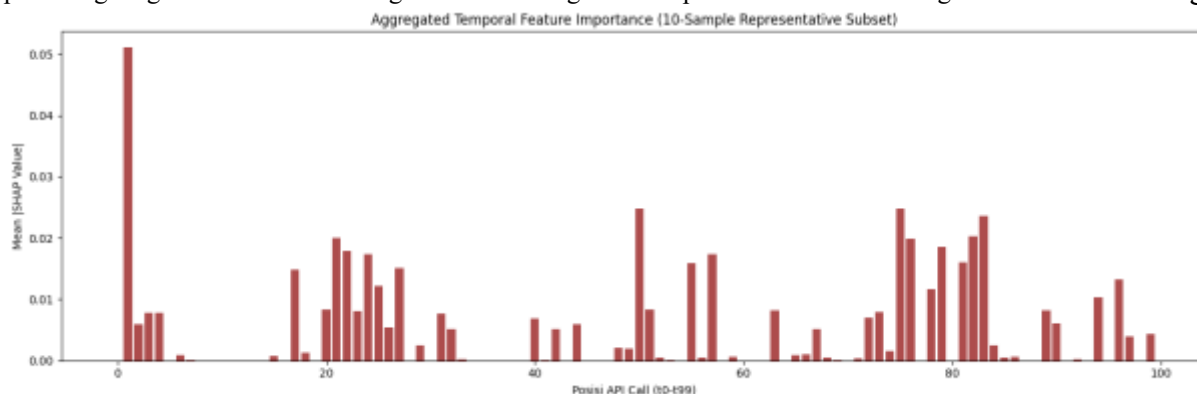


Figure 9 Aggregated Temporal Feature Importance

Attention Weights vs SHAP

To validate whether the internal representation learned by the model's attention mechanism aligns with post-hoc feature attributions, the mean attention weights extracted from the Multi-Head Attention layer were compared against the aggregated absolute SHAP values across the temporal positions. Figure 9 illustrates this comparative profile, where the blue bars represent the attention weights and the red line depicts the absolute SHAP values derived from the 10-sample representative subset. The comparison highlights the distinct perspectives offered by both interpretability layers. While the attention mechanism captures the model's internal weighting across sequence steps, the SHAP values provide a more granular attribution that isolates specific critical execution windows (such as the sharp peak observed at index 1). Examining these two dimensions allows forensic analysts to cross-verify model reliability, confirming that the high-risk timestamps flagged by post-hoc feature contributions are structurally acknowledged by the network's internal attention flow.



Figure 10 Attention Weights vs. SHAP

DISCUSSIONS

The empirical findings confirm that the proposed hybrid architecture successfully resolves the dual challenges of feature extraction and severe class imbalance (95.36% vs. 4.64%) in dynamic Windows malware detection. Achieving a mean accuracy of 95.78% and a mean ROC-AUC of 0.9562 via 5-fold cross-validation validates the synergy between Conv1D filters (capturing local n-grams), LSTM layers (modeling long-range dependencies), and Multi-Head Attention (weighting critical behavioral transitions) (Bensaoud & Kalita, 2024; Hiskiawan et al., 2023). Ablation analysis provided definitive proof of structural contributions; stripping the attention mechanism triggered a statistically significant drop in ROC-AUC to 0.9073 ($t = 5.005$, $p = 0.0075$). Although classical baselines (Random Forest, XGBoost) performed competitively in raw metrics ($p \geq 0.05$), the hybrid network

*Puguh Hiskiawan



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

excelled in handling class skew and providing intrinsic sequence awareness. Furthermore, integrating SHAP Kernel Explainer bridged the gap between black-box prediction and forensic auditability (Hiskiawan, Geasela, et al., 2025). By isolating critical temporal checkpoints t_1 , t_{50} , and t_{75} the framework arms analysts with prioritized audit targets, though operational constraints such as SHAP computational overhead and the fixed 100-step window must be managed in practical deployments.

CONCLUSION

This study established an interpretable hybrid deep learning framework integrating Conv1D, LSTM, and Multi-Head Attention for dynamic Windows malware detection, validated through rigorous 5-fold cross-validation. The architecture successfully achieved a mean accuracy of 95.78% ($\pm 1.05\%$) and a mean ROC-AUC of 0.9562 (± 0.0186), outperforming non-attention variants with statistical significance ($p < 0.05$). By bridging temporal neural networks with SHAP-based model-agnostic explainability, this research delivers three core contributions: robust performance under severe class skewness (95.36% vs. 4.64%), forensically transparent decision-making, and precise mapping of critical temporal execution checkpoints t_1 , t_{50} , and t_{75} to streamline code auditing. Future work will extend Application Programming Interface tracking windows to capture delayed payloads and evaluate model adaptability against zero-day variants.

ACKNOWLEDGMENT

The author would like to express sincere gratitude to the academic advisors and supervisors for their invaluable guidance, insight, and support throughout the course of this research. Deep appreciation is also extended to the Faculty of Technology and Design, Bunda Mulia University, for providing the facilities, resources, and institutional environment necessary to successfully complete this study.

REFERENCES

- Aboaoja, F. A., Zainal, A., Ghaleb, F. A., Al-rimy, B. A. S., Eisa, T. A. E., & Elnour, A. A. H. (2022). Malware Detection Issues, Challenges, and Future Directions: A Survey. *Applied Sciences (Switzerland)*, 12(17). <https://doi.org/10.3390/app12178482>
- Akhtar, M. S., & Feng, T. (2022). Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time. *Symmetry*, 14(11). <https://doi.org/10.3390/sym14112308>
- Ali, S., Abuhmed, T., El-Sappagh, S., Muhammad, K., Alonso-Moral, J. M., Confalonieri, R., Guidotti, R., Del Ser, J., Díaz-Rodríguez, N., & Herrera, F. (2023). Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. *Information Fusion*, 99(April), 101805. <https://doi.org/10.1016/j.inffus.2023.101805>
- Bensaoud, A., & Kalita, J. (2024). CNN-LSTM and transfer learning models for malware classification based on opcodes and API calls. *Knowledge-Based Systems*, 290. <https://doi.org/10.1016/j.knosys.2024.111543>
- Cannarile, A., Dentamaro, V., Galantucci, S., Iannacone, A., Impedovo, D., & Pirlo, G. (2022). Comparing Deep Learning and Shallow Learning Techniques for API Calls Malware Prediction: A Study. *Applied Sciences (Switzerland)*, 12(3). <https://doi.org/10.3390/app12031645>
- Catak, F. O., Yazi, A. F., Elezaj, O., & Ahmed, J. (2020). Deep learning based Sequential model for malware analysis using Windows exe API Calls. *PeerJ Computer Science*, 6, 1–23. <https://doi.org/10.7717/PEERJ-CS.285>
- Galli, A., La Gatta, V., Moscato, V., Postiglione, M., & Sperli, G. (2024). Explainability in AI-based behavioral malware detection systems. *Computers and Security*, 141(March). <https://doi.org/10.1016/j.cose.2024.103842>
- Gyamfi, N. K., Goranin, N., Ceponis, D., & Čenys, H. A. (2023). Automated System-Level Malware Detection Using Machine Learning: A Comprehensive Review. *Applied Sciences (Switzerland)*, 13(21). <https://doi.org/10.3390/app13211908>
- Hermosilla, P., Díaz, M., Berríos, S., & Allende-Cid, H. (2025). Use of Explainable Artificial Intelligence for Analyzing and Explaining Intrusion Detection Systems. *Computers*, 14(5), 1–25. <https://doi.org/10.3390/computers14050160>
- Hiskiawan, P., Chen, C.-C., & Ye, Z.-K. (2023). Processing of electrical resistivity tomography data using convolutional neural network in ERT-NET architectures. *Arabian Journal of Geosciences*, 16(10), 1–14. <https://doi.org/10.1007/s12517-023-11690-w>
- Hiskiawan, P., Geasela, Y. M., Heryanto, H., Stephanie, E., Ardianti, M., & Sukarno, F. A. (2025). Trustworthy Data Science Framework for Non-Invasive Nutritional Screening Using Computer Vision. 2025 *International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, 1743–



1748. <https://doi.org/10.1109/ICIMCIS68501.2025.11327268>
- Hiskiawan, P., Sari, M. K., Siregar, R. E., Valencia, N. A., & Wijayanti, T. P. (2026). A Deep Learning Data Fusion Approach for Prostate MRI Zonal Segmentation Using Dual-Channel U-Net with T2 and ADC Images. *2026 International Conference on Current Research in Artificial Intelligence and Data Science (ICCRAIDS)*, 1, 1–7. <https://doi.org/10.1109/ICCRAIDS67816.2026.11519671>
- Hiskiawan, P., Wijayanti, T. P., Heryanto, H., Everlin, S., Yasodhara, S. A., & Alexander, D. (2025). Mel-Frequency Cepstral Coefficients and Neural Networks for Indonesian Traditional Music Recognition. *2025 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, 1707–1712. <https://doi.org/10.1109/ICIMCIS68501.2025.11327436>
- Li, C., Lv, Q., Li, N., Wang, Y., Sun, D., & Qiao, Y. (2022). *Computers & Security A novel deep framework for dynamic malware detection based on API sequence intrinsic features*. 116. <https://doi.org/10.1016/j.cose.2022.102686>
- Liang, J., Shen, J., Wang, P., Liang, F., & Deng, X. (2026). Dynamic Malware Detection Method Based on API Multiple Subsequences. *Computers, Materials and Continua*, 87(1). <https://doi.org/10.32604/cmc.2025.073076>
- Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2023). API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques. *Journal of Network and Computer Applications*, 218(June), 103704. <https://doi.org/10.1016/j.jnca.2023.103704>
- Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2024). A systematic literature review on Windows malware detection: Techniques, research issues, and future directions. *Journal of Systems and Software*, 209(December 2023), 111921. <https://doi.org/10.1016/j.jss.2023.111921>
- Manthena, H., Kimmel, J. C., Abdelsalam, M., & Gupta, M. (2023). Analyzing and Explaining Black-Box Models for Online Malware Detection. *IEEE Access*, 11(March), 25237–25252. <https://doi.org/10.1109/ACCESS.2023.3255176>
- Manthena, H., Shajarian, S., Kimmell, J. C., Abdelsalam, M., Khorsandroo, S., & Gupta, M. (2025). Explainable Artificial Intelligence (XAI) for Malware Analysis: A Survey of Techniques, Applications, and Open Challenges. *IEEE Access*, 13(April), 61611–61640. <https://doi.org/10.1109/ACCESS.2025.3555926>
- Najibi, M., & Bidgoly, A. J. (2024). *ep rin ot pe er r ev ie we d Pr t n ot pe er weki*. 1–19.
- Palma Salas, M., & de Geus, P. L. (2024). Deep Learning Applied to Imbalanced Malware Datasets Classification. *Journal of Internet Services and Applications*, 15(1), 342–359. <https://doi.org/10.5753/jisa.2024.3907>
- Saqib, M., MahdaviFar, S., Fung, B. C. M., & Charland, P. (2024). A Comprehensive Analysis of Explainable AI for Malware Hunting. *ACM Computing Surveys*, 56(12). <https://doi.org/10.1145/3677374>
- Vuran Sari, N., & Acı, M. (2025). A hybrid CNN-GRU model with XAI-Driven interpretability using LIME and SHAP for static analysis in malware detection. *PeerJ Computer Science*, 11. <https://doi.org/10.7717/peerj-cs.3258>
- Wang, P., Lin, T., Wu, D., Zhu, J., & Wang, J. (2024). TTDAT: Two-Step Training Dual Attention Transformer for Malware Classification Based on API Call Sequences. *Applied Sciences (Switzerland)*, 14(1). <https://doi.org/10.3390/app14010092>
- Yao, Y., Zhu, Y., Jia, Y., Shi, X., Zhang, L., Zhong, D., & Duan, J. (2024). Research on Malware Detection Technology for Mobile Terminals Based on API Call Sequence. *Mathematics*, 12(1). <https://doi.org/10.3390/math12010020>
- Yousuf, M. I., Anwer, I., Riasat, A., Zia, K. T., & Kim, S. (2023). Windows malware detection based on static analysis with multiple features. *PeerJ Computer Science*, 9, 1–29. <https://doi.org/10.7717/PEERJ-CS.1319>
- Yu, B., Zhang, W., Liu, Y., Ahmad, A., Obidallah, W. J., Yousef, A., Ullah, F., Yoshigoe, K., & Zhao, Y. (2026). Explainable AI for malware analysis: a systematic review of benchmark datasets, traffic-oriented detection, and interpretability methods. *PeerJ Computer Science*, (5), 1–59. <https://doi.org/10.7717/peerj-cs.3902>
- Zhang, S., Wu, J., Zhang, M., & Yang, W. (2023). Dynamic Malware Analysis Based on API Sequence Semantic Fusion. *Applied Sciences (Switzerland)*, 13(11). <https://doi.org/10.3390/app13116526>

