Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.077928

ARTICLE

WAFDect: A Malware Detection Model Based on Multi-Source Feature Fusion

Xian Wu, Liang Wan*, Jingxia Ren and Bangfeng Zhang

College of Computer Science and Technology, Guizhou University, Guiyang, China

*Corresponding Author: Liang Wan. Email: lwan@gzu.edu.cn

Received: 19 December 2025; Accepted: 06 March 2026; Published: 08 May 2026

ABSTRACT: Traditional malware detection models rely on a single feature source for detection, resulting in high false positive or false negative rates due to incomplete information. In addition, conventional models depend on manual feature engineering, which is inefficient and hard to adapt to new malware variants. To address these challenges, this paper proposes a malware detection model called WAFDect based on a self-attention mechanism with multi-source feature fusion. The model consists of two key designs. First, we construct a multi-source feature extraction model that analyzes multi-source data such as API call sequences, registry operation logs, file operation logs, and network behavior logs, capturing malware characteristics at multiple abstraction levels and building a global representation of maliciousness, thereby overcoming the problem of single feature sources in traditional models. Second, to address the heterogeneity of multi-source features in terms of dimension, scale, and semantics, we design a feature alignment module based on attention weights. This module can dynamically learn the association strength between different feature modalities and achieve semantic alignment and adaptive fusion of cross-modal features through a weighting allocation mechanism, effectively reducing the reliance on manual feature engineering in traditional methods. The experimental results indicate that WAFDect achieved excellent detection performance on the Speakeasy (trainset) and Avast-CTU_Small datasets, with accuracies of 0.9229 and 0.9878, respectively. Compared with traditional detection models, this method shows significant improvements in key metrics such as accuracy and F1 score, thereby validating its effectiveness.

KEYWORDS: Malware detection; multi-source feature fusion; dynamic analysis; self-attention mechanism

1 Introduction

Malware is one of the most serious threats in cyberspace. According to Verizon's 2025 Data Breach Investigations Report [1], the global economic loss caused by malware attacks exceeds hundreds of billions of dollars annually. It is noteworthy that modern malware is capable of continuously generating new variants and is extremely adept at evading detection. Research by Abuadbba et al. [2] found that a large number of samples contained variations that were highly structurally similar but had completely different hashes. Malware detection technology has undergone an evolution from rule-based methods to machine learning and then to deep learning. However, these detection techniques rely on a single source of features and are therefore unable to comprehensively capture the behavioral patterns of malware [3], still facing challenges posed by malicious software [4].

Major breakthroughs in the field of artificial intelligence have provided new methods for malware detection. Saxe and Berlin [5] proposed a feature fusion detection model that combines byte-entropy histograms, PE metadata, and API call patterns, significantly improving detection accuracy. Similarly, Chen

et al. [6] achieved semantic-level feature fusion by enhancing the semantic information of API call sequence parameters and injecting the specific behavioral semantics of individual samples. MPDroid [7] constructs a unified feature representation space by extracting characteristics such as permissions, components, API calls, system call sequences, and network traffic, achieving deep integration of multi-source features. Chaganti et al. [8] proposed a multi-view, multi-source feature fusion detection model by constructing perspectives from bytecode sequences, grayscale images, and behaviors. These studies indicate that the synergistic analysis of multi-source features can more comprehensively characterize the behavior patterns of malware, thereby effectively addressing complex threats. Despite the significant progress achieved by multi-source feature fusion techniques, the following challenges still exist:

- Traditional malware detection models rely on manual feature design or are constrained by the execution environment [9], resulting in a single feature source and limited representational capacity. Consequently, they struggle to comprehensively capture the behavioral characteristics of malware [10] and are difficult to adapt to continuously evolving new types of malware and their variants.
- Heterogeneous data such as API call sequences, registry operation logs, file operation logs, and network behavior logs present issues of dimensionality, scale, and semantic differences. Direct integration can easily lead to information loss, and the dimensional discrepancies of different features may affect model convergence [11].

To address the above issues, this study proposes a malware detection model based on multi-source feature fusion, with the main design including:

- Multi-source feature extraction: A structured behavioral representation is introduced that systematically parses dynamic malware analysis reports and organizes heterogeneous behavioral events (API calls, registry operations, file system activities, and network behaviors) into modality-specific feature streams. This design enables feature extraction at multiple abstraction levels and provides a comprehensive representation beyond single-source analysis.
- Feature Fusion Mechanism: A novel attention-based feature alignment and fusion mechanism is developed to explicitly address cross-modal heterogeneity in dimensionality, scale, and semantic granularity. The WAF module learns modality-specific importance weights and performs adaptive semantic alignment, enabling effective cross-source feature integration without manual feature engineering.

The remainder of this paper is structured as follows: Section 2, Related Work. We systematically review traditional single-feature source detection models and multi-source feature fusion models, and highlight the shortcomings of existing work. Section 3, WAFDect Design. We present the overall architecture of the model and provide a detailed explanation of the proposed WAFDect model. We first introduce the data processing and feature extraction modules for multi-modal data. Then, we focus on describing the feature fusion module based on a cross-modal attention mechanism, which can adaptively learn the relationships between features from different modalities. Section 4, Experiments and Results Analysis. To verify the effectiveness of WAFDect, we designed comprehensive experiments. This section first introduces the datasets, evaluation metrics, and baseline models used for comparison; subsequently, ablation experiments are conducted to validate the contribution of each component in the model, and comparative experiments demonstrate that WAFDect outperforms existing mainstream models in both detection accuracy and F1 score. Section 5, Conclusion. We have summarized the research work presented in this paper.

2 Related Work

Traditional single-feature source detection models can be divided into static analysis and dynamic analysis. With the continuous increase in the number and complexity of malware, they have become insufficient to address the increasingly complex cyber-security threats. Malware detection technology based on multi-source feature fusion aims to build a more comprehensive and robust detection model by integrating features from multiple sources. However, the introduction of multi-source features also brings challenges in heterogeneous fusion. Different feature sources inherently differ in terms of dimensions, scales, and semantic levels, and simple feature fusion methods struggle to fully exploit the deep correlations across modalities, potentially even causing mutual interference between features, thereby limiting further improvements in model performance.

Static analysis focuses on the composition and structural characteristics of files. Static analysis detects threats by extracting quantifiable features or function call relationship graphs from files, such as PE header information, string characteristics, and n-gram opcodes. Griffin et al. [12] proposed a model for automatically extracting string features to generate malware signatures, but this model relies on common static features among samples and has limited capability in detecting polymorphic and variant malware. Santos et al. [13] proposed a feature extraction model based on opcode sequences, applying data mining and machine learning to the detection of unknown malware variants. However, it suffers from significant loss of semantic information. Static analysis models do not require code execution and can identify malware before it is executed. However, they are susceptible to techniques such as code obfuscation, packing, and virtualization protection, which can easily lead to false positives.

The core of dynamic analysis is to run the target file in a controlled environment and, by monitoring and capturing dynamic information such as behavioral characteristics, resource interactions, and system impacts during software execution, determine whether it possesses malicious attributes. Wang et al. [14] detected malicious communication behaviors by analyzing the textual semantic features in network traffic, overcoming the limitation of traditional models that rely on manually designed features. However, their approach shows insufficient adaptability to encrypted and novel malicious traffic. The core value of dynamic analysis models lies in their ability to penetrate static obfuscation techniques such as code obfuscation, packing, and encryption, allowing for direct observation of the malware's "true intentions". However, dynamic analysis is constrained by specific environments and conditions and consumes more resources.

The multi-source feature fusion technology aims to integrate features from different data sources in order to obtain more comprehensive and accurate information. In the field of malware detection, a single feature often cannot comprehensively describe the behavior and characteristics of malware, which can easily lead to inaccurate or missed detection results. Studies by Kim et al. [15] and Mc Laughlin [16] indicate that by integrating multiple features, attention mechanisms can dynamically weight different feature modalities to achieve detection. Chen et al. [17] utilized Graph Attention Networks (GAT) to extract structural and semantic features of Class-level Call Graphs (CSCG), overcoming the limitations of insufficient single-feature representation capability and traditional fusion models neglecting deep semantic correlations, yet it still heavily relies on the quality of feature construction. Trung et al. [18] addresses the issue of insufficient deep fusion of multimodal features in traditional models by building an anti-obfuscation multimodal feature system, designing a deep multimodal fusion architecture, and introducing adversarial training, but its behavior tracing capability still has limitations. Current research on feature fusion has made significant progress in improving detection accuracy; however, its anti-interference capability is generally insufficient. Model performance heavily depends on feature engineering. Although attention mechanisms and graph networks have been introduced, there remains a limitation in adequately mining the deep, nonlinear semantic interactions between features of different modalities.

In recent years, researchers have gradually enhanced detection performance by integrating multimodal behavioral features with sequence modeling techniques (Table 1). Nazim et al. [19] proposed an integrated deep neural network framework for malware classification by combining various heterogeneous features. Experimental results indicate that multimodal feature fusion provides superior detection performance compared to single-modal detection models. However, this approach primarily employs a fixed feature fusion strategy and does not explicitly model the semantic alignment relationships between different feature modalities. Similarly, Arrowsmith et al. [20] validated the complementarity between different modal features by combining various static and dynamic features. However, their feature fusion method remains relatively coarse-grained and lacks an adaptive weighting mechanism for heterogeneous feature sources. On the other hand, some studies focus on malware detection models based on behavioral sequences. Owoh et al. [21] propose a hybrid GRU–GAN model for modeling API call sequences to enhance the characterization of malware temporal behavior characteristics. Although sequence modeling methods can effectively capture fine-grained behavior patterns, they often rely solely on a single feature source and cannot fully reflect the overall behavioral characteristics of malware. Youssef et al. [22] using a Transformer-based API call sequence modeling approach, achieved favorable results in the task of dynamic malware detection, while also validating the advantages of the attention mechanism in behavior modeling. Unlike the studies mentioned above, the WAFDect model proposed in this paper integrates multi-source behavioral features, including API call sequences, registry operation logs, file operation logs, and network activity logs, within a unified attention mechanism framework. By designing a weighted attention feature fusion module, the model can dynamically learn the correlation strength between different feature modalities, achieving cross-modal semantic alignment and adaptive fusion. This reduces reliance on manual feature engineering and more comprehensively characterizes the multi-level behavioral features of malware.

Table 1: Recent advances in malware detection.

Work	Feature Source	Fusion Strategy	Attention	Multi-Source Alignment
Nazim et al. (2025) [19]	Multimodal	Fixed fusion	No	No
Arrowsmith et al. (2025) [20]	Multimodal	Concatenation	No	No
Owoh et al. (2024) [21]	API only	–	RNN	No
Transformer-based [22]	API only	–	Yes	No
WAFDect (ours)	API + Registry + File + Network	Weighted Attention Fusion	Yes	Yes

3 WAFDect Design

As shown in Fig. 1, WAFDect consists mainly of four components: (A) the application dynamic analysis module, (B) the data processing module, (C) the multi-source feature extraction module, and (D) the weighted attention fusion (WAF) module and classifier module. First, the original application enters the sandbox to obtain a behavior report t, after processing, the report yields a sequence representation containing temporal behaviors $t'_{\log}$ and an attribute representation based on statistical aggregation t'_{other}, they are respectively fed into different feature extractors; then, the extracted features are integrated through the feature fusion module WAF; finally, the fused features are sent to the classifier to obtain the detection results y.

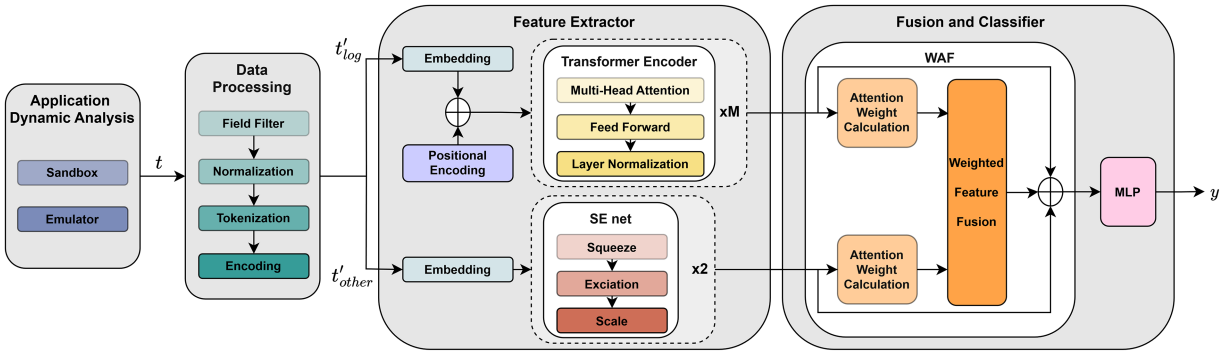


Figure 1: Overview of WAFDect.

3.1 Multi-Source Feature Data Processing Module

The main role of data cleaning in malware detection technology based on multi-source feature fusion is to transform the original noisy heterogeneous data, such as API call sequences, registry operation logs, file operation logs, and network behavior logs, into high-quality, structured, and unified feature representations. This is accomplished through data cleansing, field filtering, tokenization, and feature encoding and standardization, as detailed below.

3.1.1 Data Cleaning

The raw behavior reports generated by dynamic analysis sandboxes typically contain critical behavioral information such as the sequence of API calls, registry operation logs, file operation logs, and network activity logs of the sample in a controlled environment. However, raw reports often contain issues such as data noise, heterogeneous formats, information redundancy, and outliers, and may include invalid information such as environment-dependent paths, temporary files, differences in multi-threaded execution order, and ineffective calls triggered by anti-sandbox code. These issues will directly affect the performance of feature extraction and detection models. Therefore, regarding the characteristics of malware behavior reports, the following aspects are emphasized: the completeness of key behavior sequences such as API call chains and registry operation sequences; verification of a standardized behavior report structure, with the report content including core modules such as API call sequences, file operation logs, and network behavior logs; removal of environmental information irrelevant to behavior analysis; and the truncation of files exceeding 10 GB. The cleaned data is shown in Fig. 2.

3.1.2 Field Filtering

Field filtering, as an important component of feature engineering, can extract multi-dimensional features from behavior reports in JSON format generated by sandbox execution. By selecting effective information and eliminating redundant data, it provides high-quality input data for subsequent tasks, thereby enhancing the performance of detection models. The dynamic characteristics of malware can accurately reflect its behavioral patterns; however, due to the high dimensionality and noise in the data, it is necessary to implement effective field filtering mechanisms. At this stage, the following field filtering strategy is adopted: retain fields directly related to malicious behavior, such as API call sequences, registry operation logs, file operation logs, and network activity logs, while discarding irrelevant metadata fields; merge duplicate or highly similar fields, such as those for TCP and UDP; and impose length limits on fields containing a large number of entries, such as API call sequences, taking the first 200 records to balance information

completeness and computational efficiency. By applying targeted field selection and feature construction, key information related to malicious behavior is retained, avoiding interference from redundant data.

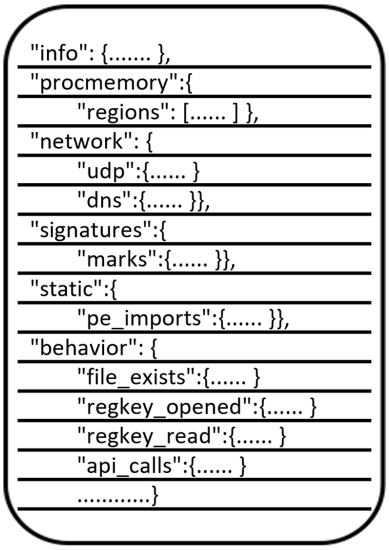


Figure 2: Sample report of post-cleaning activities.

3.1.3 Tokenization

WAFDect employs a tokenization approach centered on rule-driven and statistical mapping. It designs customized token extraction strategies for different types of features, such as labels, operation logs, APIs, and paths, ultimately converting unstructured or semi-structured data into numerical vectors suitable for model input. For example, a registry path can be split into multiple tokens using backslashes: path\to\key → ["path", "to", "key"], and a fixed-length vector for the path can be generated through hashing and hierarchical weighting. For IP addresses (such as 192.168.1.1:80), the address can be divided into IP segments and port, with features extracted to indicate whether it is private, multicast, etc., generating a fixed-length vector.

3.1.4 Feature Encoding and Standardization

For malware detection scenarios, WAFDect extracts features from multiple dimensions, including API call sequences, registry operation logs, file operation logs, and network behavior logs, and encodes these features into a unified tensor representation, providing a standardized input for subsequent malware classification models. For different types of features, WAFDect implements differentiated and independent processing flows, employing customized encoding and standardization strategies according to the characteristics of each feature, thereby achieving a comprehensive representation of the feature space. As shown in Table 2, the core module design includes:

Table 2: Design of core module for feature encoding and standardization.

Feature Source	Encoding & Standardization Strategy
API Call Sequences	• Structured Analysis • Word Frequency Statistics Mapping and Integration • API Integration + Attribute Features
Network Behavior	• Network behavior encoding

(Continued)

Table 2 (continued)

Feature Source	Encoding & Standardization Strategy
	• Statistical feature construction
	• Yeo-Johnson transformation
Memory Operations	• RobustScaler Robust Standardization
	• Quantile Transformation
File Operations	• One-hot encoding of operation types
	• Path keyword matching
Registry Operations	• One-hot encoding of operation types
	• Path-level weighted hash
Signature Tags	• TF-IDF Weighting
	• Truncated SVD Dimensionality Reduction

3.2 Multi-Source Feature Extraction Module

The primary function of feature extraction in malware detection technology based on multi-source feature fusion is to distill computable discriminative features from the cleaned raw data that can represent the intrinsic behavior and structural characteristics of malware. This process transforms the data into a unified and efficient numerical representation, providing inputs with higher information density and stronger discriminative ability for subsequent feature fusion and classification modules. The process is carried out through embedding and positional encoding, as well as multi-source feature extraction layers, as detailed below. Step 1: Each data source is independently encoded, resulting in multiple tensors such as API call sequences, registry operation logs, file operation logs, and network behavior logs. Step 2: Features with temporal information are embedded and combined with positional encoding before being fed into the Transformer encoder module to obtain outputs; other discrete features are embedded and then passed through the t-SE module to obtain outputs.

3.2.1 Embeddings and Positional Encoding

The embedding operation maps input integer sequences, such as the Token ID of an API call, into a high-dimensional continuous vector space to capture semantic and contextual relationships. Given the input sequence $x = [x_1, x_2, \ldots, x_N]$, where $x_i \in \mathbb{Z}$, represents the token index in the vocabulary, the embedding layer performs the following transformation through a learnable embedding matrix $E \in \mathbb{R}^{|V| \times d_e}$:

$$e_i = E(x_i) \cdot \sqrt{d_e} \tag{1}$$

in Eq. (1), $e_i \in \mathbb{R}^{d_e}$ is the embedding vector of the i-th token, and d_e is the embedding dimension. The scaling factor $\sqrt{d_e}$ is used to scale the embedding values to alleviate gradient instability in deep networks [23]. Output as embedding sequence $e = [e_1, e_2, \ldots, e_N]$, where each vector x_i encodes the semantic information of the corresponding token. Since the self-attention mechanism of the Transformer inherently lacks the ability to perceive sequence order, positional information must be explicitly injected. WAFDect uses sinusoidal-cosine positional encoding, which is defined as follows:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_e}}}\right); PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_e}}}\right) \tag{2}$$

in Eq. (2), pos is the position of the token in the sequence ($pos \in [1, N]$), i is the index of the embedding dimension ($i \in [1, \frac{2}{d_e}]$), and the wavelengths range from 2π to $10000 \cdot 2\pi$ in a geometric series to cover positional relationships at different scales. The resulting positional encoding matrix $\mathbf{PE} \in \mathbb{R}^{N \times d_e}$ is added to the embedding sequence to obtain the output with integrated positional information:

$$e'_{pos} = e_{pos} + PE_{pos} \tag{3}$$

3.2.2 Multi-Source Feature Extraction Layer

This layer is designed with differentiated feature extraction modules for different types of input features. As shown in Fig. 1, the layer consists of a Transformer encoder module and a t-SE module.

- The Transformer encoder module handles $t'_{\log}$ and is mainly composed of linear projection and Transformer encoders. Linear projection is the mapping of input dimensions to target dimensions, as specifically shown in Eq. (4):

$$x_{proj} = W_{proj} \cdot x + b_{proj} \tag{4}$$

in Eq. (4), $W_{proj} \in \mathbb{R}^{d_{out} \times d_{in}}$ is the projection matrix, d_{in} is the input dimension, and d_{out} is the target dimension.

The Transformer encoder module consists of 8 layers of Transformer Encoder Layers, each containing multi-head self-attention (with the input divided into 8 heads) and a feedforward network. Calculating attention weights:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

in Eq. (5), Q, K and V serve as the query, key, and value, respectively, and are used as the inputs to the self-attention layer, while d_k represents the dimensionality of the key. The feedforward network structure consists of two layers of linear transformations followed by a *ReLU* activation:

$$FFN(x) = \max(0, x \cdot W_1 + b_1) \cdot W_2 + b_2 \tag{6}$$

The feature vector at the final time step integrates information from the entire sequence; therefore, this module takes the final time step:

$$h_{trans} = \left(x_{proj} + PE\right)[-1, :, :] \tag{7}$$

as the output feature vector.

- The t-SE module handles t'_{other} and enhances key features through channel attention. It is mainly composed of convolutional layers, SE blocks [24], and a global pooling output module. See Fig. 3a for details.

The convolutional layer compresses high-dimensional inputs to a uniform dimension, extracts local patterns from normalized features, and introduces nonlinear transformations through the ReLU activation function:

$$X_{conv} = ReLU\left(Conv1D_{3\times1}(X_{in})\right) \tag{8}$$

in Eq. (8), X_{in} serves as the input data. The SE attention block consists of two SE blocks. Each SE block contains three stages: Squeeze, Excitation, and Scale. In the Squeeze stage, the input feature $X \in \mathbb{R}^{B\times C\times L}$ undergoes a global average pooling operation to capture the global information of the channels:

$$z_c = \frac{1}{L}\sum_{i=1}^{L} X_{c,i}, \forall c \in \{1, 2, \ldots, C\} \tag{9}$$

in Eq. (9), t represents the t-th time step along the sequence dimension.

The Excitation stage uses a bottleneck structure, consisting of two fully connected layers:

$$s_c = Sigmoid\left(W_2 \cdot ReLU\left(W_1 \cdot z_c\right)\right) \tag{10}$$

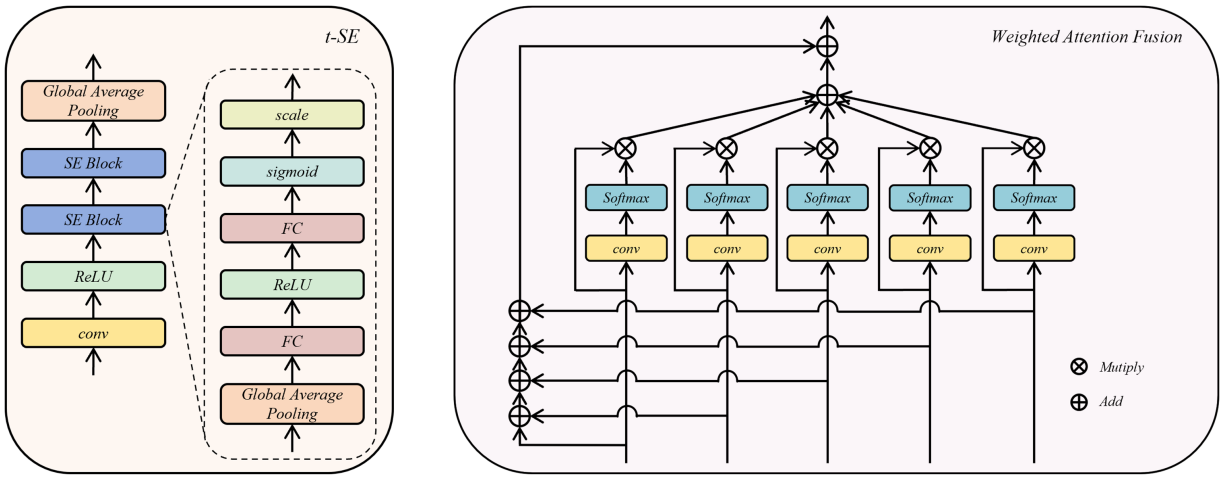
in Eq. (10), $W_1 \in \mathbb{R}^{\frac{output_{dim}}{r}\times output_{dim}}$ is the weight of the first fully connected layer, and $W_2 \in \mathbb{R}^{output_{dim}\times\frac{output_{dim}}{r}}$ is the weight of the second fully connected layer. In the Scale stage, the weights of the Excitation output are regarded as the importance of each feature channel after feature selection, and then they are applied to the previous features through channel-wise multiplication to accomplish the recalibration of the original features along the channel dimension:

$$\check{x}_{c,t} = s_c \cdot x_{c,t} \tag{11}$$

The global pooling output module produces the output:

$$y_c = \frac{1}{L}\sum_{t=1}^{L} x_{c,t} \tag{12}$$

in Eq. (12), $y \in \mathbb{R}^{B\times output_{dim}}$.



(a) t-SE Module Structure Diagram.

(b) Weighted Attention Fusion (WAF) Module Structure Diagram.

Figure 3: Multi-source feature extraction module: (**a**) t-SE module structure; (**b**) weighted attention fusion (WAF) module structure.

3.3 Weighted Attention Fusion (WAF) Module and Classifier Module

The primary function of the weighted attention fusion and classifier module is to integrate key information from multi-source features, ultimately making accurate classification decisions based on the fused feature representation. This module consists of a weighted attention fusion module and a multilayer perceptron classifier. Specifically, the weighted attention fusion module, as illustrated in Fig. 3b, calculates attention weights using a combination of convolution and softmax. By dynamically adjusting the weights, it enhances important features while suppressing less significant ones. To prevent the attention mechanism from excessively suppressing potentially useful information, feature concatenation and residual connections are introduced. This module employs a data-driven attention weight allocation strategy, which can adaptively identify and highlight key features in different scenarios. By combining weighted fusion with concatenation mapping, it not only preserves the dominance of important features but also retains the potential information of secondary features, thereby achieving a comprehensive representation of features. The specific process is as follows.

1. **Feature stacking:** Stacking individual feature vectors along a new dimension to form a feature matrix:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}_1' \\ \boldsymbol{h}_2' \\ \boldsymbol{h}_3' \\ \boldsymbol{h}_4' \\ \boldsymbol{h}_5' \end{bmatrix} \in \mathbb{R}^{5\times 64}, \ (N = 5) \tag{13}$$

in Eq. (13), h_i' respectively represents the results obtained from the feature extraction module for API call sequences, registry operation logs, file operation logs, network behavior logs, and other data, while H represents the matrix obtained after concatenating the features.

2. **Channel compression and weight generation:** Cross-feature information interaction and dimensionality reduction are achieved through convolution, and the compression results are normalized into attention weights using the softmax function:

$$\boldsymbol{W} = Softmax(Conv(\boldsymbol{H})) \tag{14}$$

3. **Weighted Feature Fusion:** Performing a weighted summation of the feature matrix with the attention weights:

$$\boldsymbol{h}_{attn} = \boldsymbol{H}^T \cdot \boldsymbol{w} = \sum_{i=1}^{5} w_i \cdot \boldsymbol{h}_i' \in \mathbb{R}^{64} \tag{15}$$

4. **Feature Concatenation:** Concatenating N feature vectors along the feature dimension:

$$\boldsymbol{h}_{cat} = \boldsymbol{h}_1' \oplus \boldsymbol{h}_2' \oplus \boldsymbol{h}_3' \oplus \boldsymbol{h}_4' \oplus \boldsymbol{h}_5' \in \mathbb{R}^{256}, \ (N = 5) \tag{16}$$

in Eq. (16), $\oplus$ represents the feature dimension concatenation operation.

5. **Linear Mapping and Residual Fusion:** The concatenated features are mapped back to a 64-dimensional space through a linear layer and form a residual connection with the attention-weighted features:

$$\boldsymbol{h}_{fused} = \boldsymbol{h}_{attn} + Linear(\boldsymbol{h}_{cat}) \tag{17}$$

in Eq. (17), $Linear(\cdot)$ represents a linear transformation:

$$Linear(\boldsymbol{h}_{cat}) = \boldsymbol{W}_{cat} \cdot \boldsymbol{h}_{cat} \tag{18}$$

in Eq. (18), $\boldsymbol{W}_{cat} \in \mathbb{R}^{64\times256}$ and $\boldsymbol{h}_{cat} \in \mathbb{R}^{256}$ are learnable parameters.

6. The mathematical significance of residual connections can be expressed as ($N = 5$):

$$\boldsymbol{h}_{fused} = \underbrace{\sum_{i=1}^{5} w_i \cdot \boldsymbol{h}_i'}_{\text{Attention-Weighted Features}} + \underbrace{W_{cat} \cdot (\boldsymbol{h}_1' \oplus \boldsymbol{h}_2' \oplus \boldsymbol{h}_3' \oplus \boldsymbol{h}_4' \oplus \boldsymbol{h}_5') + b_{cat}}_{\text{Concatenate Feature Maps}} \tag{19}$$

4 Experimental Results and Analysis

4.1 Dataset

***Speakeasy* Dataset [25]:** This dataset was generated using the Windows kernel emulator Speakeasy v1.5.9 and contains behavioral reports for approximately 93,500 samples, covering both benign and malicious behaviors, all presented in JSON format. Malicious samples belong to seven different types of malware, and the distribution of samples under each label is detailed in Table 3. Therefore, this dataset is suitable for research on malware detection and classification. This dataset specifically provides a test set (April 2022), which was collected at a different time from the training set (January 2022). Due to limitations of the experimental equipment, in this study, only the training set portion was used for experiments, and the training, validation, and test sets were re-divided.

Table 3: Structure and size of the *Speakeasy (trainset)* dataset.

Label	Backdoor	Clean	Coinminer	Dropper	Keylogger	Ransomware	Rat	Trojan	Windows_ syswow64	Total
Count	11062	24434	6891	8243	4378	9627	1697	8733	236	75301
Size	1.11 GB	1.14 GB	462 MB	252 MB	261 MB	596 MB	84 MB	434 MB	1.5 MB	4.3 GB

***Avast-CTU_Small* Dataset [26]:** This dataset contains sandbox reports stored in JSON format, originating from an improved tool called CAPEv2, which is based on the Cuckoo Sandbox. The dataset includes approximately 400,000 samples, covering the period from January 2017 to January 2020. The distribution of samples across various labels is detailed in Table 4, involving ten different malware families (see Table 5 for details). Due to the lack of legitimate samples, this dataset is intended solely for malware classification tasks. Limited by device configurations, only the small version of the dataset was used in the experiments, with the training, validation, and test sets being restructured accordingly. Furthermore, the dataset lacks sequential information and only provides summaries of events collected by the sandbox.

Table 4: Structure and size of the *Avast-CTU_Small* dataset.

Label	Banker	Coinminer	Keylogger	Pws	Rat	Trojan	Total
Count	27463	655	29	4191	3343	13295	48976
Size	4.69 GB	53 MB	16 MB	960 MB	1.42 GB	2.33 GB	9.46 GB

Table 5: Distribution of malware families in the *Avast-CTU_Small* dataset.

Family	Adload	Emotet	HarHar	Lokibot	njRAT	Qakbot	Swisyn	Trickbot	Ursnif	Zeus	Total
Count	704	14429	655	4191	3372	4895	12591	4202	1343	2594	48976

***WAF-data* Dataset:** This dataset was collected from November 2024 to May 2025. The data consist of sandbox reports generated using Cuckoo v2.0.5 and are all stored in JSON format. During the preparation of the dataset, approximately 6000 original PE samples (sourced from online resources) were collected, including both legitimate software and Trojan malware (see Table 6).

Table 6: Structure and size of the *WAF-data* dataset.

Family	Benign	Trojan	Total
Count	1694	2029	3723
Size	61.6 GB	32.3 GB	93.9 GB

To eliminate any potential data leakage, all datasets were split strictly at the sample level using file hashes as unique identifiers. No duplicate hashes appear across training, validation, or test sets. All feature extraction and normalization steps were conducted independently on the training set and then applied to validation and test sets without recalibration.

4.2 Experimental Setup and Evaluation Metrics

Experimental Setup: The experiments were conducted on an NVIDIA GeForce GTX 1650, the calculation of costs and deployment considerations can be seen in Table 7. To accommodate hardware limitations, the batch size was set to 16 for the public dataset and 32 for the *WAF-data* dataset. The AdamW optimizer was used, with an initial learning rate of 1×10^{-4} and a weight decay of 1×10^{-5}. A learning rate scheduler was employed to dynamically adjust the learning rate, and the loss function was set to cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{N} \mathbf{y}_i \log\left(\hat{\mathbf{y}}_i\right) \tag{20}$$

in Eq. (20), $\mathbf{y}_i$ represents the true labels, and $\hat{\mathbf{y}}_i$ represents the model's predicted probabilities. Considering the class imbalance in the samples, all datasets were divided into training, validation, and test sets using stratified sampling, with a ratio of 7:1.5:1.5. Additionally, an early stopping strategy was implemented during the experiments, which was triggered when the validation loss did not improve for five consecutive epochs.

Table 7: Computational cost and deployment considerations.

Metric	Value
Model size	6.53 MB (float32)
Number of parameters	1.71 M
Training time (1 epoch)	10.5 min (GTX 1650)
Peak GPU memory	1.6 GB (estimated)
Inference latency	303 ms/sample

Evaluation Indicators: To comprehensively and objectively evaluate the performance of WAFDect, WAFDect uses four widely recognized evaluation indicators: accuracy, AUC value, true case rate TPR, and F1 score to measure the model's classification ability, robustness, and practicality from different perspectives.

4.3 Review of Detection Models

1. Neurlux [27]: This model does not perform any intervention during the data cleaning phase and directly passes the raw behavior reports to the feature extraction phase. The feature extraction phase primarily carries out simple whitespace tokenization and uses sequence encoding with a vocabulary size of V = 10,000. The modeling phase employs a combined model of one-dimensional convolution, LSTM, and conventional attention mechanisms.
2. Gated CNN [28]: This model retains only the API call data and performs customized feature engineering on each API call record during the feature extraction phase. These feature-engineered vector sequences are processed through a gated convolutional network.
3. Quo.Vadis [25]: This hybrid model simultaneously considers context-related features, static features, and dynamic features. During the data cleaning phase, Quo.Vadis retains only the API call names. In the feature extraction phase, each API call name is tokenized and encoded using a vocabulary containing 600 terms, which are then modeled using a one-dimensional convolutional neural network. Additionally, this study releases a comprehensive dataset that includes Speakeasy simulation reports.
4. CruParamer [6]: This model retains only the API call information from the original reports during the data cleaning phase. The feature extraction step employs a parameter-based API annotation and a sensitivity-inspired API embedding method. In the modeling phase, two separate networks based on 2D convolution and LSTM are used.
5. Nebula [10]: This model integrates features such as API calls, file operations, and network traffic through a self-attention mechanism, overcoming the limitations of traditional models that rely solely on single features and local modeling. During the data cleaning phase, domain knowledge-driven multimodal filtering and normalization address the issues of noise and sparsity in behavioral logs; in the feature extraction phase, a combination of tokenization and dynamic truncation balances semantic retention with computational efficiency.

In Table 8, we systematically summarize the processes of the aforementioned models [10]. A prevalent trend in current academic research is the use of traditional techniques, such as one-dimensional convolutional neural networks, sometimes combined with recurrent layers like long short-term memory (LSTM), as core modeling methods. Nevertheless, each model introduces unique procedures in data cleaning or feature extraction, thereby contributing to the diversification of malware analysis approaches.

Table 8: Malware analysis and modeling techniques.

Work	Data Processing	Feature Extractor	Model
Neurlux	×	Tokenization	CNN, LSTM, Attention
Gated CNN	API filter	Feature Hashing	CNN, LSTM
Quo.Vadis	API filter	Tokenization	CNN
CruParamer	API filter	API “labeling”	CNN, LSTM
Nebula	API, network, file, registry filters and normalization	Tokenization	Transformer (Self-Attention)

4.4 Comparison Research

In the comparative experiment section, this section will systematically compare and discuss the two tasks of malware detection and classification.

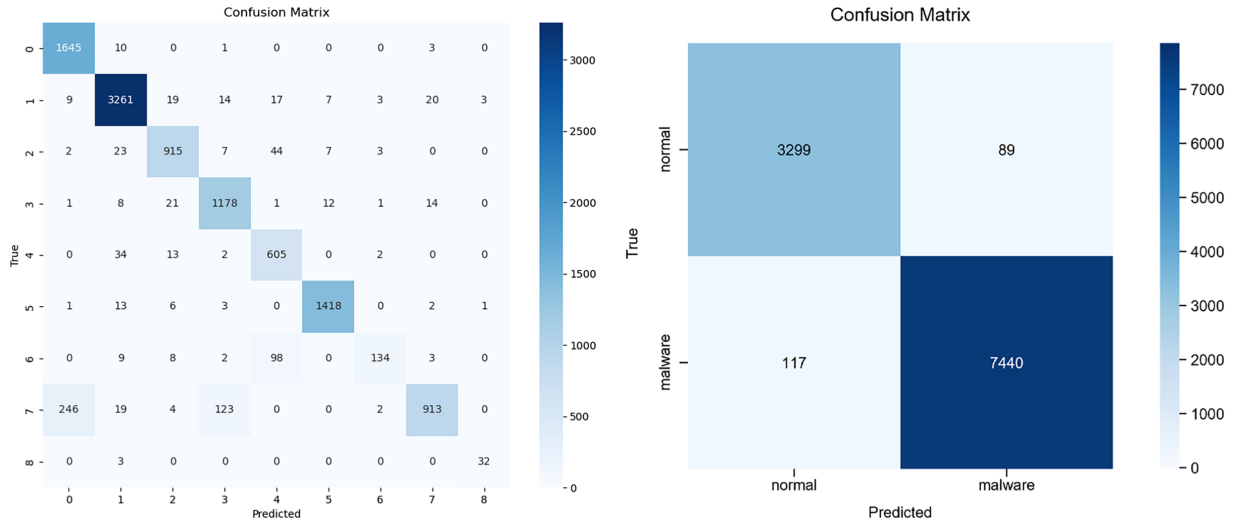
Malware Detection: This study compared four modeling techniques, including Neurlux proposed by Jindal et al., the Gated CNN model proposed by Zhang et al., Quo.Vadis released by Trizna, and Nebula.

WAFDect outperformed all competing architectures on the *Speakeasy (trainset)* dataset. The metrics on the *Speakeasy (trainset)* dataset are shown in Table 9. In addition, Fig. 4 presents the confusion matrix of WAFDect on the *Speakeasy (trainset)* dataset, and Fig. 5 shows the training history of WAFDect on the *Speakeasy (trainset)* dataset.

The comprehensive performance comparison on the *Speakeasy (trainset)* dataset is presented in Table 9. The performance metrics of baseline and comparative models are taken directly from the published results of their respective studies, ensuring consistency in evaluation under the same experimental setups. WAFDect demonstrates significant advantages across all key metrics: an accuracy of 0.9229, which is 2.22 percentage points higher than the second-best model, Nebula; an AUC value of 0.9943, approaching near-perfect classification performance; and notably, a breakthrough TPR of 0.8801, reflecting its malware detection capability, representing an improvement of over 54% compared to Nebula and more than 107% over the traditional benchmark model Neurlux. These results validate the effectiveness of multi-source feature fusion combined with the attention weight alignment mechanism—not only enhancing overall classification accuracy but also significantly improving the model's sensitivity to malicious behaviors through deep feature interactions, thereby addressing the shortcomings of existing models in detection rates under complex adversarial environments.

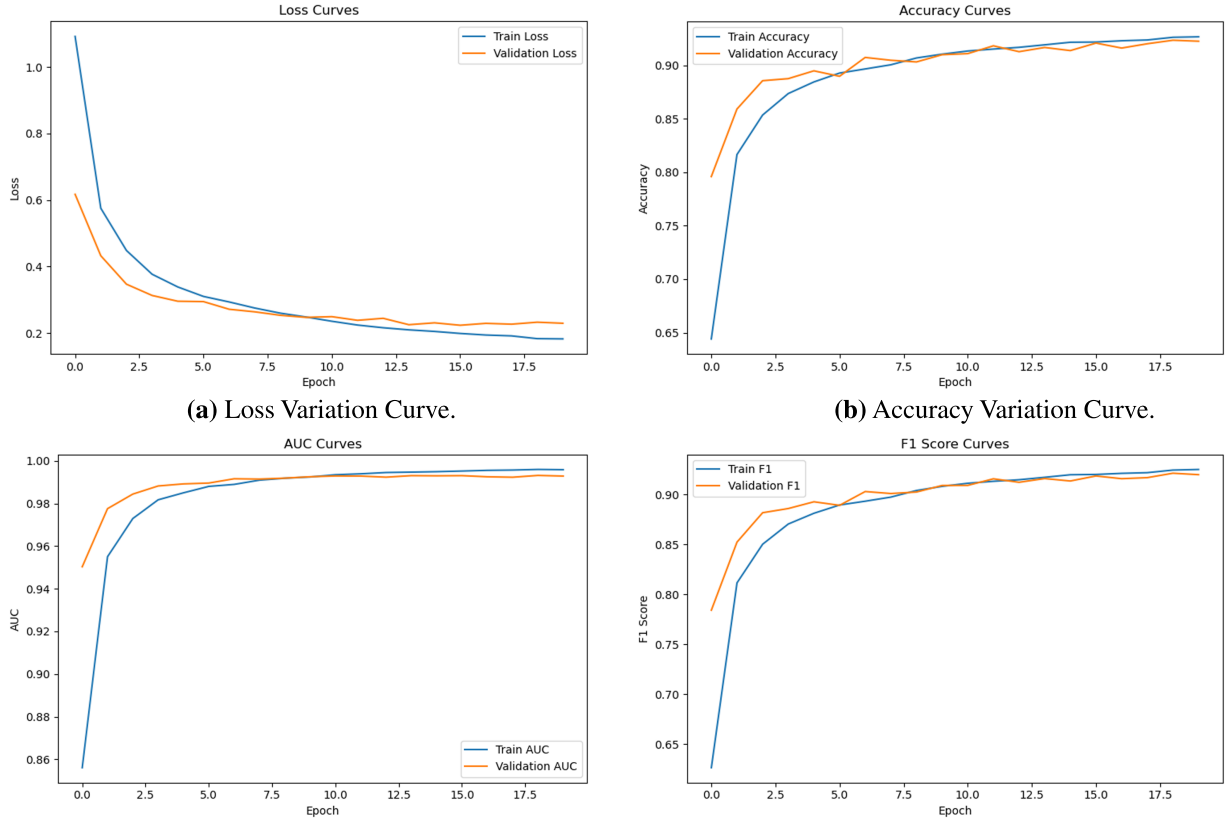
Table 9: Malware detection metrics on the *Speakeasy (trainset)* dataset.

Model	Architecture	Acc	AUC	F1	TPR
Neurlux	LSTM, Attention	0.8786	0.9528	0.8792	0.4250
Gated CNN	LSTM	0.7014	0.8879	0.6465	0.2152
Quo.Vadis	CNN	0.8173	0.9224	0.8065	0.3081
Nebula	Transformer (Multi-head Attention)	0.9053	0.9664	0.9058	0.5703
WAFDect	Transformer, WAF	0.9229± 0.0025	0.9949± 0.0006	0.9201± 0.0031	0.8602± 0.0026



(a) Confusion Matrix for Multiclass Classification Task. **(b)** Confusion Matrix for Binary Classification Task.

Figure 4: Confusion matrix of WAFDect on the *Speakeasy (trainset)* dataset: (**a**) confusion matrix for multiclass classification task; (**b**) confusion matrix for multiclass classification task.



(a) Loss Variation Curve.

(b) Accuracy Variation Curve.

(c) AUC Variation Curve.

(d) F1 Score Variation Curve.

Figure 5: Performance curves of WAFDect during training on the *Speakeasy (trainset)* dataset.

The confusion matrix analysis reveals the performance of WAFDect on both binary and multi-class tasks using the *Speakeasy (trainset)* dataset. Fig. 4a indicates that WAFDect achieves high accuracy in most individual classes, with false positives primarily occurring when class 7 (Trojan) is misclassified as class 0 (Backdoor) and class 1 (Clean) is misclassified as class 7 (Trojan). Notably, the number of false negatives for most classes remains in the single digits, indicating that the risk of missed detections is manageable. At the same time, major misclassifications are concentrated among classes with high feature similarity, which further validates WAFDect's effectiveness in class differentiation. When aggregated for a binary classification task, class 1 (Clean) and class 8 (windows_syswow64) are defined as benign, with all other classes considered malware. As shown in Fig. 4b, the false positive rate (FPR) in the binary confusion matrix is 1.55%, indicating a low probability of incorrectly labeling benign samples as malicious. Additionally, the false negative rate (FNR) is 2.63%, reflecting a balanced trade-off between reducing false alarms and missing actual threats.

Fig. 5 illustrates the performance evolution curves of the model trained on the *Speakeasy (trainset)* dataset, showing the trends of four key metrics: loss, accuracy, AUC, and F1 score for both the training and validation sets. As shown in Fig. 5a, the training loss decreases rapidly within the first 10 epochs (dropping from an initial value close to 1.0 to below 0.2), and then enters a stable convergence phase; the validation loss reaches its minimum (around 0.15) at approximately the 15th epoch and subsequently experiences minor fluctuations without significant rebound. This convergence pattern indicates that the model's optimization strategy effectively promoted loss reduction; the gap between training and validation loss remains small throughout, suggesting that the model did not exhibit noticeable overfitting and that the chosen number

of training epochs is reasonable. Training accuracy (Fig. 5b) stabilizes above 0.95 after 15 epochs, while validation accuracy converges to approximately 0.93, maintaining a difference within 2 percentage points, indicating good generalization capability of the model. Validation AUC (Fig. 5c) surpasses 0.95 around the middle of training (approximately the 10th epoch) and eventually reaches a stable plateau close to 1.0, demonstrating excellent discrimination ability between positive and negative samples. Validation F1 score (Fig. 5d) rises in parallel with accuracy, ultimately converging around 0.92, with the slope of its curve gradually flattening as training progresses, reflecting the model's incremental optimization in balancing precision and recall.

Malware Classification: For the multi-label task of predicting malware families, we present the F1 scores on the *Speakeasy (trainset)* and *Avast-CTU_Small* datasets in detail in Tables 10 and 11. Due to the lack of necessary sequential information in the *Avast-CTU_Small* dataset, Quo.Vadis and Gated CNN are omitted here, as both models have a high dependency on temporal information. Therefore, we only evaluated this dataset using Neurlux, Nebula, and WAFDect.

Table 10: Mean F1 score for malware classification on the *Speakeasy (trainset)* dataset.

Model	Backdoor	Clean	Coinminer	Dropper	Keylogger	Ransom ware	Rat	Trojan
Neurlux	0.8329	0.8453	0.6910	0.4488	0.2032	0.5527	0.6625	0.6153
Gated CNN	0.6870	0.7588	0.5586	0.2015	0.0794	0.3584	0.0000	0.5282
Quo.Vadis	0.8520	0.8338	0.4884	0.3580	0.2119	0.6861	0.1195	0.5359
Nebula	0.8548	0.8526	0.6303	0.2850	0.1295	0.7421	0.3683	0.6827
WAFDect	0.9234	0.9687	0.9210	0.9182	0.8515	0.9820	0.6717	0.8073

Table 11: Mean F1 score for malware classification on the *Avast-CTU_Small* dataset.

Model	Adload	Emotet	HarHar	Lokibot	njRAT	Qakbot	Swisyn	Trickbot
Neurlux	0.7150	0.9294	0.9031	0.8320	0.8479	0.9320	0.9991	0.9536
Nebula	0.697	0.9319	0.8363	0.9048	0.8896	0.9768	0.9984	0.9056
WAFDect	1.0000	1.0000	1.0000	0.9871	0.9700	1.0000	0.9995	0.9984

In the *Speakeasy (trainset)* dataset (Table 10), it should be noted that due to the very limited number of windows_syswow64 samples, they were merged into the Clean category in the classification experiments. WAFDect achieved an outstanding performance with a 0.9820 F1 score in Ransomware detection, representing a 32.2% improvement over Nebula. For the more challenging-to-detect Keylogger and Dropper, WAFDect achieved F1 scores of 0.8515 and 0.9182, respectively, surpassing the baseline models by over 300% and 200%, thereby addressing the shortcomings of traditional methods in detecting such covert threats.

In the *Avast-CTU_Small* dataset (Table 11), WAFDect achieved strong detection performance for 7 out of 8 modern malware families and reached the highest detection level of 0.9871 for the Lokibot family. Notably, WAFDect achieved zero false negatives in detecting advanced persistent threats such as Emotet, Qakbot, and Trickbot, demonstrating the model's robust representational capability for complex and obfuscated malware.

By integrating the classification results of two datasets, WAFDect's consistent advantages in multi-family malware detection tasks indicate that the multi-source feature fusion model not only enhances the overall malware identification capability but also effectively captures the unique behavioral patterns of different malware families through multi-source feature learning.

4.5 Ablation Research

To evaluate the effectiveness of the core components in WAFDect, we conducted systematic ablation experiments. The experiments aimed to investigate the impact of the number of Transformer Encoder blocks, the role of positional encoding in the t-SE module, and the placement of the weighted attention fusion module on the model's performance.

The baseline is as follows: the number of Transformer Encoder blocks is 8, the t-SE module does not include a positional encoding module, and the WAF module is placed after the Transformer Encoder module and the t-SE module. We keep all other hyperparameters and training settings fixed, and sequentially change the following three variables to observe their effects:

1. Variable A: Depth of the Transformer encoder. The number of Transformer Encoder blocks in the model was set to 4, 8, and 16 separately to study the impact of model depth on feature abstraction capability and performance. We hypothesize that deeper networks can capture more complex dependencies, but may also pose a risk of overfitting.
2. Variable B: Positional encoding in the t-SE module. In the t-SE module, adding or not adding a positional encoding module is used to verify whether explicit positional information can enhance the attention mechanism of the t-SE module when capturing positional relationships within sequences or channels.
3. Variable C: Integration position of the weighted attention fusion (WAF) module. Compare the two integration strategies of the WAF module. Strategy one involves integrating only the output of the Transformer Encoder module into the WAF module, followed by a simple concatenation with the t-SE module. Strategy two integrates the outputs of both the Transformer Encoder module and the t-SE module into the WAF module, aiming to explore the impact of the fusion timing on the model.

By combining these variables, we constructed multiple model variants for comparison. The experimental design matrix is shown in Table 12.

Table 12: Ablation study design matrix.

Experiment Number	Number of Transformer Encoder Blocks	Whether PE is Added before t-SE Moudle	WAF Module Integration Integration Strategy
Baseline	8	×	Strategy Two
A-1	4	×	Strategy Two
A-2	16	×	Strategy Two
B-1	8	✓	Strategy Two
C-1	8	×	Strategy One

All model variants were trained and evaluated on the same training/validation/test sets using the same evaluation protocol. We used the following metrics for performance comparison: accuracy, F1 score, AUC, and TPR, with the primary metric on the validation set serving as the basis for model selection. The metrics for each variant on the *Speakeasy (trainset)* dataset, *Avast-CTU_Small* dataset, and *WAF-data* dataset are presented in Table 13.

According to the results of the ablation experiments, the impact of each component on model performance has been systematically validated. In the comparison of the number of Transformer Encoder blocks, the 16-layer structure (A-2) demonstrated the best overall performance on the *WAF-data* dataset (Acc: 0.9349, TPR: 0.9770), indicating that a deep architecture provides significant benefits for complex feature extraction. The combination of positional encoding and the SE module (B-1) showed a slight improvement on

the *Avast-CTU_Small* dataset, but led to performance degradation on the other two datasets, demonstrating that explicit positional information can interfere with the channel attention mechanism. The weighted attention fusion module, placed after the Transformer Encoder blocks (C-1), remained stable in cross-dataset testing, but its TPR (0.8804) on the *Speakeasy (trainset)* dataset was lower than the baseline, suggesting that a late fusion strategy is more beneficial for maintaining detection sensitivity. Ultimately, the baseline model was identified as the optimal architecture due to its more balanced performance on the *Speakeasy (trainset)* and *Avast-CTU_Small* datasets (with AUC values all exceeding 0.987). Although the ablation matrix combines multiple architectural components, several observations can still be made regarding modality contributions. Configurations that remove cross-modal fusion or attention-based alignment consistently exhibit a larger performance drop compared to variants that only simplify the encoder depth. This indicates that performance gains are not solely due to increased model capacity, but primarily stem from effective multi-source feature integration.

Table 13: Ablation experiment results on the *Speakeasy(trainset)* dataset, the *Avast-CTU_Small* dataset, and the *WAF-data* dataset.

Model	*Speakeasy (trainset)*				*Avast-CTU_Small*				*WAF-data*			
	Acc	**AUC**	**F1**	**TPR**	**Acc**	**AUC**	**F1**	**TPR**	**Acc**	**AUC**	**F1**	**TPR**
Baseline	0.9229	0.9949	0.9201	0.8602	0.9878	0.9875	0.9057	0.8147	0.9168	0.9713	0.9258	0.9410
A-1	0.9254	0.9943	0.9232	0.9078	0.9914	0.9912	0.9098	0.8218	0.9060	0.9653	0.9150	0.9180
A-2	0.9227	0.9936	0.9203	0.9227	0.9804	0.9013	0.9803	0.8067	0.9349	0.9738	0.9430	0.9770
B-1	0.9081	0.9937	0.9082	0.9081	0.9916	0.9913	0.9095	0.8211	0.8825	0.9420	0.8957	0.9148
C-1	0.9186	0.9944	0.9180	0.8804	0.9891	0.9889	0.9077	0.8181	0.9349	0.9788	0.9406	0.9344

Previous studies have shown that malware detection models based on a single feature source [21,22], such as API call sequences or network traffic alone, are limited in their ability to capture comprehensive malicious behaviors. For instance, Transformer-based API sequence models achieve competitive performance on dynamic malware detection, but their reliance on a single modality restricts robustness against complex malware behaviors. By integrating multiple behavioral sources, WAFDect effectively addresses this limitation.

4.6 Discussion

Experimental results indicate that the fusion of multi-source features can significantly enhance malware detection performance. Compared to a single feature source, multimodal behaviors can provide a more comprehensive description of malicious patterns. The Weighted Attention Fusion (WAF) module introduced in WAFDect can adaptively model the importance of different feature modalities, achieving cross-modal feature interaction more effectively than fixed weights or simple concatenation strategies. Ablation experiments show that removing the attention fusion mechanism leads to a noticeable performance decline, suggesting that the performance improvement primarily stems from the fusion strategy itself rather than from increased network depth or parameter scale. The relatively high classification performance observed on certain datasets should be interpreted cautiously in the context of data size and dataset diversity. From a practical perspective, WAFDect's inference overhead is feasible for offline or near-real-time detection scenarios. This study has not yet systematically evaluated robustness under conditions of feature loss, noise, or adversarial perturbations; these issues will be explored in future work.

5 Conclusion

WAFDect, by introducing a multi-source feature extraction mechanism, collaboratively integrates heterogeneous features from multiple dimensions, including API call sequences, registry operation logs, file operation logs, and network behavior logs. This effectively overcomes the limitation of a single feature source in traditional detection models and achieves comprehensive representation of malware characteristics. Furthermore, the attention weight-based multimodal feature fusion module proposed by WAFDect effectively addresses the challenge of aligning heterogeneous features. By dynamically calculating the interaction weights of different modal features, it realizes deep semantic fusion of multi-source information. Comprehensive analysis of the above experimental results demonstrates that WAFDect exhibits consistent and superior performance on two public datasets as well as the *WAF-data* dataset. In terms of detection and classification tasks, the model surpasses existing mainstream models in overall accuracy, F1 score, and other metrics.

Acknowledgement: Not applicable.

Funding Statement: This research was funded by Name of the National Nature Science Foundation of China, grant number 62262004.

Author Contributions: Xian Wu conceived the study, designed the experiments, took primary responsibility for data collection, preprocessing, and feature engineering, and contributed to manuscript preparation and writing. Jingxia Ren participated in experimental design and manuscript editing. Bangfeng Zhang was responsible for manuscript revision. Liang Wan provided research guidance, reviewed the academic content of the manuscript, and secured funding for the study. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: *Speakeasy* dataset is obtained from https://www.kaggle.com/ds/3231810 (accessed on 25 September 2025). *Avast-CTU_Small* dataset is obtained from https://github.com/avast/avast-ctu-cape-dataset (accessed on 25 December 2025).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Verizon Communications. 2025 data breach investigations report [Internet]. New York, NY, USA: Verizon Communications; 2025 [cited 2025 Sep 1]. Available from: https://www.verizon.com/business/resources/reports/dbir/.
2. Abuadbba A, Lamont S, Ahmed E, Christopher C, Ikram M, Tupakula U, et al. Enhancing malware fingerprinting through analysis of evasive techniques. arXiv:2503.06495. 2025.
3. Islam R, Tian R, Batten LM, Versteeg S. Classification of malware based on integrated static and dynamic features. J Netw Comput Appl. 2013;36(2):646–56. doi:10.1016/j.jnca.2012.10.004.
4. Lindorfer M, Kolbitsch C, Milani Comparetti P. Detecting environment-sensitive malware. In: Sommer R, Balzarotti D, Maier G, editors. Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID). Berlin, Germany: Springer; 2011. p. 338–57. doi:10.1007/978-3-642-23644-0_18.
5. Saxe J, Berlin K. Deep neural network based malware detection using two dimensional binary program features. In: Saxena A, Aghaee P, editors. Proceedings of the 10th International Conference on Malicious and Unwanted Software (MALWARE). Piscataway, NJ, USA: IEEE; 2015. p. 11–20. doi:10.1109/malware.2015.7413680.
6. Chen X, Hao Z, Li L, Cui L, Zhu Y, Ding Z, et al. CruParamer: learning on parameter-augmented API sequences for malware detection. IEEE Trans Inf Forensics Secur. 2022;17:788–803. doi:10.1109/TIFS.2022.3144568.

7. Zhang S, Su H, Liu H, Yang W. MPDroid: a multimodal pre-training Android malware detection method with static and dynamic features. Comput Secur. 2025;150(5):104262. doi:10.1016/j.cose.2024.104262.
8. Chaganti R, Ravi V, Pham TD. A multi-view feature fusion approach for effective malware classification using deep learning. J Inf Secur Appl. 2023;72:103402. doi:10.1016/j.jisa.2023.103402.
9. Wang S, Chen Z, Yan Q, Yang B, Peng L, Jia Z. A mobile malware detection method using behavior features in network traffic. J Netw Comput Appl. 2019;133(3):15–25. doi:10.1016/j.jnca.2019.01.006.
10. Trizna D, Demetrio L, Biggio B, Roli F. Nebula: self-attention for dynamic malware analysis. IEEE Trans Inf Forensics Secur. 2024;19:6155–67. doi:10.1109/TIFS.2024.3445888.
11. Dai X, Yu Z, Liang C, Gao C, He Q, Wu D, et al. Detecting novel malware classes with a foundational multi-modality data analysis model. Data Intell. 2024;6(4):968–93. doi:10.3724/2096-7004.di.2024.0056.
12. Griffin K, Schneider S, Hu X, Chiueh TC. Automatic generation of string signatures for malware detection. In: Kirda E, Jha S, Balzarotti D, editors. Recent Advances in Intrusion Detection: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection (RAID 2009). Berlin, Germany, Berlin Heidelberg: Springer; 2009. p. 101–20. doi:10.1007/978-3-642-04342-0_6.
13. Santos I, Brezo F, Ugarte-Pedrero X, Bringas PG. Opcode sequences as representation of executables for data-mining-based unknown malware detection. Inf Sci. 2013;231:64–82. doi:10.1016/j.ins.2013.01.026.
14. Wang W, Zhu M, Zeng X, Ye X, Sheng Y. Malware traffic classification using convolutional neural network for representation learning. In: Proceedings of the 2017 International Conference on Information Networking (ICOIN); 2017 Jan 11–13; Da Nang, Vietnam. p. 712–7.
15. Kim TG, Kang B, Rho M, Sezer S, Im EG. A multimodal deep learning method for android malware detection using various features. IEEE Trans Inf Forensics Secur. 2018;14(3):773–88. doi:10.1109/TIFS.2018.2861751.
16. Mc Laughlin N. Malceiver: perceiver with hierarchical and multi-modal features for android malware detection. arXiv:2204.05994. 2022.
17. Chen S, Lang B, Liu H, Chen Y, Song Y. Android malware detection method based on graph attention networks and deep fusion of multimodal features. Expert Syst Appl. 2024;237(1):121617. doi:10.1016/j.eswa.2023.121617.
18. Trung DM, Hao TDA, Minh LH, Khoa NH, Cam NT, Pham V-H, et al. DMLDroid: deep multimodal fusion framework for android malware detection with resilience to code obfuscation and adversarial perturbations. arXiv:2509.11187. 2025.
19. Nazim S, Alam MM, Rizvi S, Mustapha JC, Hussain SS, Su'ud MM. Multimodal malware classification using proposed ensemble deep neural network framework. Sci Rep. 2025;15(1):18006. doi:10.1038/s41598-025-96203-3.
20. Arrowsmith J, Susnjak T, Jang-Jaccard J. Multimodal deep learning for android malware classification. Mach Learn Knowl Extr. 2025;7(1):23. doi:10.3390/make7010023.
21. Owoh N, Adejoh J, Hosseinzadeh S, Ashawa M, Osamor J, Qureshi A. Malware detection based on API call sequence analysis: a gated recurrent unit-generative adversarial network model approach. Future Internet. 2024;16(10):369. doi:10.3390/fi16100369.
22. Youssef N, Elbaraway N, Elmaghraby A. Transformer-based API call sequence modeling for dynamic malware detection. In: SoutheastCon 2025. Piscataway, NJ, USA: IEEE; 2025. p. 494–500.
23. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Adv Neural Inf Process Syst. 2017;30:5998–6008. doi:10.65215/ctdc8e75.
24. Hu J, Shen L, Sun G. Squeeze-and-excitation networks. arXiv:1709.01507. 2017.
25. Trizna D. Quo Vadis: hybrid machine learning meta-model based on contextual and behavioral malware representations. arXiv:2208.12248. 2022.
26. Kudo T, Richardson J. SentencePiece: a simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv:1808.06226. 2018.
27. Jindal C, Salls C, Aghakhani H, Long K, Kruegel C, Vigna G. Neurlux: dynamic malware analysis without feature engineering. In: Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC); 2019 Dec 9–13; San Juan, Puerto Rico. p. 444–55.
28. Graves A. Long short-term memory. In: Supervised sequence labelling with recurrent neural networks. Berlin/Heidelberg, Germany: Springer; 2012. p. 37–45. doi:10.1007/978-3-642-24797-2_4.