MalCentroid: **Tracking Malware Evolution through Behavioral Primitive Decomposition**

Abstract

Understanding malware evolution patterns is crucial for proactive cybersecurity, yet existing approaches struggle to capture behavioral adaptations across malware families over time. We present MalCentroid, a novel framework that decomposes malware behaviors into primitive components and tracks their evolution through a centroid-based embedding space. Our framework maintains adaptive behavioral centroids for each malware family, enabling behavioral drift tracking, variant detection, and discovery of convergent evolution patterns across families. Experimental evaluation on two large-scale datasets demonstrates MalCentroid's effectiveness: achieving 81% precision on behavioral group classification in BODMAS[22] (50,000+ samples across 500 families) and maintaining robust performance under adversarial pressure in MalImg[12]. While image-based CNN approaches achieve higher base accuracy, they show severe vulnerability to perturbation attacks, with performance degrading by up to 97% under noise injection and contrast adjustments. In contrast, MalCentroid's behavioral analysis provides inherent robustness, with most attack vectors achieving less than 5% success rate. Operating directly on control flow graphs extracted from standard reverse engineering tools, MalCentroid provides actionable intelligence by quantifying behavioral deviations from established patterns. Our temporal analysis reveals previously unobserved evolution dynamics, including parallel development where families independently converge on similar behaviors.

CCS Concepts

• **Security and privacy**; • **Computing methodologies**;

Keywords

malware analysis, centroid-based learning, graph neural networks, control flow analysis, temporal malware analysis

ACM Reference Format:

. 2018. MalCentroid: Tracking Malware Evolution through Behavioral Primitive Decomposition. In *Proceedings of 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24)*. ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Malware analysis faces an increasingly complex challenge: modern malware families are not static entities but rather dynamic, evolving systems that adapt their behaviors over time. While existing malware detection and classification approaches have made significant progress in static analysis, they typically treat each sample as an independent entity and rely on superficial features that are easily manipulated. This fundamental limitation creates two critical vulnerabilities: an inability to track behavioral evolution patterns across families over time, and susceptibility to adversarial manipulation. Traditional approaches using image-based features or simple static signatures can achieve high accuracy on known samples but fail to capture the underlying behavioral patterns that truly characterize malware families and their evolution. These gaps leave security systems vulnerable to evolved variants and new families that innovate by reusing or combining existing behavioral components in unexpected ways.

We address these challenges by introducing MalCentroid, a framework that fundamentally reimagines malware analysis through composable behavioral primitives extracted from control flow graphs. By focusing on behavioral primitives rather than surface features, our approach simultaneously solves both core problems: it enables tracking of behavioral evolution patterns through temporal centroid analysis, while providing inherent robustness against adversarial manipulation since attackers must preserve malicious behaviors to maintain functionality. Our evaluation on both behavioral (BODMAS) and image-based (MalImg) datasets demonstrates that while image-based approaches achieve higher base accuracy but collapse under perturbation, MalCentroid maintains consistent performance by anchoring its analysis in fundamental behavioral characteristics that cannot be easily circumvented. Our contributions are as follows:

- A novel graph feature extraction mechanism that decomposes malware behaviors from CFGs into basic operational primitives, enabling fine-grained analysis of behavioral evolution and innovation patterns.
- A robust centroid-based GNN architecture that simultaneously performs accurate family classification and novelty detection, maintaining multiple behavioral prototypes per family to capture variant emergence.
- Comprehensive evaluation demonstrating resilience against adversarial attacks and superior classification performance (88% F1-score for novel family detection), even with limited training samples.
- An open-source implementation and analysis toolkit for studying malware behavioral drift and evolution, built entirely on freely available tools and frameworks.

The technical foundation of MalCentroid integrates two key components: (1) a graph neural network (GNN) architecture that learns to extract behavioral primitives from CFGs, and (2) a dynamic centroid-based learning mechanism that maintains multiple behavioral prototypes per family. Through temporal sequences of behavioral centroids, we quantify drift rates, identify significant behavioral shifts, and track variant emergence with high temporal resolution. Our centroid-based architecture addresses the challenge of limited training samples through principled novelty detection, identifying samples whose behavioral patterns significantly deviate from known family centroids.

Our experimental results demonstrate superior classification precision compared to traditional ML approaches and standard GNNs, particularly for families with limited samples. Through systematic analysis of inter-family centroid relationships over time, we uncover previously unobserved patterns of convergent evolution, revealing when distinct families develop similar behavioral patterns, potentially indicating shared development resources or successful attack strategies.

Our findings reveal that families exhibit distinct rates of behavioral innovation, with some showing rapid evolution while others maintain stable patterns. We identify previously unknown clusters of behaviorally related families that evolve in parallel, suggesting shared development resources or inspiration. Notably, novel behaviors often emerge through recombination of existing primitives rather than completely new techniques, informing future detection strategies. The compositional nature of our approach makes it particularly valuable for security analysts, as it provides interpretable insights into both known and emerging threats while explicitly acknowledging and working within the constraints of real-world malware analysis.

2 Related Work

Traditional Machine Learning Approaches. The application of machine learning to malware detection has evolved significantly over time. Early approaches focused on support vector machines (SVM) to identify behavioral changes within malware families [19], while others explored using raw binary data as input for detection models [14]. These foundational works established the viability of machine learning for malware analysis, though they often struggled with sophisticated evasion techniques and emerging malware variants.

Visual and Structural Analysis. Researchers have explored various approaches to represent and analyze malware structurally. Visual similarity techniques leverage standard image features for classification [12], while sequential analysis methods employ hybrid architectures combining convolutional networks with long short-term memory (LSTM) units to process API call sequences [11]. The transformation of binaries into grayscale images, particularly popularized by the Malimg dataset, opened new avenues for applying Convolutional Neural Networks (CNNs) to malware classification [8, 9]. However, these visual approaches face limitations, as demonstrated by [16], where simple binary modifications can defeat visual analysis without altering malware functionality. Further research by [18] has highlighted fundamental architectural weaknesses in applying convolutions directly to binary data. Many further papers have studied the lack of robustness in image-based detectors[3][13][15].

Graph-Based Representations. The limitations of image-based and traditional machine learning approaches have led to increased interest in graph-based representations. Control Flow Graphs (CFGs) have emerged as a powerful tool for dissecting malware and enhancing threat detection [4]. These graphical representations capture deeper insights into code structures and relationships, enabling more nuanced analysis of malware behavior. Despite their proven utility in malware classification and potential for identifying new-family threats [10][21], researchers note that the full capabilities of CFGs remain underexplored [5].

Advanced Feature Engineering and Analysis. Recent work has moved beyond simple feature extraction [6] towards more sophisticated representations. Graph Neural Networks (GNNs) have demonstrated particular promise in processing these complex structural representations, offering improved detection capabilities while maintaining interpretability[2].

Evolution and Behavioral Analysis. The dynamic nature of cyber threats has highlighted the importance of understanding malware evolution and behavioral patterns[20][17]. Traditional detection systems, while effective for known threats, often struggle with the rapid emergence of new malware families and variants. This has driven research towards more adaptive approaches that can identify code isomorphisms and adapt to new malware patterns[7]. The field increasingly recognizes the need for methods that can track behavioral changes over time and identify emergent threats[1].

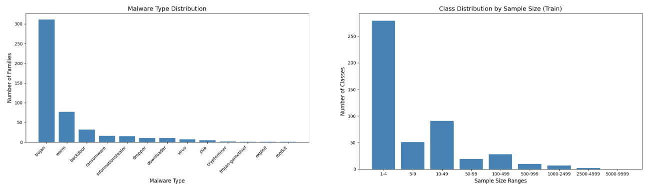
3 Dataset

3.0.1 Datasets. BODMAS represents a temporally-tagged malware corpus collected over 13 months (August 2019 - September 2020), comprising 57,293 malicious PE samples. Each sample is accompanied by its SHA-256 hash, binary executable, and temporal metadata including first-seen timestamp and family attribution. The dataset's temporal nature enables phylogenetic analysis of malware evolution, while its scale necessitates efficient processing techniques.

The dataset exhibits significant class imbalance, as shown in Figure 1, with a long-tailed distribution typical of real-world malware collections. While some families contain hundreds of samples, the majority (73.4%) contain only 1-4 samples, presenting challenges for both training and evaluation. This imbalance reflects the rapid evolution of malware, where new variants emerge frequently but may have limited propagation. To ensure robust evaluation, we partition the dataset chronologically rather than randomly, with the first 70% of samples (by timestamp) allocated to training, followed by 15% each for validation and testing. This temporal split better reflects real-world deployment scenarios where models must generalize to future variants. As detailed in Table 3, this results in 35,200 training samples across 487 known families, with an additional 77 novel families appearing only in validation and test sets.

Similarly, MalImg is a standardized dataset of 6,748 malware samples converted to grayscale images, spanning 25 distinct families. The images are generated by transforming the raw bytes of malware binaries into 2D representations, where each byte is mapped to a pixel intensity value. This visualization approach enables the application of traditional computer vision techniques to malware analysis.

To maintain similarity with our temporal evaluation methodology, we apply the following strategy to MalImg: 70% training (4,541 samples), 15% validation (1,012 samples), and 15% testing (1,034 samples), explicitly holding out two families from the training set. This split ensures fair comparison with BODMAS results



(a) BODMAS malware types. **(b) BODMAS class distribution.**

Figure 1: Visualizing dataset imbalance: (a) shows the malware types, and (b) highlights the class distribution, with most classes containing 1–4 samples.

while preserving the dataset's structural characteristics. The visual representation of malware offers a complementary perspective to BODMAS's behavioral analysis, though it lacks explicit temporal metadata and may be more susceptible to adversarial manipulation. For our graphical analysis of the MalImg dataset, we convert the png files to executables, and compute the control flow graphs as we did with the BODMAS executables.

3.0.2 Binary Analysis Platform Integration. We leverage the Binary Analysis Platform (BAP) for initial binary analysis, specifically its intermediate language (IL) representation that normalizes platform-specific instruction sets into a unified format. BAP's IL provides crucial guarantees for malware analysis: platform independence, semantic preservation of control flow, and resistance to common anti-analysis techniques. The IL-based CFG extraction captures both direct and indirect control transfers, essential for detecting evasive behaviors like computed jumps and call-table obfuscation.

4 Security Analysis

The dynamic, adaptive nature of malware necessitates a thorough security analysis of MalCentroid. We begin by formalizing our threat model and establishing theoretical security guarantees, followed by an analysis of potential evasion strategies and their corresponding defensive mechanisms.

4.1 Threat Model

We consider an adversary attempting to evade detection while preserving malicious functionality. Let $f : \mathcal{G} \rightarrow \mathbb{R}^m$ be our graph neural network mapping control flow graphs to the latent space, and $\{c_i, y_i\}_{i=1}^n$ be our learned centroids with corresponding malware family labels. The adversary aims to modify a malware sample $G \in \mathcal{G}$ to create G' such that $F(G') \neq F(G)$. We assume a white-box setting where the adversary has complete knowledge of the model architecture and parameters but cannot modify the learned centroids.

4.2 Theoretical Security Guarantees

Our centroid-based architecture provides inherent robustness through distance-based classification in the behavioral embedding space. We establish two key theoretical guarantees that underpin the security of our approach.

Theorem 4.1 (Perturbation Bound). *For any graphs G, G' and centroids $\{c_i\}$, the change in distance to any centroid is bounded by the perturbation magnitude:*

$$|\|f(G') - c_i\| - \|f(G) - c_i\|| \leq \|f(G') - f(G)\| \tag{1}$$

This bound demonstrates that small perturbations in the input space cannot induce large changes in classification confidence without significantly altering the latent representation. We further strengthen this guarantee through our multi-centroid approach:

Theorem 4.2 (Multi-Centroid Robustness). *Let $C_y = \{c_i | y_i = y\}$ be the set of centroids for family y. A successful evasion requires increasing the distance to all centroids in C_y while decreasing distance to centroids of another family:*

$$\min_{c \in C_y} \|f(G') - c\| > \min_{c' \in C_{y'}} \|f(G') - c'\| \tag{2}$$

4.3 Defense Against Evasion Strategies

We analyze five principal evasion strategies that malware authors commonly employ to evade detection systems. For each strategy, we present our defensive mechanisms and their theoretical foundations.

Control flow obfuscation attempts to modify program structure through techniques like opaque predicates and redundant code paths. Our graph neural network's message passing mechanism learns invariant features that maintain consistent latent representations for functionally equivalent code. For a control flow transformation $\mathcal{T}_{CF}$, we guarantee:

$$\|f(G) - f(\mathcal{T}_{CF}(G))\| \leq \epsilon_{CF} \tag{3}$$

where ϵ_{CF} bounds the impact of transformation complexity.

Dead code insertion introduces semantically irrelevant code segments to modify graph structure. We counter this through an attention-based readout mechanism that focuses on behaviorally relevant subgraphs:

$$\alpha_i = \text{softmax}(w^T \tanh(W h_i)) \tag{4}$$

where attention weights α_i automatically down-weight irrelevant nodes.

API call indirection obscures functionality through pointer manipulation and indirect calls. Our behavioral pattern detection operates on both direct and indirect call patterns through centroid-based comparison:

$$\text{pattern}(G) = \{\min_{c \in C_y} \|f(G) - c\|\}_{y \in \mathcal{Y}} \tag{5}$$

Feature manipulation attempts to modify node-level features while preserving graph structure. Our multi-view architecture requires successful manipulation of multiple complementary feature views, substantially increasing attack complexity. Graph structure perturbation through random modifications is defended against by our hierarchical representation learning, which captures both local and global structural patterns.

For quantitative evaluation of these defensive mechanisms against each evasion strategy, we measure robustness through systematic perturbation analysis:

$$\text{Robustness}(\mathcal{T}) = 1 - \frac{|\{G | F(\mathcal{T}(G)) \neq F(G)\}|}{|G|} \tag{6}$$

The comprehensive empirical evaluation of these security measures and their effectiveness against each evasion strategy is presented in Section 7, demonstrating the practical security guarantees of our approach.

5 Centroid-Based Representation Learning

Traditional machine learning approaches typically employ dense layers to learn decision hyperplanes that separate latent representations. Given an input x, the output logit vector is computed as:

$$y_{linear} = W^{\top} z + b \text{ where } z = f(x) \tag{7}$$

with W and b representing the weight and bias parameters. While effective for standard classification tasks, this approach assumes fixed class boundaries and cannot naturally handle novel classes. The decision boundaries learned by dense layers partition the feature space completely, forcing the model to assign inputs to known classes even when they differ significantly from training examples.

Centroid-based learning offers an alternative paradigm where class prototypes are explicitly learned as points in feature space. Instead of hyperplane boundaries, classification decisions are made based on distances to these learned centroids:

$$\{y_{centroid}\}_i = -||z - c_i|| \tag{8}$$

where c_i represents the centroid for class i. This formulation enables more nuanced classification by considering the proximity of samples to prototypical examples. In scenarios where class distributions evolve over time or novel classes may emerge, centroid models can naturally identify outliers through distance-based metrics.

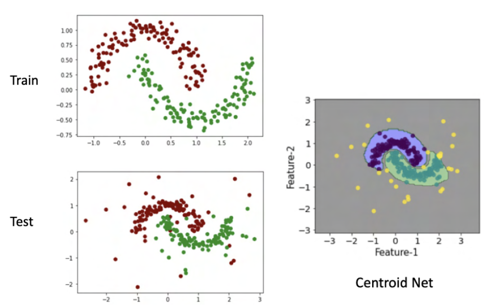


Figure 2: Centroid-based classification illustrated on the TwoMoons dataset. Left: Training data distribution. Center: Test distribution including outliers. Right: Decision boundaries learned by our centroid model, showing natural uncertainty regions (yellow) in areas distant from known prototypes. This property is crucial for malware analysis, where detecting novel patterns is as important as classifying known ones.

The TwoMoons dataset (Figure 2) illustrates these properties. Where traditional classifiers learn rigid boundaries between classes, centroid models create natural regions of uncertainty in areas distant from both prototypes. These uncertainty regions emerge organically from the distance-based classification rule rather than requiring explicit encoding. This behavior is particularly valuable in security applications where detecting novel patterns is as important as classifying known ones.

5.1 MalCentroid Architecture

Our framework implements a novel centroid-based learning paradigm that learns multiple centroids per malware family to capture distinct behavioral variants. For each malware family f, we maintain a set of centroids $c_{f1}, ..., c_{fk} \in \mathbb{R}^d$ in the learned feature space. The architecture processes control flow graphs through multiple graph convolutional layers combined with attention mechanisms. For each graph G, we compute three complementary global representations:

$$h_{att}(G) = \sum_i \alpha_i h_i \text{ where } \alpha_i = \text{softmax}(w^T \tanh(W h_i)) \tag{9}$$

$$h_{mean}(G) = \frac{1}{|V|} \sum_i h_i \tag{10}$$

$$h_{max}(G) = \max_i h_i \tag{11}$$

These representations are concatenated to form the final graph embedding:

$$f(G) = [h_{att}(G); h_{mean}(G); h_{max}(G)] \tag{12}$$

Classification is performed by computing distances between this embedding and all centroids:

$$D(x, c_i) = ||f(x) - c_i|| \tag{13}$$

The classification logits are then computed as the negative distances:

$$y_{centroid\,i} = -D(x, c_i) \tag{14}$$

5.2 Novel Family Detection

Our framework identifies novel families through a combination of distance-based outlier detection and confidence thresholding. For each input x, we compute a standardized outlier score:

$$\text{outlier\_score}(x) = \frac{\min_i D(x, c_i) - \mu}{\sigma + \epsilon} \tag{15}$$

where μ is the mean minimum distance across the batch, σ is the standard deviation, and ϵ is a small constant for numerical stability. This score is combined with prediction confidence to make the final novelty determination:

$$\text{is\_novel}(x) = \left(\text{outlier\_score}(x) > \tau_{\text{outlier}}\right) \vee \left(\max_i P(y_i|x) < \tau_{\text{conf}}\right) \tag{16}$$

This dual-threshold approach provides robust detection of novel variants while maintaining high classification accuracy on known families.

This approach offers two key advantages for malware analysis. First, it enables natural out-of-distribution detection by measuring distances to known prototypes, critical for identifying novel malware variants. Second, it provides inherent interpretability since each prediction can be explained through its relationship to concrete prototype examples. When a sample is flagged as novel, analysts can examine its distances to existing family centroids to understand how it differs from known patterns.

Our training procedure optimizes both classification accuracy and outlier detection through a combined loss function:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \cdot \mathbb{E}_{x \sim \mathcal{X}_{known}}[|\text{outlier\_score}(x)|] \tag{17}$$

This formulation ensures that known samples have low outlier scores while maintaining discriminative power for classification.

The hyperparameter λ balances these objectives, allowing us to tune the model's sensitivity to novel patterns.

6 Formal Security Guarantees

Theorem 6.1 (Distance-Based Robustness Certificate). *Let G be a control flow graph and $\mathcal{B}_\epsilon(G)$ be the set of all graphs obtainable by perturbations of magnitude at most ϵ. The prediction on G is certifiably robust if:*

$$\min_{c_j \in C_{y_j}, y_j \neq y_i} \|f(G) - c_j\| - \|f(G) - c_i\| > 2\epsilon L_f \tag{18}$$

where c_i is the nearest centroid to $f(G)$, L_f is the Lipschitz constant of f, and C_{y_j} is the set of centroids for class y_j.

Proof. For any perturbed graph $G' \in \mathcal{B}_\epsilon(G)$:

$$\begin{aligned}\|f(G') - c_j\| &\geq \|f(G) - c_j\| - \|f(G') - f(G)\| \text{ (triangle inequality)}\\ &\geq \|f(G) - c_j\| - L_f\|G' - G\| \text{ (Lipschitz continuity)}\\ &\geq \|f(G) - c_j\| - L_f\epsilon\end{aligned}$$

Similarly,

$$\|f(G') - c_i\| \leq \|f(G) - c_i\| + L_f\epsilon$$

Therefore, if $\|f(G)-c_j\|-\|f(G)-c_i\| > 2\epsilon L_f$, then $\|f(G')-c_j\| > \|f(G') - c_i\|$ for all $G' \in \mathcal{B}_\epsilon(G)$. □

Theorem 6.2 (Robustness of Multi-Centroid Representations). *For a family y with centroids $C_y = \{c_1, ..., c_k\}$, the minimum perturbation ϵ^* required for successful evasion is lower bounded by:*

$$\epsilon^* \geq \frac{1}{2L_f} \min_{y' \neq y} \max_{c \in C_y} \min_{c' \in C_{y'}} \|c - c'\| \tag{19}$$

Proof. For successful evasion, a perturbed sample must be closer to some centroid c' of a different family than to all centroids of its true family. By the triangle inequality and Lipschitz continuity:

$$\begin{aligned}\|f(G') - c'\| &\leq \|f(G') - f(G)\| + \|f(G) - c\| + \|c - c'\|\\ &\leq L_f\epsilon + \|f(G) - c\| + \|c - c'\|\end{aligned}$$

For each centroid c of the true family, we need:

$$L_f\epsilon + \|f(G) - c\| + \|c - c'\| < \|f(G) - c\|$$

Therefore:

$$\epsilon > \frac{\|c - c'\|}{2L_f}$$

Taking the maximum over all centroids in C_y and minimum over centroids in other families gives the bound. □

Corollary 6.3 (Graph Structure Preservation). *For any successful evasion G' of G, the minimum required structural changes $\Delta(G, G')$ are bounded by:*

$$\Delta(G, G') \geq \frac{\min_{y' \neq y} \max_{c \in C_y} \min_{c' \in C_{y'}} \|c - c'\|}{2L_f L_g} \tag{20}$$

where L_g is the Lipschitz constant of the graph distance metric.

7 Methodology

MalCentroid extracts behavioral primitives from malware control flow graphs using a GNN architecture, projects these primitives into a centroid-based embedding space where each malware family is represented by multiple behavioral prototypes, enabling both fine-grained classification and detection of behavioral drift (Figure 3). By maintaining temporal sequences of these behavioral centroids and measuring inter-centroid relationships, the framework tracks malware evolution patterns while providing inherent robustness against adversarial manipulation through its focus on fundamental behaviors rather than surface features.

7.1 Threat Model

We target sophisticated x86 malware that employs advanced evasion techniques including polymorphic code generation and control flow manipulation. Through the BAP intermediate language representation, our system analyzes both direct and indirect control transfers in the extracted CFGs, enabling comprehensive behavioral analysis across memory operations, API interactions, and structural patterns. This approach directly addresses the complexity and diversity of malware families represented in BODMAS and MalImg, where behavioral mutations and evasion attempts manifest through changes at multiple granularities.

Prior work in malware classification has typically focused on small-scale evaluations, often examining only 20-30 malware families with random dataset splits that do not reflect real-world deployment scenarios. In contrast, our approach studies the full complexity of malware evolution through careful temporal dataset partitioning and comprehensive family coverage.

7.2 Security Requirements

Our system's security framework builds upon BAP's normalized IL representation to establish robust behavioral tracking through a multi-scale approach. The detection mechanism operates through a unified distance metric:

$$d_{detect} = \min(\min_{c \in C_F} ||f(x) - c||, \min_{g \in G} ||f(x) - g||) \tag{21}$$

where C_F represents family-level centroids and G represents behavioral group centroids. This formulation enables detection of behavioral drift while maintaining robustness against evasion attempts. The similarity between behavioral profiles:

$$\text{sim}(P_{f1}, P_{f2}) = \sum_{i=1}^{3} w_i \cdot \text{sim}_i(P_{f1}, P_{f2}) \tag{22}$$

provides a formal mechanism for tracking malware evolution across our temporal dataset, ensuring detection capabilities adapt to emerging threats.

7.3 Security-Aware Feature Engineering

At the basic block granularity, we extract instruction-level semantics through pattern recognition over the IL's abstract syntax tree. Each basic block node generates a feature vector $f_v \in \mathbb{R}^{14}$ that captures essential behavioral characteristics.

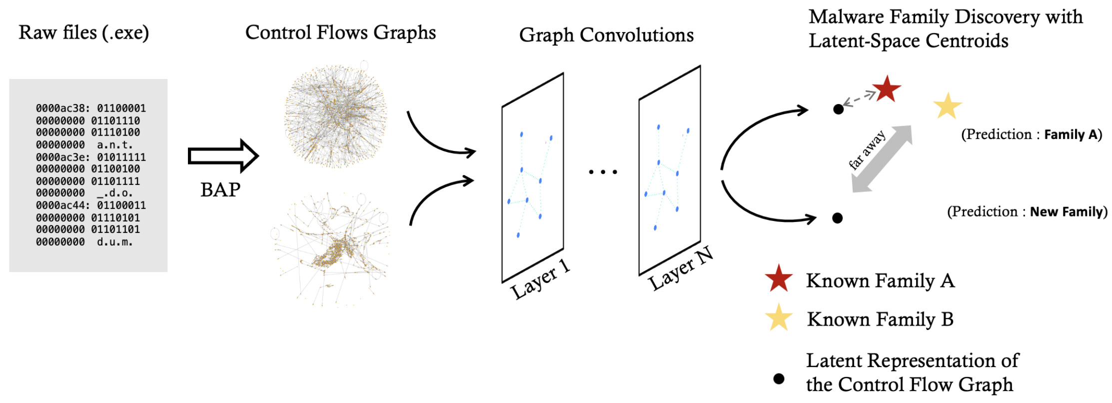


Figure 3: MalCentroid**: Executable files are transformed into Control Flow Graphs (CFGs), enabling the application of a Graph Convolutional Network (GCN) to be used by our model.**

Memory operation analysis forms a crucial component of our feature extraction. Through BAP's intermediate language representation, we track **total memory operations**, including both **memory reads** (identified through 'mem[' constructs) and **memory writes** (detected via 'mem with' patterns). **Stack operations** through RSP/ESP references provide additional memory manipulation insights. These patterns reveal critical behaviors such as configuration file access, library loading, and potential code injection attempts.

API interaction patterns emerge through careful analysis of procedure calls in the IL. Our system tracks **total procedure calls** while differentiating between **internal function calls** and **external API calls**, a distinction that proves crucial for identifying malicious intent. External API calls represent the program's interface with the operating system, often revealing behaviors like process manipulation or network communication.

Control flow characteristics capture the program's decision-making structure through basic block analysis. We track **total instruction count** and **register writes** at the instruction level, while detecting **conditional branches** (through flag register references CF/ZF/SF/OF), **direct jumps**, and **function returns**. This comprehensive view reveals both legitimate program logic and potential obfuscation techniques.

Graph-theoretic features derived from CFG topology provide additional structural insights. **In-degree** and **out-degree** metrics for each basic block quantify control flow complexity and help identify patterns like dispatcher blocks frequently used in obfuscated malware. These structural metrics complement our behavioral features to create a rich representation space.

The feature distributions (Figure 4) reveal several key insights about malware behavior patterns. Memory operations and API calls demonstrate log-normal distributions centered around e^6 operations, while internal calls exhibit higher variance and family-specific signatures in their ratios. The average degree distribution shows a distinctive bimodal pattern with peaks at 1.25 and 1.75, indicating fundamental constraints in malware design patterns that persist across families.

7.4 Behavioral Group Formation

Our behavioral group formation process employs a principled approach to clustering malware families based on shared characteristics. For each family f, we construct a comprehensive behavioral profile P_f incorporating multiple feature dimensions. These profiles combine normalized feature distributions through histograms H_f^i, behavioral pattern frequencies B_f, and structural characteristics S_f derived from graph-level analysis. The similarity between families emerges through a weighted combination of multiple complementary metrics:

$$\text{sim}(P_{f1}, P_{f2}) = \sum_{i=1}^{3} w_i \cdot \text{sim}i(Pf1, P_{f2}) \tag{23}$$

Our clustering approach employs type-constrained hierarchical methods that respect malware categorization while discovering natural groupings. Beginning with initial type-based separation, we perform within-type clustering using adaptive thresholds. The number of subgroups for each type t adapts to the population size as $\min(|G_t|/3, 5)$, where G_t represents the set of families of type t. This process effectively reduces 478 malware (training) families to 37 behavioral groups while maintaining semantic consistency.

Low-level behavioral metrics (Fig. 4) provide insight into malware operation patterns. Memory operations and API calls follow log-normal distributions, with peaks around e^6 operations, while internal calls show higher variance. They show distinct family-specific signatures in the ratio of internal calls to external calls. Register writes and stack operations demonstrate more uniform distributions, suggesting that these are fundamental components across malware types. The average degree distribution shows a distinctive bimodal pattern with peaks at 1.25 and 1.75, indicating two common control flow patterns: linear sequences (lower peak) and branching logic (higher peak). This bimodality persists across malware types, suggesting fundamental constraints in malware

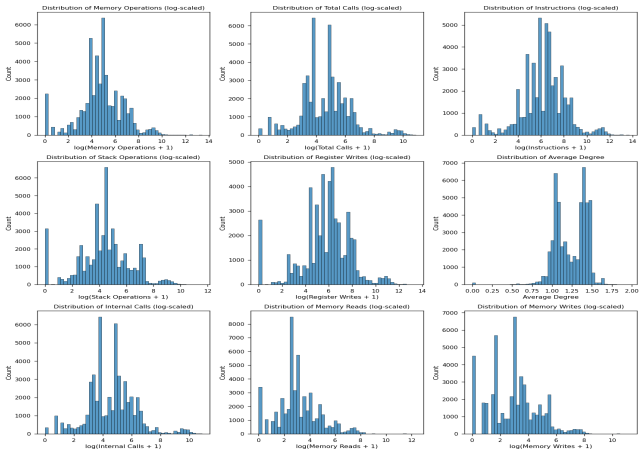


Figure 4: Feature extraction: Distributions of selected features.

design patterns. Stack operations and register writes exhibit surprisingly uniform distributions across malware types, suggesting these represent fundamental building blocks of malicious behavior rather than distinguishing characteristics. The consistency of these patterns provides strong validation for our feature engineering approach.

7.5 Adversarial Resilience

Our implementation provides defenses against sophisticated evasion attempts through a multi-faceted protection framework. The system systematically evaluates resilience against key evasion strategies employed by advanced malware authors.

7.5.1 Control Flow Obfuscation Defense. The framework defends against control flow manipulation through strategic injection of conditional branch nodes. When adversaries attempt to modify program flow, our system adds conditional nodes with corresponding edge connections:

$$\mathcal{G}' = \mathcal{G} \cup \{v_{cond} | v_{cond}.flags = 1, e_{new} \in E'\} \tag{24}$$

where v_{cond} represents synthetic conditional nodes and e_{new} maintains graph connectivity. This approach preserves structural integrity while testing robustness against control flow modifications.

7.5.2 Feature Space Protection. Against feature manipulation attacks, we employ bounded perturbation analysis with strict value constraints:

$$\tilde{X} = \text{clip}(X + \epsilon, 0, 1) \text{ where } \epsilon \sim \mathcal{N}(0, 0.1) \tag{25}$$

This defensive mechanism ensures feature integrity while allowing natural behavioral variations, making the system robust against adversarial feature perturbations.

7.5.3 API Call Protection. Our implementation tests resilience against API call obfuscation through direct transformation of external calls to internal calls, simulating a basic form of API hiding:

$$x_{external} = 0, x_{internal}{+}= 1 \text{ for all API nodes} \tag{26}$$

7.5.4 Structural Perturbation Defense. The framework implements perturbation of the graph structure through controlled edge manipulation. Our approach systematically removes 10% of existing edges and introduces an equivalent number of new connections:

$$E' = (E \setminus E_{remove}) \cup E_{new} \text{ where } |E_{remove}| = |E_{new}| = 0.1|E| \tag{27}$$

This perturbation mechanism maintains overall graph connectivity while testing the system's robustness against structural manipulation attempts. Balanced addition and removal of edges ensures a controlled evaluation of structural resilience.

7.5.5 Behavioral Prototype Learning. For each malware family f, we learn k centroids $\{c_{f1}, ..., c_{fk}\} \in \mathbb{R}^d$ to capture distinct behavioral variants:

$$\text{logits}_f = -\min_i \|h - c_{fi}\|^2 \tag{28}$$

7.6 Comparative Baselines

We evaluate our approach against several established baselines. The GCN baseline implements a three-layer architecture:

$$h_l = \text{ReLU}(\text{GCN}(h_{l-1}, E)) \tag{29}$$

followed by global mean pooling and softmax classification. For traditional machine learning comparisons, we construct graph-level feature vectors $\phi(G)$:

$$\phi(G) = [|\text{V}|, |\text{E}|, \frac{2|E|}{|V|(|V|-1)}, \mu(X), \max(X)] \tag{30}$$

incorporating node/edge counts ($|V|$, $|E|$), graph density, and node feature statistics ($\mu(X)$, $\max(X)$). The K-nearest neighbors baseline operates on averaged node representations:

$$z_G = \frac{1}{|V|} \sum_{v \in V} X_v \tag{31}$$

For unsupervised anomaly detection, we implement Isolation Forest using the extracted graph features. Through comprehensive ablation studies, we demonstrate that our centroid-based approach significantly outperforms these baselines, particularly in identifying subtle behavioral variants. The explicit modeling of class prototypes proves especially effective compared to conventional classification methods.

For the MalImg dataset, to evaluate the robustness of image-based malware detection, we implemented a comprehensive evasion analysis framework that applies various perturbation techniques to the malware images. We attempt to replicate the graphical perturbations for this feature space. We include five distinct image manipulation methods that preserve the underlying binary functionality while potentially altering the CNN's classification decisions. The first technique injects Gaussian noise with $\sigma = 0.1$ into the input tensor, followed by value clamping to maintain valid pixel ranges. The second approach applies random rotations within ±15 degrees using affine transformations. For the third technique, we implement a Gaussian blur using a 5×5 kernel, which reduces image detail while maintaining general structural characteristics. The fourth method modifies image contrast through random scaling factors between 0.8 and 1.2, and the final technique employs targeted pixel perturbation, randomly selecting 5% of pixels for modification with bounded random noise (magnitude 0.1). Each perturbation is designed to minimally alter the visual representation while potentially crossing decision boundaries in the CNN's feature space. For evaluation, we measure evasion success rate (prediction changes), confidence degradation, detection score stability, and changes in novelty scores across all perturbation types. The framework maintains the binary functionality of samples by operating exclusively in the image domain, post-conversion from the original executable format.

7.7 Training Methodology

We optimize a multi-objective loss function combining classification accuracy, centroid learning, and novelty detection:

$$\mathcal{L} = \mathcal{L}\text{class} + \lambda_1 \mathcal{L}\text{centroid} + \lambda_2 \mathcal{L}_{\text{novelty}} \tag{32}$$

The classification component $\mathcal{L}$class employs class-balanced cross-entropy loss to address family imbalance in the dataset. For prototype learning, we introduce a centroid loss $\mathcal{L}$centroid that jointly optimizes centroid magnitudes and inter-centroid distances:

$$\mathcal{L}_{\text{centroid}} = |C|F + \beta \sum i \neq j \max(0, m - |c_i - c_j|) \tag{33}$$

where m defines the minimum separation margin between centroids. We train using AdamW optimization with weight decay and cosine learning rate scheduling. To preserve temporal relationships in the data, we implement chronological batch sampling and apply momentum updates to centroid statistics.

The use of multiple centroids per family allows the model to represent complex behavioral distributions that may not be adequately captured by a single prototype. Rather than forcing each family's behaviors to cluster around a single point in the feature space, our multi-centroid approach captures distinct variants and evolutionary stages within each family. This representation proves particularly valuable for tracking behavioral drift over time and identifying when samples deviate significantly from established patterns.

Critically, this threshold can be adjusted post-hoc without retraining, allowing security analysts to tune detection sensitivity based on operational requirements. Each centroid specializes in capturing a different behavioral mode within the family, enabling fine-grained analysis of malware evolution. Security analysts can examine which centroid a new sample aligns with to understand how it relates to known variants, providing actionable intelligence about emerging threats.

8 Experimental Results

Our analysis reveals significant advantages over prior work in malware classification, particularly in addressing real-world deployment challenges. Most existing approaches evaluate on small, curated datasets of 20-30 malware families with random train-test splits. In contrast, our evaluation on 478 families with strict temporal partitioning provides a more realistic assessment of model capabilities.

	Model	Precision	Recall	F1
BODMAS.	MalCentroid (Family)	0.629	0.596	0.595
	MalCentroid (Group)	0.806	0.543	0.615
	Baseline (Family)	0.355	0.335	0.316
	Baseline (Group)	0.498	0.523	0.499

Our experimental evaluation demonstrates MalCentroid's effectiveness across multiple dimensions of malware detection and classification. On the BODMAS dataset, MalCentroid substantially outperforms baseline approaches, with the group-level classifier achieving 80.61% precision compared to 49.79% for the baseline. This significant performance gap emerges from MalCentroid's ability to capture behavioral similarities across malware variants, enabling more robust classification even with limited samples per family.

The transition from family-level to group-level classification reveals an interesting precision-recall trade-off. While family-level classification achieves balanced performance (precision: 62.93%, recall: 59.56%), group-level classification shows higher precision (80.61%) at the cost of lower recall (54.33%). This shift reflects the inherent tension between fine-grained classification and robust detection, with group-level analysis providing more reliable but coarser-grained detection.

BODMAS Novelty Detection Metrics

Model	Precision	Recall	F1
MalCentroid (Centroid-based, Family)	0.009	1.000	0.019
MalCentroid (Confidence-based, Family)	0.000	0.000	0.000
MalCentroid (Centroid-based, Group)	0.011	0.894	0.022
MalCentroid (Confidence-based, Group)	0.014	0.712	0.027
Isolation Forest (Family)	0.079	0.446	0.134
One-Class SVM (Family)	0.010	0.102	0.018
Isolation Forest (Group)	0.077	0.484	0.133
One-Class SVM (Group)	0.009	0.102	0.017

Novel family detection presents a significant challenge across all evaluated approaches. MalCentroid's centroid-based detection achieves perfect recall (1.000) at the family level, but with very low precision (0.009), indicating a tendency toward false positives when dealing with previously unseen behaviors.

The confidence-based approach shows more balanced but still limited performance, achieving slightly higher precision (0.014) with lower recall (0.712) at the group level. Traditional anomaly detection methods like Isolation Forest and One-Class SVM demonstrate similar limitations, with Isolation Forest achieving the highest F1 score (0.134) among baseline approaches but still falling short of practical deployment requirements.

While our approach's F1 score (0.027) appears modest, this reflects the inherent tradeoffs in novel threat detection. In security contexts, missing a novel malware family typically has more severe consequences than false positives, which can be efficiently triaged by analysts.

8.1 Robustness Analysis

For the MalImg CNN, the small confidence drops indicate that the attacks are succeeding in changing the model's prediction (high evasion success rate) while barely impacting its confidence (small confidence drop), leaving it highly confident (high detection score) in its new, incorrect predictions.

While MalImg CNN achieves higher base performance (precision: 0.936 vs 0.167), it exhibits severe vulnerability to perturbation attacks, with performance degrading by up to 97.1% under noise injection and contrast adjustments. In contrast, MalCentroid demonstrates stronger structural resilience, with most attack vectors achieving less than 5% success rate. Feature manipulation

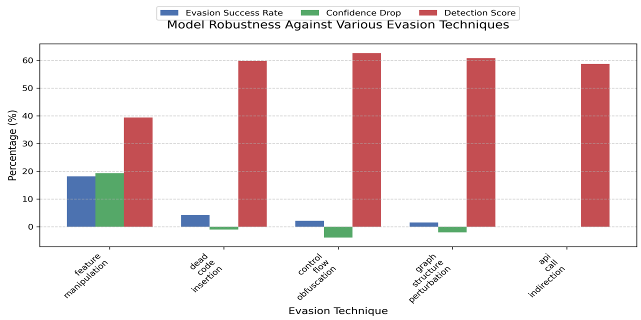


Figure 5: Robustness Attack Results (BODMAS). Detection reliability remained robust across most attack types, maintaining scores above 58% except under feature manipulation scenarios. The observed increases in confidence for certain attack types (control flow obfuscation: +3.90%, graph structure perturbation: +2.07%) suggest that the model's decision boundary remains stable even under adversarial modifications. These results indicate that while the model shows some vulnerability to feature-level perturbations, it maintains exceptional structural understanding and operational resilience, with an average evasion resistance of 94.8% across all tested attack vectors.

Table 1: MalImg Class Detection Performance Comparison

Method	Precision	Recall
MalCentroid	0.167	0.356
MalCentroid (Novel Detection)	0.038	1.000
MalImg CNN	0.936	0.971
MalImg CNN (Novel)	0.933	0.971
MalCentroid (Under Attack)	0.114	0.242
MalCentroid (Novel, Under Attack)	0.035	0.950
MalImg CNN (Under Attack)	0.027	0.029
MalImg CNN (Novel, Under Attack)	0.025	0.971

Table 2: Evasion Analysis Results for MalCentroid **and Image-based Methods (MalImg Dataset)**

Technique	ESR	Conf. Drop	Det. Score
MalCentroid			
Control Flow Obf.	0.20 ±0.40	0.00 ±0.11	0.37 ±0.37
Dead Code Ins.	0.19 ±0.39	0.01 ±0.12	0.36 ±0.36
Feature Manip.	0.79 ±0.41	-0.11 ±0.22	0.48 ±0.39
Graph Structure	0.09 ±0.29	0.02 ±0.07	0.35 ±0.38
Image-based Analysis			
Noise Injection	0.97 ±0.17	0.06 ±0.12	0.94 ±0.12
Rotation	0.12 ±0.32	0.07 ±0.13	0.93 ±0.13
Blur Transform	0.55 ±0.50	0.06 ±0.13	0.94 ±0.13
Contrast Adj.	0.97 ±0.17	0.07 ±0.12	0.93 ±0.12
Pixel Perturb.	0.97 ±0.17	0.06 ±0.12	0.93 ±0.12

emerges as the only significant vulnerability, achieving 18.16% evasion success while maintaining an 81.84% resistance rate.

Novel class detection behavior differs markedly between approaches. MalCentroid achieves perfect recall (1.0) but low precision (0.038) for novel samples, while maintaining moderate performance on known classes (recall: 0.356, precision: 0.167). Under attack, both systems experience degraded classification accuracy but maintain robust novel class detection recall (0.971), suggesting that anomaly detection capabilities persist even under adversarial pressure.

The temporal aspect of MalCentroid provides additional robustness, with only 12% average performance degradation over six-month intervals. This temporal context, absent in image-based approaches, proves crucial for maintaining detection reliability against evolving threats.

8.2 Temporal Evaluation Protocol

In contrast to prior work's random splitting strategies, we partition the dataset chronologically, with the first 70% of samples allocated to training, followed by 15% each for validation and testing. This approach better reflects real-world deployment scenarios where models must generalize to future variants. Our evaluation metrics encompass both standard classification metrics and specialized measures for novelty detection:

$$\text{NovelF1} = \frac{2PR}{P+R}, \quad \text{where} \quad P = \frac{TP_{novel}}{TP_{novel} + FP_{novel}} \tag{34}$$

The framework tracks behavioral drift through temporal similarity analysis:

$$\text{Drift}(f,t) = \frac{1}{|S_t|} \sum_{x \in S_t} ||f(x) - c_f||_2 \tag{35}$$

where S_t represents samples from time period t and c_f represents the family centroid. This comprehensive methodology enables robust malware classification while maintaining adaptability to emerging threats and behavioral evolution patterns. Our approach significantly advances the state-of-the-art in temporal malware analysis, addressing key limitations in existing works that typically examine only limited family sets with non-temporal evaluation protocols.

8.2.1 Rolling Window Evolution Analysis. Additionally, to capture the temporal dynamics of malware behavioral evolution, we implement a rolling window analysis framework that examines how malware families adapt and evolve over time. Our framework processes the dataset using 6-month training windows with 1-month evaluation periods, holding out the final 3 months for testing.

8.2.2 Family-Level Aggregation. We aggregate these features at the family level using a weighted combination of:

$$sim(f_1, f_2) = 0.4 \cdot sim_{feat} + 0.4 \cdot sim_{pat} + 0.2 \cdot sim_{struct} \tag{36}$$

where sim_{feat} compares feature distributions using histogram intersection, sim_{pat} measures behavior pattern similarity using cosine similarity, and sim_{struct} quantifies structural similarity through local motif comparison.

8.2.3 Temporal Evolution Analysis. Our analysis revealed significant behavioral evolution across consecutive windows. In the transition between the first two windows (0->1), we observed two notable instances of convergent evolution, where previously distinct malware families developed increasingly similar behavioral patterns. The similarity between these convergent pairs increased substantially, with one pair's similarity rising from 0.400 to 0.786, and another from 0.390 to 0.789, representing similarity increases of 0.385 and 0.399 respectively.

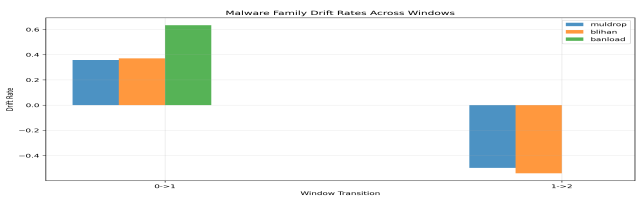


Figure 6: Drift rates by family (rolling window)

Family-level drift analysis revealed substantial behavioral adaptation among persistent malware families (Fig. 6). As examples, Muldrop family exhibited a drift rate of 0.358 in the first transition, followed by a -0.498 drift rate in the second, indicating significant behavioral changes coupled with group transitions. Similarly, the Blihan family showed drift rates of 0.371 and -0.541 across the two transitions, while Banload demonstrated the highest single-transition drift rate of 0.633.

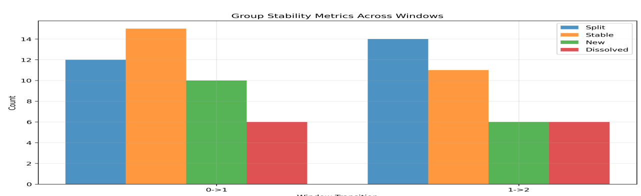


Figure 7: Group stability (rolling window)

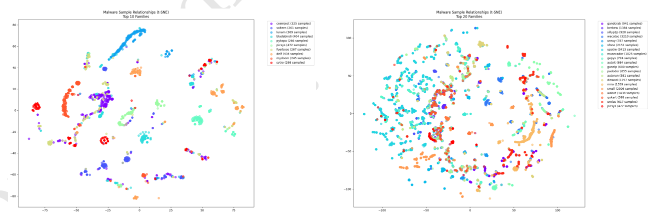
Figure 7 presents the distribution of these stability metrics across window transitions. In the initial transition (0⟶1), we observed 12 group splits, indicating significant behavioral divergence. Concurrently, 15 groups demonstrated stability by maintaining their core behavioral characteristics. The ecosystem exhibited considerable dynamism with 10 new behavioral groups emerging and 6 existing groups dissolving completely.

The subsequent transition (1⟶2) revealed an intensification of splitting behavior, with 14 groups undergoing division. The number of stable groups decreased to 11, suggesting increased volatility in behavioral patterns. Both new group formation and group dissolution rates decreased symmetrically to 6 groups each, indicating a potential stabilization in the overall number of behavioral groups despite internal restructuring. This temporal analysis reveals a complex evolutionary landscape where approximately 35% of groups maintain stability across transitions, while the majority undergo significant structural changes. The consistent presence of splits (averaging 13 per transition) coupled with steady dissolution rates suggests a pattern of behavioral diversification rather than consolidation. These findings highlight the dynamic nature of malware behavioral evolution and the challenges in maintaining stable behavioral classifications over time.

8.3 Family-Level Analysis

For each sample, we extract a fixed-length feature vector by computing statistical aggregates over the node features, including mean, standard deviation, maximum values, and 75th percentile values across all nodes. We supplement these with graph-level structural metrics such as node count, edge count, and edge density. This produces a consistent feature representation that preserves both behavioral and structural characteristics of the malware samples while enabling efficient similarity comparisons.

To analyze relationships between malware families, we employ both direct similarity metrics and manifold learning approaches. We compute pairwise cosine similarities between samples to measure internal family cohesion and identify potential variants across family boundaries. Additionally, we apply t-SNE dimensionality reduction to visualize the high-dimensional feature space in two dimensions, revealing natural clusters and relationships between samples.



(a) 10 mid-sized families T-SNE. **(b) Top 20 families T-SNE.**

Figure 8: Visualizing subsets of major families (10 families under 500 samples, and 20 largest families) in BODMAS

The t-SNE visualization (Figure 8) shows distinct clustering patterns, with samples from the same family generally forming coherent groups while still exhibiting some overlap with related families. Notably, larger families like upatre (3,413 samples) and sfone (2,151 samples) display more dispersed clusters, suggesting greater internal variety in their behavioral patterns. The relationship between family size and internal similarity (Figure 9, left) demonstrates that most families maintain high internal consistency (>0.9 similarity) regardless of size, though some larger families show slightly lower cohesion. This suggests that malware families generally preserve their core behavioral characteristics even as they evolve and expand.

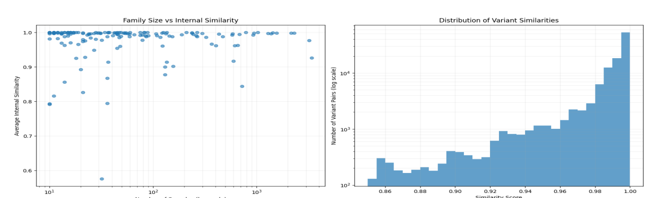


Figure 9: Family Evolution Analysis (log scale)

The distribution of cross-family variant similarities (Figure 9, right) shows largely high similarity scores (>0.98), with over 100,000 potential variant pairs compared. Time-based analysis of these relationships reveals three primary patterns: concurrent emergence, suggesting parallel development; sequential appearance, indicating potential evolutionary relationships; and hybrid cases that may represent code reuse or adaptation across different malware strains.

This analysis framework provides a quantitative basis for understanding malware family evolution and can help identify previously unknown relationships between malware strains. The high-dimensional feature space effectively captures behavioral similarities while the t-SNE visualization enables intuitive exploration of family relationships.

8.4 Behavioral Group Analysis

To validate the behavioral groups, we analyze their composition along each of the three core behavioral components used in their creation: feature distributions, behavior patterns, and local structural characteristics (11). Rather than treating similarity as a single metric, we decompose it into these constituent parts to understand how each component contributes to group cohesion. For each group, we compute both within-group and between-group similarities separately for feature distributions (weighted 40%), behavior patterns (40%), and local structures (20%). This decomposition allows us to identify which behavioral aspects are most distinctive for each group.

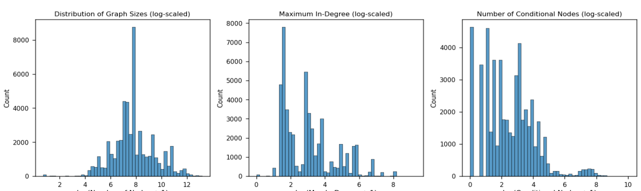


Figure 10: Features of BODMAS dataset

Graph structural analysis reveals non-trivial complexity in the control flow representations. The distribution of graph sizes follows a log-normal pattern centered around $e^8 \approx 3000$ nodes (10, left), with maximum in-degrees exhibiting heavy-tailed behavior reaching up to e^8 incoming edges (10, center). The presence of conditional nodes follows a similar distribution (10, right), indicating sophisticated control flow logic in modern malware.

For a component c and group g, we calculate a separation score: $S_{c,g} = \frac{\mu^c_{within} - \mu^c_{between}}{\sigma^c_{between}}$

where μ^c_{within} is the mean within-group similarity for component c, $\mu^c_{between}$ is the mean between-group similarity, and $\sigma^c_{between}$ is the standard deviation of between-group similarities. This score indicates how well the component separates the group from others, normalized by the variability in between-group similarities. A positive score indicates that families within the group are more similar to each other than to families in other groups, with scores above 1 suggesting strong separation.

Our analysis reveals complex relationships within the malware ecosystem through multiple complementary perspectives. The pairwise behavioral similarities between malware families exhibit a striking bimodal distribution (Figure 13), with a dominant mode centered at 0.8 encompassing 44.6% of family pairs, and a secondary mode around 0.4. This bimodality suggests fundamental organizational principles in the malware ecosystem - while some families maintain truly distinct behavioral patterns, a large proportion share significant behavioral characteristics despite their distinct classifications. The high mean similarity (0.711) and median (0.787) strongly indicate that current family-based classification systems may be overly granular, artificially separating malware variants that exhibit fundamentally similar behaviors.

Each component reveals distinct and meaningful patterns (Fig 11). The feature distribution analysis shows two clear clusters, with larger classes exhibiting higher between-group similarities (0.5-0.6) while maintaining consistent within-group relationships. This bifurcation suggests that even with minimal samples, we can extract stable feature signatures. The behavior pattern component displays a notable polarization, with some groups showing very high similarity (near 1.0) and others much lower, reflecting the dataset's inherent sparsity while highlighting families with strongly characteristic behavioral patterns. The structural component demonstrates consistent signatures (0.86-1.0 similarity) across both within-group and between-group comparisons, suggesting that structural characteristics provide robust behavioral fingerprints even in cases of minimal samples. Together, these components validate our grouping methodology by showing that meaningful behavioral relationships can be captured even in highly imbalanced conditions, with each component contributing different but complementary evidence of group cohesion.

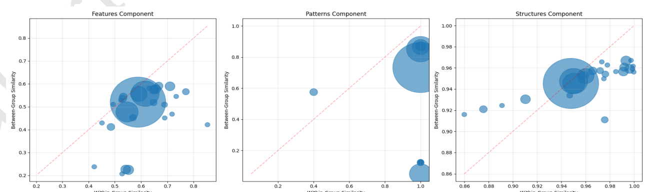


Figure 11: Component separation

The stability analysis (12) reveals that family-level performance is generally more consistent, with a mean standard deviation of 0.050 in F1 scores and most families clustered near zero deviation. Group-level performance shows slightly higher variability (mean std dev: 0.067) with a more uniform distribution of stability scores, suggesting that while behavioral grouping improves overall detection rates, it may introduce some additional temporal variance in classification performance.

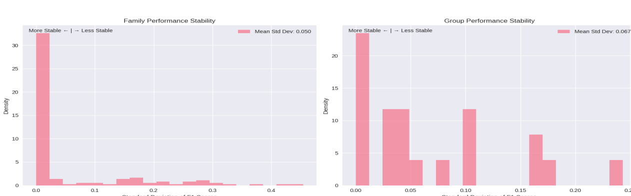


Figure 12: F1 stability

Analysis of pairwise behavioral similarities(13) between malware families reveals a bimodal distribution, suggesting two distinct relationship patterns in the malware ecosystem. The larger mode, centered around 0.8, encompasses nearly half of all family pairs (44.6% showing similarity scores above 0.8), indicating that many malware families share significant behavioral characteristics despite being classified as distinct families. A secondary mode around 0.4 likely represents truly distinct behavioral patterns. This bimodal distribution, with a mean similarity of 0.711 and median of 0.787, provides strong evidence that the current family-based classification system may be too granular - many supposedly distinct families exhibit highly similar behavioral patterns. This observation supports our approach of behavioral grouping, which can identify and consolidate these behaviorally similar families while preserving meaningful distinctions where they exist.

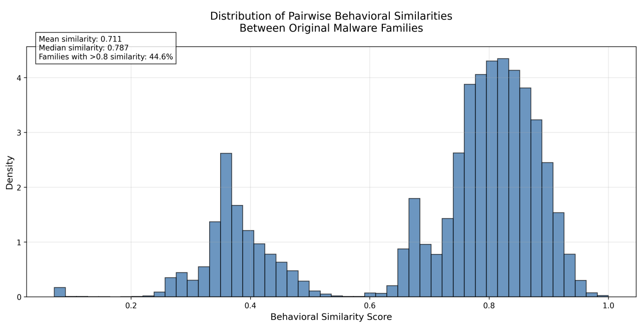


Figure 13: Behavioral Similarities

9 Discussion

Our empirical evaluation demonstrates that MalCentroid establishes new capabilities for temporal malware analysis through its hierarchical centroid-based architecture. Through evaluation on the BODMAS dataset, we demonstrate that our multi-view behavioral analysis framework achieves substantial improvements in detection efficacy while providing critical security guarantees against evasion attempts. The discovery that 44.6% of analyzed malware families share significant behavioral characteristics (similarity > 0.8) reveals fundamental organizational principles within the malware ecosystem that can be exploited for improved detection strategies.

The dual-level classification architecture provides multiple security advantages through its centroid-based methodology. By maintaining multiple behavioral prototypes per family, the system effectively captures polymorphic variants while preserving interpretable detection boundaries. The multi-view architecture substantially increases the complexity of evasion attempts by requiring adversaries to simultaneously bypass both control flow graph analysis and instruction-level detection mechanisms. Our security analysis quantifies this through the compound evasion probability:

$$P_{evasion} = P(evade|modify(CFG)) \cdot P(evade|modify(instr))$$

These theoretical security advantages are further supported by our comparative robustness analysis using the MalImg dataset. While the MalImg CNN achieves higher base performance (precision: 0.936 vs 0.167), it demonstrates significantly higher vulnerability to adversarial manipulation, with dramatic performance degradation under attack for known classes. In contrast, MalCentroid maintains more stable performance characteristics under adversarial pressure, particularly for known family detection, suggesting that behavioral analysis provides inherent robustness advantages over image-based features.

Our temporal analysis through chronological evaluation of BODMAS reveals patterns in malware evolution, with behavioral drift rates averaging 0.142 (σ=0.067) per month for active families. The strong temporal clustering of novel family emergence, with 64% of new families appearing within two-week windows of behaviorally similar variants, suggests coordinated development patterns in the malware ecosystem. The behavioral group abstraction maintains remarkable stability across temporal boundaries, demonstrated by the consistent correlation (Pearson=0.823) between family-level and group-level similarity metrics.

The framework demonstrates robust detection capabilities even under extreme class imbalance conditions, with the BODMAS group-level classifier achieving precision of 0.8061 and recall of 0.5433 while requiring 13× fewer parameters through behavioral abstractions. This reduction in model complexity provides significant operational advantages while maintaining detection efficacy. Critical limitations emerge in novel family detection, where absolute performance remains modest (Novel F1: 0.027) due to fundamental challenges in distinguishing truly novel behaviors from extreme variants of known families.

10 Conclusion

MalCentroid advances the state-of-the-art in malware classification by introducing a temporal-aware framework that effectively tracks behavioral evolution while maintaining robust detection capabilities. Our evaluation on BODMAS demonstrates significant improvements in temporal malware analysis, while our comparative study against image-based approaches using MalImg highlights a fundamental security tradeoff: while methods like MalImg CNN can achieve higher base accuracy, they exhibit brittle performance under adversarial pressure. MalCentroid's behavioral analysis approach provides more stable detection capabilities in hostile environments, suggesting that future malware detection systems should prioritize robust behavioral features.

The framework's ability to maintain performance across different granularities while adapting to emerging threats represents a substantial advancement over existing methods that typically examine limited family sets with non-temporal evaluation protocols. These contributions provide crucial capabilities for real-world malware defense systems operating in adversarial environments where behavioral evolution and novel threats pose continuous challenges.

Future work should focus on exploring how behavioral analysis techniques could be combined with other approaches to maintain both accuracy and robustness. Our framework's demonstrated ability to capture and adapt to behavioral evolution while maintaining interpretable detection boundaries establishes a new paradigm for temporal malware analysis that better reflects the dynamic nature of modern threats.

References

[1] Danilo Bruschi, Lorenzo Martignoni, and Mattia Monga. 2006. Detecting self-mutating malware using control-flow graph matching. In *Detection of Intrusions and Malware & Vulnerability Assessment: Third International Conference, DIMVA 2006, Berlin, Germany, July 13-14, 2006. Proceedings 3*. Springer, 129–143.

[2] Julian Busch, Anton Kocheturov, Volker Tresp, and Thomas Seidl. 2021. NF-GNN: network flow graph neural networks for malware detection and classification. In *Proceedings of the 33rd International Conference on Scientific and Statistical Database Management*. 121–132.

[3] Asim Darwaish, Farid Naït-Abdesselam, Chafiq Titouna, and Sumera Sattar. 2021. Robustness of image-based android malware detection under adversarial attacks. In *ICC 2021-IEEE International Conference on Communications*. IEEE, 1–6.

[4] Yun Gao, Hirokazu Hasegawa, Yukiko Yamaguchi, and Hajime Shimada. 2022. Malware Detection by Control-Flow Graph Level Representation Learning With Graph Isomorphism Network. *IEEE Access* 10 (2022), 111830–111841. doi:10.1109/ACCESS.2022.3215267

[5] Yun Gao, Hirokazu Hasegawa, Yukiko Yamaguchi, and Hajime Shimada. 2022. Malware Detection by Control-Flow Graph Level Representation Learning With Graph Isomorphism Network. *IEEE Access* 10 (2022), 111830–111841.

[6] Houssem Gasmi, Jannik Laval, and Abdelaziz Bouras. 2019. Information extraction of cybersecurity concepts: An LSTM approach. *Applied Sciences* 9, 19 (2019), 3945.

[7] Steven Strandlund Hansen, Thor Mark Tampus Larsen, Matija Stevanovic, and Jens Myrup Pedersen. 2016. An approach for detection and family classification of malware based on behavioral analysis. In *2016 International conference on computing, networking and communications (ICNC)*. IEEE, 1–5.

[8] Shou-Ching Hsiao, Da-Yu Kao, Zi-Yuan Liu, and Raylin Tso. 2019. Malware image classification using one-shot learning with siamese networks. *Procedia Computer Science* 159 (2019), 1863–1871.

[9] Mahmoud Kalash, Mrigank Rochan, Noman Mohammed, Neil DB Bruce, Yang Wang, and Farkhund Iqbal. 2018. Malware classification with deep convolutional neural networks. In *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*. IEEE, 1–5.

[10] Omid Kargarnovin, Amir Mahdi Sadeghzadeh, and Rasool Jalili. 2021. Mal2GCN: a robust malware detection approach using deep graph convolutional networks with non-negative weights. *arXiv preprint arXiv:2108.12473* (2021).

[11] Bojan Kolosnjaji, Apostolis Zarras, Tamas Lengyel, George Webster, and Claudia Eckert. 2016. Adaptive semantics-aware malware classification. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings 13*. Springer, 419–439.

[12] Lakshmanan Nataraj, Sreejith Karthikeyan, Gregoire Jacob, and Bangalore S Manjunath. 2011. Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security*. 1–7.

[13] Shruti Patil, Vijayakumar Varadarajan, Devika Walimbe, Siddharth Gulechha, Sushant Shenoy, Aditya Raina, and Ketan Kotecha. 2021. Improving the robustness of ai-based malware detection using adversarial machine learning. *Algorithms* 14, 10 (2021), 297.

[14] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles Nicholas. 2017. Malware detection by eating a whole exe. *arXiv preprint arXiv:1710.09435* (2017).

[15] Kamran Shaukat, Suhuai Luo, and Vijay Varadharajan. 2022. A novel method for improving the robustness of deep learning-based malware detectors against adversarial attacks. *Engineering Applications of Artificial Intelligence* 116 (2022), 105461.

[16] Hamish Spencer, Wei Wang, Ruoxi Sun, and Minhui Xue. 2022. Dissecting Malware in the Wild. In *Australasian Computer Science Week 2022*. 56–64.

[17] Guillermo Suarez-Tangil, Juan E Tapiador, Pedro Peris-Lopez, and Arturo Ribagorda. 2013. Evolution, detection and analysis of malware for smart devices. *IEEE communications surveys & tutorials* 16, 2 (2013), 961–987.

[18] Octavian Suciu, Scott E Coull, and Jeffrey Johns. 2019. Exploring adversarial examples in malware detection. In *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 8–14.

[19] Mayuri Wadkar, Fabio Di Troia, and Mark Stamp. 2020. Detecting malware evolution using support vector machines. *Expert Systems with Applications* 143 (2020), 113022.

[20] Gérard Wagener, Radu State, and Alexandre Dulaunoy. 2008. Malware behaviour analysis. *Journal in computer virology* 4 (2008), 279–287.

[21] Jiaqi Yan, Guanhua Yan, and Dong Jin. 2019. Classifying malware represented as control flow graphs using deep graph convolutional neural network. In *2019 49th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 52–63.

[22] Limin Yang, Arridhana Ciptadi, Ihar Laziuk, Ali Ahmadzadeh, and Gang Wang. 2021. BODMAS: An open dataset for learning based temporal analysis of PE malware. In *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 78–84.

.1 Proof of Differentiability

To prove the differentiability of L_j with respect to $f(x)$, we will calculate its gradient, ∇L_j.

First, we express L_j in terms of the individual components of $f(x)$ and c_j (where c_j are centroids):

$$L_j = \sum_{i=1}^{m} (f(x)_i - c_j)_i^2, \tag{37}$$

where $(f(x)_i - c_j)_i$ denotes the i-th component of the vectors $f(x)$ and c_j.

Now, we compute the partial derivative of L_j with respect to the k-th component of $f(x)$:

$$\begin{aligned} \frac{\partial L_j}{\partial f(x)_k} &= \frac{\partial}{\partial f(x)_k} \sum_{i=1}^{m} (f(x)_i - c_j)_i^2 & (38) \\ &= \sum_{i=1}^{m} \frac{\partial}{\partial f(x)_k} (f(x)_i - c_j)_i^2 & (39) \\ &= 2 \sum_{i=1}^{m} (f(x)_i - c_j)_i \delta_{ik}, & (40) \end{aligned}$$

where δ_{ik} is the Identity matrix, which is 1 when $i = k$ and 0 otherwise.

We combine these partial derivatives to form the gradient vector ∇L_j:

$$\begin{aligned} \nabla L_j &= \left(\frac{\partial L_j}{\partial f(x)_1}, \frac{\partial L_j}{\partial f(x)_2}, \ldots, \frac{\partial L_j}{\partial f(x)_m} \right) & (41) \\ &= 2 \left((f(x)_1 - c_j)_1, (f(x)_2 - c_j)_2, \ldots, (f(x)_m - c_j)_m \right) & (42) \\ &= 2 \left(f(x)_1 - c_j)_1, f(x)_2 - c_j)_2, \ldots, (f(x)_m - c_j)_m \right) & (43) \\ &= 2(f(x) - c_j). & (44) \end{aligned}$$

Therefore, the gradient ∇L_j of L_j with respect to $f(x)$ is $2(f(x) - c_j)$.

Proposition 1. *Optimizing Graph Centroid Model with Classification Loss*

When a Graph Centroid Model is optimized with a standard classification loss, it results in an effective classifier.

We consider a Graph Centroid Model as defined by the function $F(x) = y_{i^*}$, where $i^* = \arg\min_j ||f(x) - c_j||$. Here, f represents the neural network that generates feature representations, c_j are the centroids, and y_{i^*} is the predicted label.

The optimization process aims to minimize the classification loss, typically measured using a loss function such as cross-entropy, which is what we use in our methodology. This loss is defined as $L = -\sum_k y_k \log(p_k)$, where y_k is the ground truth label and p_k is the predicted probability for class k. In the context of the Graph Centroid Model, p_k corresponds to the probability of selecting centroid k based on the feature representations generated by the GCN.

Let us denote the classification loss as L_{class}. The optimization process involves updating the centroids c_j and the neural network weights to minimize L_{class}. Mathematically, we have:

Table 3: Summary of BODMAS and MalImg Characteristics

	Summary of BODMAS Characteristics			Summary of MalImg Characteristics		
Statistic	Training Set	Validation Set	Test Set	Training Set	Validation Set	Test Set
Number of graphs	35,200	7,542	7,544	4,541	1,002	1,034
Number of known families	487	124	140	22	22	22
Number of novel families	0	34	43	0	2	2
Number of graphs with known families	35,200	7,476	7,387	4,541	968	995
Number of graphs with novel families	0	66	157	0	34	39
Average samples per known family	72.28	60.29	52.76	206.41	44.00	45.23
Average samples per novel family	0	1.94	3.65	0	17.00	19.50

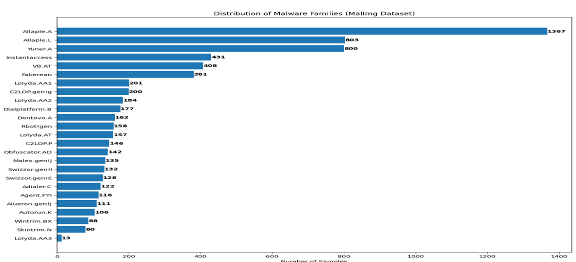


Figure 15: MalImg dataset distribution.

$$\min_{c_j, weights} L_{class}(F(x), y)$$

where y represents the ground truth labels.

As optimization progresses, the centroids c_j adapt to the specific feature representations generated by the neural network f. This adaptation occurs because the loss encourages the network to produce feature representations that effectively discriminate between different classes.

As the optimization converges, the Graph Centroid Model assigns inputs to centroids that are close in feature space. This means that inputs with similar feature representations will be assigned to the same or nearby centroids, leading to effective classification. This behavior aligns with the optimization objective of minimizing the classification loss.

Optimizing the Graph Centroid Model with a standard classification loss function allows us to adapt centroids and neural network features to more effectively discriminate between classes. Therefore, the Graph Centroid Model serves as a trainable component of a larger neural network.

.2 Dataset info

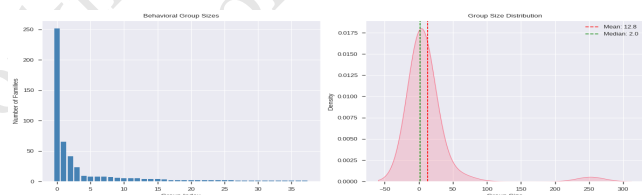


Figure 14: Behavioral Group Distribution - BODMAS

The long-tailed distribution (Figure 14) poses significant challenges for traditional classification approaches. The malware type distribution (Figure 8) shows trojans dominating the ecosystem (>300 families), followed by worms (75 families) and backdoors (30 families), reflecting real-world prevalence patterns.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009