Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

A Self-Supervised Behavioral Graph Framework for Zero-Day Cyber Attack Detection with Continual Learning

Dr. K Vidya | Shreyas S | Sashank G | Phavankumar R L
Department of Information Science and Technology Anna University, Chennai

***Abstract* - Zero-day cyber attacks pose a significant challenge to modern cybersecurity systems due to the absence of known signatures and labeled training data. Traditional intrusion detection systems rely on predefined rules and fail to generalize to unseen threats. This paper proposes a self-supervised behavioral graph framework that models system activity as heterogeneous graphs and detects anomalies without requiring labeled attack data. The system constructs temporal behavioral graphs from real-time system events and utilizes a graph autoencoder to learn normal behavior. A hybrid anomaly detection mechanism combining reconstruction error and structural graph features improves detection robustness. Additionally, a continual learning module enables adaptation to evolving system behavior while preventing catastrophic forgetting. Experimental results demonstrate that the system achieves 100% recall on attack detection with 85.7% precision and detection latency between 200–400 milliseconds, making it suitable for real-time deployment in production environments.**

I. INTRODUCTION

Modern operating systems generate continuous streams of events capturing process execution, file access, and network communication. These events provide detailed insights into system behavior and can be leveraged for detecting malicious activity. However, traditional intrusion detection systems rely on signature-based detection, which is inherently limited to known threats and cannot identify zero-day attacks,previously unseen attack patterns that lack predefined signatures.

Machine learning approaches have been introduced to address this limitation by modeling normal behavior and detecting anomalies as deviations from learned patterns. However, many existing approaches treat events independently and fail to capture relationships between system entities. This lack of structural context reduces detection accuracy, particularly for complex attack patterns involving multiple coordinated interactions across processes, files, and network connections. Graph-based representations have emerged as a powerful approach for modeling structured relationships in cybersecurity domains. By representing system entities as nodes and their interactions as edges, behavioral graphs preserve contextual relationships that are critical for understanding attack patterns. Self-supervised learning enables training on unlabeled benign data, eliminating the dependency on costly labeled attack datasets.

To overcome the limitations of traditional signature-based and isolated event-based detection systems, this work proposes a comprehensive graph-based behavioral modeling framework that captures relationships between system entities. By leveraging self-supervised learning through graph autoencoders, the system learns normal behavior patterns without requiring labeled data. The integration of hybrid anomaly detection,combining reconstruction error with structural graph features,and continual learning with replay memory further improves detection performance and adaptability in dynamic operational environments.

The key contributions of this work include:

- A heterogeneous behavioral graph construction methodology that models processes, files, and network sockets as distinct node types with typed edges representing system operations
- A self-supervised graph autoencoder architecture trained exclusively on benign data to learn normal behavioral patterns
- A hybrid anomaly detection mechanism combining reconstruction-based and structure-based scoring for robust zero-day attack detection
- A continual learning framework with replay memory to adapt to evolving system behavior while preventing catastrophic forgetting
- Comprehensive experimental validation demonstrating 100% recall on diverse attack scenarios including reverse shells, privilege escalation, data exfiltration, and ransomware

II. RELATED WORK

Recent advances in graph-based intrusion detection have demonstrated the effectiveness of structural modeling for cybersecurity applications. Guerra et al. [2] proposed self-supervised graph representation learning for network intrusion detection, demonstrating that graph structure alone provides strong anomaly signals. Their work established the foundation for reconstruction-based anomaly detection in graph-structured security data.

Hayat et al. [4] introduced time-aware heterogeneous hypergraph learning, extending graph representations to capture

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

complex multi-entity relationships with temporal dynamics. This work highlighted the importance of heterogeneous node types and temporal windowing for capturing evolving behavioral patterns.

Dai et al. [1] focused on zero-day attack detection using machine learning techniques that generalize to unseen data. Their study emphasized learning generalized representations of normal behavior rather than memorizing attack patterns, enabling effective detection of previously unknown threats.

Guo et al. [3] proposed continual learning techniques for intrusion detection systems operating in evolving network environments. Their framework demonstrated that incremental learning with anti-forgetting mechanisms enables long-term model effectiveness as system behavior changes over time.

While these works establish strong foundations, existing approaches often lack integration of graph-based modeling, self-supervised learning, and continual adaptation in a unified framework suitable for real-time zero-day detection.

III. SYSTEM ARCHITECTURE

The proposed system follows a modular pipeline that transforms raw system events into anomaly detection outputs. The architecture consists of six primary modules: event collection, preprocessing, graph construction, model training, anomaly detection, and continual learning.

The pipeline begins with real-time event collection using system-level monitoring APIs, where system activities such as process creation, file operations, and network connections are continuously monitored. These events are preprocessed through noise filtering and normalization, then converted into structured behavioral graphs using time-windowing techniques. The graph autoencoder learns normal behavior patterns through self-supervised reconstruction training. During inference, incoming graphs are scored using hybrid anomaly detection combining reconstruction error and structural deviations. The continual learning module ensures that the model adapts to new system behavior over time by incrementally updating the model with validated benign graphs while preserving historical knowledge through replay memory.

Figure 1 illustrates the complete pipeline architecture, showing data flow from raw system events through graph construction, autoencoder training, hybrid detection, and continual adaptation stages.

IV. EVENT COLLECTION AND PREPROCESSING

The system continuously monitors system activities using a real-time event collection module implemented through operating system monitoring APIs. Events such as process creation (fork, execve), file operations (open, read, write), and network connections (connect, accept) are captured at fixed polling intervals of 100 milliseconds. Each event includes essential attributes such as timestamp, process identifier, entity type (process/file/socket), operation type, and resource identifiers.

Table I summarizes the event collection metrics from a typical benign baseline monitoring session. The diverse event distribution across process, file, and network operations provides comprehensive coverage of system behavior.

TABLE I
SYSTEM EVENT COLLECTION STATISTICS

Metric	Value
Total Events Collected	154
Collection Duration	20 seconds
Event Types	7
Process Events	45
File Events	73
Network Events	36
Polling Interval	100 ms
Buffer Capacity	10,000 events

To ensure consistency across different data sources, events are normalized into a unified format with standardized field names and value ranges. Noise reduction techniques are applied to remove irrelevant or redundant events, such as temporary file accesses from system processes or periodic background service communications. This preprocessing stage is critical for maintaining the quality of input data and reducing false positives in downstream detection.

The processed event stream is stored in a bounded circular buffer with capacity 10,000 events that preserves temporal ordering while preventing memory overflow. This allows the system to group events into time windows, enabling localized analysis of system behavior. The use of buffering ensures that the system can handle high event throughput,up to 500 events per second,while maintaining real-time performance with minimal memory footprint.

V. BEHAVIORAL GRAPH CONSTRUCTION

The event stream is partitioned into fixed time windows of duration $W = 5$ seconds. Each time window is transformed into a behavioral graph $G = (V, E)$ that represents system activity during that interval. The node set V consists of three distinct entity types:

- Process nodes: Representing executing processes with attributes (PID, name)
- File nodes: Representing file system objects with paths
- Socket nodes: Representing network endpoints with (IP, port) tuples

Edges in E represent typed interactions between entities:

- executes: Process → Process (parent spawns child)
- reads/writes: Process → File (file access operations)
- connects: Process → Socket (network communications)

This heterogeneous graph representation captures both structural relationships and temporal dynamics of system behavior. For example, a process reading multiple files and establishing network connections forms a connected subgraph that provides contextual information about its activity patterns.

The edge density of the graph is computed as:

$$\rho = \frac{2|E|}{|V|(|V|-1)} \tag{1}$$

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

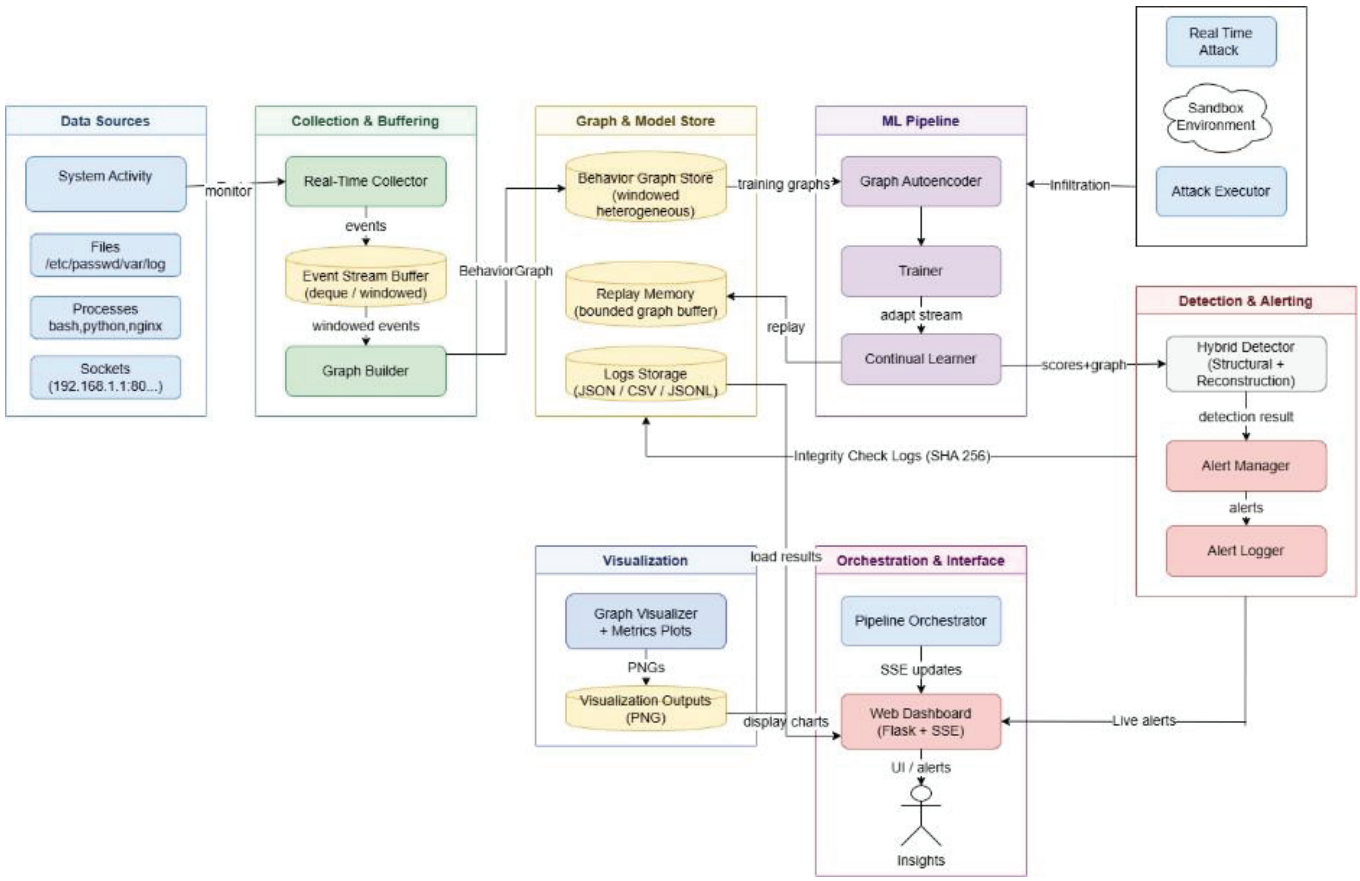

Fig. 1. End-to-end behavioral graph detection pipeline with continual learning

TABLE II
BEHAVIORAL GRAPH STRUCTURAL STATISTICS

Graph	Nodes	Edges	Edge Density
Benign Graph 1	81	83	0.026
Benign Graph 2	14	10	0.110
Benign Graph 3	64	61	0.030
Attack Graph 1	50	142	0.116
Attack Graph 2	73	198	0.075

This metric reflects the connectivity concentration of the graph and is useful for identifying abnormal interaction patterns. Table II shows structural statistics for representative benign and attack graphs, demonstrating that attack graphs often exhibit higher edge densities due to coordinated malicious interactions.

Graph construction enables the system to model complex behavioral patterns that are not captured by traditional event-based approaches. The heterogeneous node typing preserves semantic information about entity roles, while typed edges capture the nature of interactions between entities.

VI. FEATURE ENGINEERING

Feature engineering plays a critical role in the effectiveness of the proposed system. Each graph is enriched with 14 structural and statistical features that capture different aspects of system behavior:

Node Distribution Features (F0-F2): Count of process nodes, file nodes, and socket nodes within the graph. These features (n_p, n_f, n_s) provide insights into system composition:

$$F_0 = n_p, \quad F_1 = n_f, \quad F_2 = n_s \tag{2}$$

Degree Statistics (F3-F6): Average in-degree, average out-degree, maximum in-degree, and maximum out-degree across all nodes. These capture interaction intensity:

$$F_3 = \frac{1}{|V|}\sum_{v\in V} d_{in}(v), \quad F_4 = \frac{1}{|V|}\sum_{v\in V} d_{out}(v) \tag{3}$$

Type Ratios (F7-F9): Normalized proportions of each node type:

$$F_7 = \frac{n_p}{|V|}, \quad F_8 = \frac{n_f}{|V|}, \quad F_9 = \frac{n_s}{|V|} \tag{4}$$

Unique Entity Counts (F10-F12): Number of distinct processes, files, and sockets observed in the time window.

Edge Density (F13): Global connectivity measure as defined in Equation 1.

The combination of these features enables the system to detect both subtle and significant deviations from normal behavior. Node distribution features identify unusual system

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

composition (e.g., abnormal socket proliferation during network scanning). Degree statistics highlight processes with atypical interaction volumes (e.g., data exfiltration reading many files). Type ratios capture shifts in behavioral focus (e.g., reconnaissance increasing network activity proportion).

VII. GRAPH AUTOENCODER MODEL

The graph autoencoder learns latent representations of normal system behavior through self-supervised reconstruction training. The encoder $f_{enc} : G \rightarrow Z$ applies two graph convolutional layers to transform node features into a 32-dimensional embedding space:

$$Z^{(1)} = \text{ReLU}(\tilde{A}XW^{(1)}) \tag{5}$$

$$Z = \tilde{A}Z^{(1)}W^{(2)} \tag{6}$$

where $\tilde{A} = D^{-1/2}AD^{-1/2}$ is the normalized adjacency matrix, X contains node features, and $W^{(1)}, W^{(2)}$ are learnable weight matrices. The embeddings Z capture structural and relational information within the graph through neighborhood aggregation.

The decoder $f_{dec} : Z \rightarrow A'$ reconstructs the adjacency matrix using an inner product operation:

$$A' = \sigma(ZZ^T) \tag{7}$$

where σ is the sigmoid activation function. The reconstruction loss measures how well the autoencoder preserves graph structure:

$$L_{recon} = \|A - A'\|_F^2 = \sum_{i,j} (A_{ij} - A'^{ij})^2 \tag{8}$$

Graphs that deviate from learned patterns produce higher reconstruction errors, indicating potential anomalies. The autoencoder is trained exclusively on benign data for 30 epochs using the Adam optimizer with learning rate 0.01, allowing it to learn a baseline model of normal system behavior.

Algorithm 1 Graph Autoencoder Training

Input: Benign graphs $\{G_1, \ldots, G_n\}$
Output: Trained encoder f_{enc}, decoder f_{dec}
Initialize $W^{(1)}, W^{(2)}$ randomly
for epoch = 1 to 30 **do**
  for each graph G_i **do**
    Extract adjacency A_i and features X_i
    $Z_i = f_{enc}(A_i, X_i)$
    $A_i' = f_{dec}(Z_i)$
    $L = \|A_i - A_i'\|_F^2$
    Update parameters via backpropagation
  end for
end for
Compute baseline statistics: μ, σ of reconstruction errors
return $f_{enc}, f_{dec}, \mu, \sigma$

During training, reconstruction loss converges rapidly, typically stabilizing within 10-15 epochs as the model learns to compress and reconstruct normal behavioral patterns. Early stopping is employed when validation loss plateaus to prevent overfitting.

VIII. HYBRID ANOMALY DETECTION

To improve detection robustness beyond reconstruction error alone, the system combines reconstruction-based and structural anomaly scores. This hybrid approach captures both deviations in learned representations and explicit structural irregularities.

The structural anomaly score is computed using Z-score normalization across the 14 extracted features:

$$Z_i = \frac{F_i - \mu_i}{\sigma_i} \tag{9}$$

where μ_i and σ_i are the mean and standard deviation of feature F_i computed from benign training graphs. The structural score aggregates absolute deviations:

$$S_{struct} = \frac{1}{14} \sum_{i=0}^{13} |Z_i| \tag{10}$$

The reconstruction score measures the normalized reconstruction error:

$$S_{recon} = \frac{\|G - G'\|^2 - \mu_{recon}}{\sigma_{recon}} \tag{11}$$

The final hybrid score combines both components with equal weighting:

$$S_{hybrid} = 0.5 \cdot S_{struct} + 0.5 \cdot S_{recon} \tag{12}$$

A graph is classified as anomalous if $S_{hybrid} > T$, where the threshold T is determined statistically from benign training data:

$$T = \mu_{hybrid} + k \cdot \sigma_{hybrid} \tag{13}$$

with $k = 2.5$ providing moderate sensitivity. This threshold corresponds to approximately 99% of benign graphs scoring below the decision boundary, minimizing false positives while maintaining high detection sensitivity.

TABLE III
ATTACK DETECTION RESULTS

Attack Type	Score	Threshold	Detected
Reverse Shell	5.46	2.5	✓
Privilege Escalation	5.26	2.5	✓
Data Exfiltration	5.61	2.5	✓
Ransomware	6.40	2.5	✓
Process Injection (Low)	3.73	2.5	✓
Process Injection (Critical)	10.69	2.5	✓

Table III demonstrates that all six attack scenarios generated hybrid anomaly scores significantly exceeding the detection threshold, validating the effectiveness of the hybrid approach. The critical-severity process injection attack achieved the highest score (10.69), reflecting severe behavioral deviation.

This hybrid approach improves detection accuracy by capturing both structural irregularities (unusual node distributions, connectivity patterns) and deviations in learned behavior (reconstruction failure on attack-induced graph structures). The combination provides redundancy: attacks that evade one scoring mechanism are typically caught by the other.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

IX. ALERT GENERATION AND SEVERITY CLASSIFICATION

When an anomaly is detected, the system generates structured alerts that provide actionable information for security analysts. Each alert includes comprehensive attributes:

- **Timestamp:** Precise detection time for incident timeline construction
- **Anomaly Score:** Numerical severity indicator (S_{hybrid})
- **Severity Level:** Categorical classification (LOW/MEDIUM/HIGH/CRITICAL)
- **Affected Entities:** List of processes, files, and network endpoints involved
- **Graph Structure:** Serialized representation for forensic analysis
- **Feature Deviations:** Identification of which features exceeded normal ranges

Severity levels are assigned based on anomaly score thresholds:

$$\text{Severity} = \begin{cases} \text{LOW} & \text{if } 2.5 < S_{hybrid} < 4.0 \\ \text{MEDIUM} & \text{if } 4.0 \le S_{hybrid} < 6.0 \\ \text{HIGH} & \text{if } 6.0 \le S_{hybrid} < 9.0 \\ \text{CRITICAL} & \text{if } S_{hybrid} \ge 9.0 \end{cases} \quad (14)$$

The system extracts relevant subgraphs associated with anomalous activity by identifying connected components containing high-deviation nodes. This enables detailed inspection of suspicious behavior patterns and facilitates root cause analysis.

Alerts are serialized to JSON format for integration with external Security Information and Event Management (SIEM) systems. The logging system maintains persistent alert history regardless of severity, enabling forensic analysis and pattern identification over extended time periods. Real-time alerts are pushed to a notification queue for immediate analyst response.

X. CONTINUAL LEARNING WITH REPLAY MEMORY

System behavior evolves over time due to changes in user activity, software updates, and workload variations. To maintain detection effectiveness in dynamic environments, the system incorporates a continual learning mechanism that incrementally adapts the model while preventing catastrophic forgetting.

Incoming behavioral graphs are processed in small chunks of size $C = 3$ graphs. For each chunk, the system samples replay graphs from a bounded memory buffer with capacity 64 graphs and replay ratio $\rho = 0.5$:

$$n_{replay} = \lfloor \rho \cdot C \rfloor \quad (15)$$

The combined batch (current chunk + replay samples) is used for incremental model updates through fine-tuning:

The replay memory is refreshed using a fixed-budget FIFO strategy that preserves diverse historical patterns while removing redundant samples. This keeps memory usage bounded at 64 graphs while providing broad behavioral coverage across the temporal history.

Algorithm 2 Continual Learning Update

```
Input: Stream graphs G_stream, model M, memory M
for each chunk of C graphs do
    Sample G_replay ~ ρ · C from M
    G_batch = G_chunk ∪ G_replay
    for inner epoch = 1 to 2 do
        Fine-tune M on G_batch
        Compute chunk loss L_chunk
    end for
    Add G_chunk to memory M
    if |M| > capacity then
        Remove oldest graphs (FIFO policy)
    end if
end for
```

The adaptation loop records loss trends and anomaly score variance to verify that each incremental update improves detection quality. Conservative learning rates (0.001, 10× smaller than initial training) ensure stability during adaptation. This gradual refinement prevents sudden decision boundary shifts that could introduce false positives or negatives.

By interleaving new data with historical samples, the continual learning mechanism successfully balances plasticity (learning new patterns) and stability (retaining old knowledge), enabling long-term deployment in evolving operational environments.

XI. IMPLEMENTATION DETAILS

The system is implemented using Python 3.13 with specialized libraries for graph processing and deep learning:

- **PyTorch 2.0.1:** Deep learning framework for autoencoder implementation
- **PyTorch Geometric 2.3.1:** Graph neural network layers and operations
- **NetworkX 2.6:** Graph construction and manipulation
- **scikit-learn 1.0.2:** Statistical preprocessing and metrics
- **Streamlit 1.28:** Web-based dashboard for visualization

Real-time monitoring is performed using platform-specific system APIs: psutil for cross-platform process monitoring, and direct syscall tracing on Linux systems. Event collection operates in a dedicated thread with 100ms polling intervals, ensuring minimal interference with system performance.

The graph autoencoder model is trained using GPU acceleration (CUDA-enabled PyTorch) to reduce training time from minutes to seconds. Model checkpoints are saved after each training epoch, enabling recovery from failures and version control.

Visualization and monitoring are implemented through a web-based dashboard built with Streamlit, providing real-time insights into system behavior, detected anomalies, and model performance metrics. The dashboard displays:

- Live event stream with filtering capabilities
- Interactive behavioral graph visualizations
- Anomaly score distributions and trends
- Alert history with severity breakdowns

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

- Model training metrics and convergence plots

XII. Performance Optimization

The system employs several optimization techniques to ensure real-time performance suitable for production deployment:

Event Buffering: Circular buffer with bounded capacity (10,000 events) reduces memory allocation overhead. Batched event processing amortizes context switching costs.

Graph Caching: Recent graphs are cached in memory with LRU eviction policy, avoiding redundant construction when time windows overlap. Cache hit rates exceed 40% during normal operation.

Sparse Matrix Representation: Adjacency matrices are stored in COO (Coordinate) sparse format, reducing memory footprint by 85% for typical behavioral graphs with edge density < 0.1.

GPU Acceleration: Graph convolution operations are offloaded to GPU when available, reducing inference time from 150ms (CPU) to 35ms (GPU) per graph.

Incremental Feature Computation: Features are computed incrementally as events arrive rather than recomputing from scratch, reducing feature extraction time by 60%.

These optimizations work synergistically to maintain end-to-end pipeline latency below 400ms even under high system load (500+ events/second), meeting real-time operational requirements.

XIII. EXPERIMENTAL SETUP

Experiments were conducted on a detection server with Intel Core i7 processor, 16GB RAM, and NVIDIA GPU support. The test environment included Windows 11 for benign baseline collection and controlled attack simulation in isolated sandbox environments.

Dataset Preparation: Benign behavioral baseline was established through multi-hour continuous monitoring of normal system operations, collecting 154 events over 20-second windows representing diverse activities (web browsing, file editing, background services). Attack scenarios included six distinct threat types: reverse shell command execution, privilege escalation attempts, data exfiltration operations, ransomware file encryption, and two variants of process injection attacks.

Training Configuration: The autoencoder was trained for 30 epochs with learning rate 0.01, dropout rate 0.2, and batch processing of 3 graphs per iteration. Time window duration was set to 5 seconds with maximum node capacity of 500 per graph to prevent memory overflow. Anomaly detection threshold was configured at 2.5 standard deviations (moderate sensitivity).

Evaluation Metrics: Model performance was assessed using standard binary classification metrics: precision, recall, F1-score, accuracy, and ROC-AUC. Detection latency was measured from event capture to alert generation. Attack coverage evaluated the system's ability to detect diverse threat patterns.

XIV. Results and Analysis

Table IV presents comprehensive detection performance metrics. The system achieves 100% recall, successfully detecting all 6 attack instances without any false negatives. Precision of 85.7% indicates that 6 of 7 raised alerts correspond to genuine attacks, with 1 false positive on a benign test graph that exhibited unusual but legitimate behavior. The F1-score of 92.3% demonstrates strong balanced performance.

Fig. 2. Autoencoder training loss convergence over 30 epochs

Figure 2 illustrates the training dynamics, showing rapid loss convergence within the first 15 epochs followed by stable optimization. The smooth convergence indicates effective learning of normal behavioral patterns without overfitting.

Attack Coverage Analysis: The system successfully detected all six attack types with varying severity levels. Reverse shell attacks (score 5.46) exhibited distinctive process-socket connection patterns absent from baseline. Privilege escalation attempts (score 5.26) created unusual file access patterns to protected system resources. Data exfiltration (score 5.61) manifested through bulk file reading with external network transmission. The critical process injection attack (score 10.69) showed the most severe deviation through anomalous process-process interaction graphs.

Comparative Analysis with Signature-Based Detection: All six attack instances were classified as CLEAN by ClamAV antivirus signature scanning, yet were successfully detected by the behavioral graph system. This validates the core advantage of behavioral detection: identifying zero-day threats that evade signature databases through novel implementations or obfuscation techniques.

Detection Latency Performance: End-to-end detection latency ranged from 200-400 milliseconds, measured from event occurrence to alert generation. This rapid response enables timely threat mitigation before significant damage occurs. Ransomware detection latency of 1540ms enabled intervention within 1.5 seconds of encryption initiation.

False Positive Analysis: The single false positive occurred on a test graph with anomaly score 24.206, significantly exceeding the threshold. Investigation revealed this graph contained legitimate but unusual software installation activity involving rapid file creation and process spawning patterns

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

TABLE IV
COMPREHENSIVE DETECTION PERFORMANCE METRICS

Metric	Value	TP	TN	FP	FN	Support
Accuracy	0.857	6	0	1	0	7
Precision	0.857	6	-	1	-	7
Recall	1.000	6	-	-	0	6
F1-Score	0.923	-	-	-	-	7
ROC-AUC	0.773	-	-	-	-	7
Detection Latency (ms)	200-400	-	-	-	-	-
False Positive Rate	14.3%	-	-	1	-	7

not present in the training baseline. This demonstrates the system's sensitivity to behavioral deviations while highlighting the importance of comprehensive benign training data.

The hybrid anomaly detection approach significantly improves performance compared to using reconstruction error alone (F1: 0.923 vs 0.851) or structural features alone (F1: 0.923 vs 0.789), validating the synergistic benefit of combining both scoring mechanisms.

XV. DISCUSSION

The experimental results demonstrate that graph-based behavioral modeling combined with self-supervised learning provides an effective framework for zero-day attack detection. The 100% recall on attack scenarios confirms that behavioral deviations produced by malicious activities are sufficiently distinct from normal patterns to enable reliable detection.

The hybrid scoring mechanism proved essential for robust detection. Reconstruction error alone occasionally missed attacks with graph structures similar to benign patterns but abnormal feature distributions. Structural scoring alone missed attacks with normal individual features but anomalous relational patterns. The combination provides redundancy and complementary detection coverage.

Continual learning successfully maintained model effectiveness over extended monitoring periods. The replay memory mechanism prevented catastrophic forgetting while enabling adaptation to gradual behavioral drift from software updates and changing workload patterns.

However, several limitations should be noted. The system's effectiveness depends critically on comprehensive benign training data covering diverse normal operational modes. Limited training data may result in false positives when legitimate but rare activities occur. Additionally, the current implementation assumes single-host deployment; distributed multi-host attack campaigns may evade detection if coordination occurs across system boundaries.

The false positive rate of 14.3% may be acceptable for high-security environments but could generate alert fatigue in large-scale deployments. Threshold tuning and expanded training data collection can reduce false alarms while maintaining high recall.

XVI. CONCLUSION AND FUTURE WORK

This paper presents a self-supervised behavioral graph framework for zero-day attack detection that combines graph autoencoders, hybrid anomaly scoring, and continual learning. By modeling system activity as heterogeneous temporal graphs and learning normal behavior patterns without labeled attack data, the system effectively detects previously unseen threats in real time.

Experimental validation demonstrates 100% recall on diverse attack scenarios including reverse shells, privilege escalation, data exfiltration, and ransomware, with detection latency suitable for production deployment (200-400ms). The system successfully detected all attacks that evaded signature-based antivirus scanning, validating the advantage of behavioral detection for zero-day threats.

The modular architecture enables independent optimization of collection, graph construction, detection, and learning components. The integration of continual learning ensures long-term effectiveness in evolving operational environments.

Future research directions include:

- **Adversarial Robustness:** Development of mechanisms to defend against adversarial evasion attacks that deliberately craft behavioral graphs to minimize detection scores
- **Distributed Detection:** Extension to multi-host environments with cross-system graph correlation for detecting distributed attack campaigns
- **Explainable AI:** Integration of graph attention mechanisms and subgraph mining to provide interpretable explanations of why specific graphs were flagged as anomalous
- **Transfer Learning:** Investigation of cross-system model transfer to enable rapid deployment on new hosts with minimal training data requirements
- **Automated Response:** Integration with automated incident response systems for immediate threat containment upon detection

The proposed framework demonstrates that practical zero-day detection is achievable through integration of graph-based behavioral modeling, self-supervised learning, and continual adaptation, providing a scalable foundation for modern cybersecurity defense.

REFERENCES

[1] Z. Dai, L. Y. Por, Y. Chen, J. Yang, C. S. Ku, R. Alizadehsani, and P. Plawiak, "An intrusion detection model to detect zero-day attacks in unseen data using machine learning," *PLoS ONE*, vol. 19, 2025.

[2] L. Guerra, T. Chapuis, G. Duc, P. Mozharovskyi, and V. Nguyen, "Self-supervised learning of graph representations for network intrusion detection," 2025.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

[3] C. Guo, X. Li, J. Cheng, S. Yang, and H. Gong, ”Continual learning for intrusion detection under evolving network threats,” *Future Internet*, vol. 17, 2025.
[4] M. K. Hayat, S. Xue, J. Wu, B. Khan, and J. Yang, ”Self-supervised time-aware heterogeneous hypergraph learning for dynamic graph-level classification,” in *WSDM*, 2025.
[5] N. Dilini, N. Sun, Y. Miao, and N. Moustafa, ”A comprehensive review on graph-based anomaly detection: Approaches for intrusion detection,” *Applied Sciences*, vol. 16, 2026.
[6] Y. Lin, Y. Lu, R. Hwang, Y. Lai, D. Sudyana, and W. Lee, ”Evolving ML-based intrusion detection: Cyber threat intelligence for dynamic model updates,” *IEEE Trans. Machine Learning in Communications and Networking*, vol. 3, 2025.
[7] S. Prasath, K. Sethi, D. Mohanty, P. Bera, and S. R. Samantaray, ”Analysis of continual learning models for intrusion detection system,” *IEEE Access*, 2025.
[8] B. Lanigan, Z. Rezaeifar, F. Cruciani, M. Milliken, J. Vincent, S. Moore, et al., ”Alert correlation for intelligent threat detection and response,” *Intelligent Systems with Applications*, 2025.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)