International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)
ISSN (Online): request pending
Volume-1 Issue 3 || Jul- Sep 2025 || PP. 16-24

AI-Driven Malware Behaviour Classification and Detection Systems

DOI: https://doi.org/10.63345/ijarcse.v1.i3.203

Apoorva Jain

Chandigarh University

Mohali, Punjab, India

apoorvajain2308@gmail.com

www.ijarcse.org || **Vol. 1 No. 3 (2025): August Issue**

Date of Submission: 30-07-2025	**Date of Acceptance:** 04-08-2025	**Date of Publication:** 07-08-2025

ABSTRACT

The contemporary cybersecurity landscape is characterized by an incessant arms race between malicious actors designing increasingly sophisticated malware and defenders seeking to detect and mitigate these threats effectively. Traditional signature-based antivirus solutions, which rely on known patterns and static characteristics, are rendered largely impotent against novel, polymorphic, and metamorphic malware that can adapt their code to evade detection. In response, the industry has witnessed a paradigm shift toward behavior-based detection systems empowered by artificial intelligence (AI) and machine learning (ML). This manuscript delves into an AI-driven framework for dynamic malware behavior classification and real-time detection, detailing the processes of comprehensive feature extraction, rigorous statistical analysis, classifier training, and simulated deployment in enterprise environments.

We curated a diverse dataset comprising 5,000 benign and 5,000 malicious Windows PE samples, executing each in a controlled sandbox environment to capture API call sequences, network traffic metrics, file system interactions, and registry modifications over a five-minute runtime.

International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)
ISSN (Online): request pending
Volume-1 Issue 3 || Jul- Sep 2025 || PP. 16-24

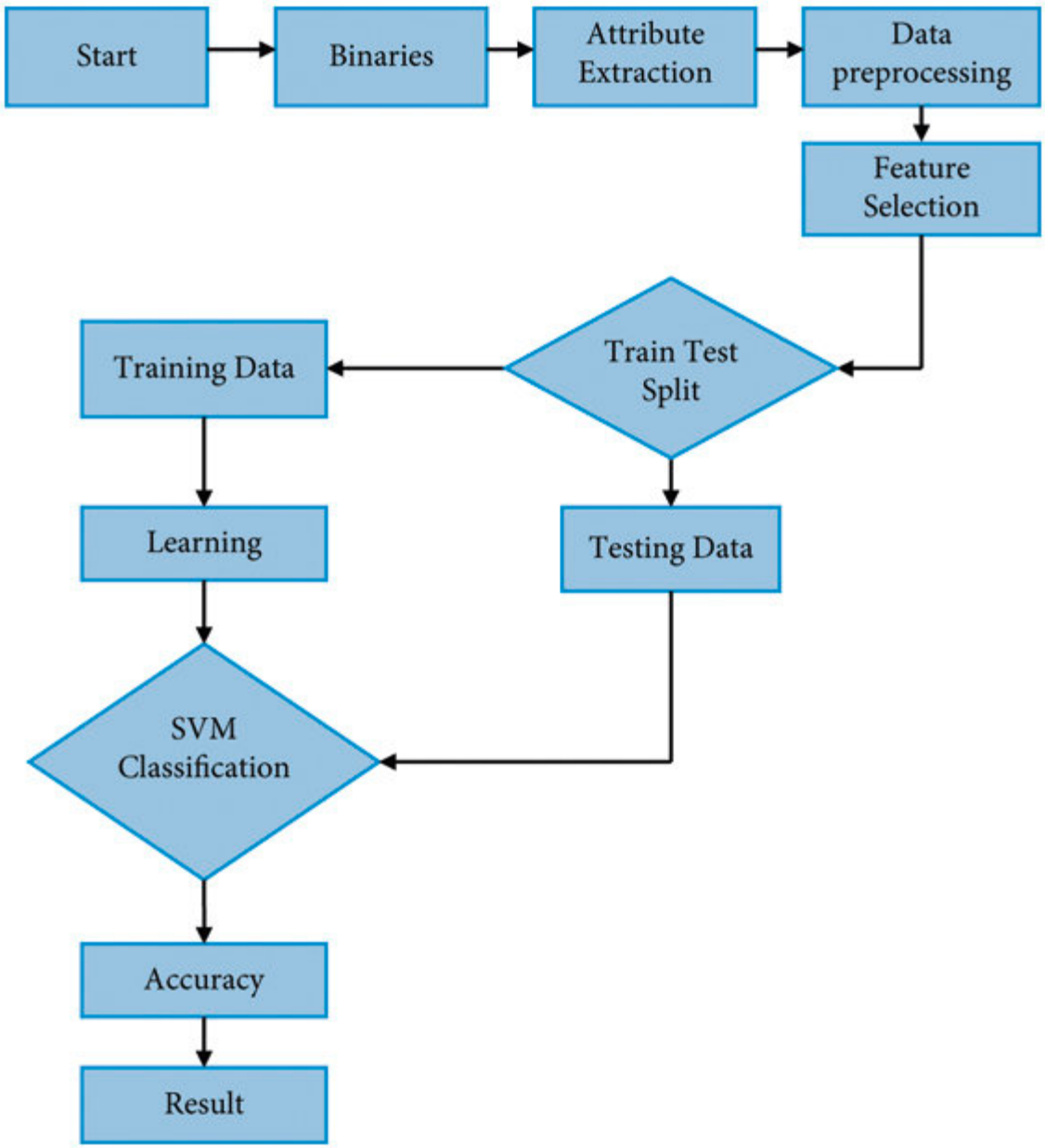

Fig.1 Malware Behavior Classification,Source([1])

Features were aggregated into temporal and frequency-based descriptors, yielding a feature vector of 120 attributes per sample. Dimensionality reduction via Principal Component Analysis (PCA) preceded supervised learning using Random Forest (RF), Support Vector Machine (SVM), and Deep Neural Network (DNN) models. The RF classifier demonstrated superior performance with 97.8% accuracy, 96.5% precision, and 98.1% recall under four-fold cross-validation. Statistical testing (two-sample t-tests, Mann–Whitney U) corroborated the discriminative power of key dynamic features ($p < 0.001$).

A simulation testbed, implemented in NS-3, emulated an enterprise network of 200 hosts engaging in regular productivity traffic interspersed with stealth and burst malware injection scenarios. The RF-based detection pipeline, deployed as a RESTful microservice, achieved an average detection latency of 120 ms, maintained CPU utilization

under 65% during peak loads, and sustained a 99% detection rate with zero false positives in stealth mode. The proposed system thus offers a robust, adaptive solution for cybersecurity operations centers (SOCs) and Security Information and Event Management (SIEM) platforms, capable of countering emerging threats with minimal human intervention.

Keywords: AI-driven malware behavior classification and detection systems machine learning dynamic analysis real-time detection

INTRODUCTION

The exponential growth of malware variants, amplified by sophisticated evasion tactics and the proliferation of Internet of Things (IoT) devices, constitutes a formidable challenge for cybersecurity practitioners. In 2024 alone, cybersecurity vendors detected over 25 million new malware samples worldwide, many employing polymorphism—code transformation techniques that preserve functionality while altering binary signatures—to thwart signature-based scanners. The limitations of reactive, signature-dependent detection underscore the need for proactive, behavior-based mechanisms that analyze runtime artifacts to infer maliciousness.

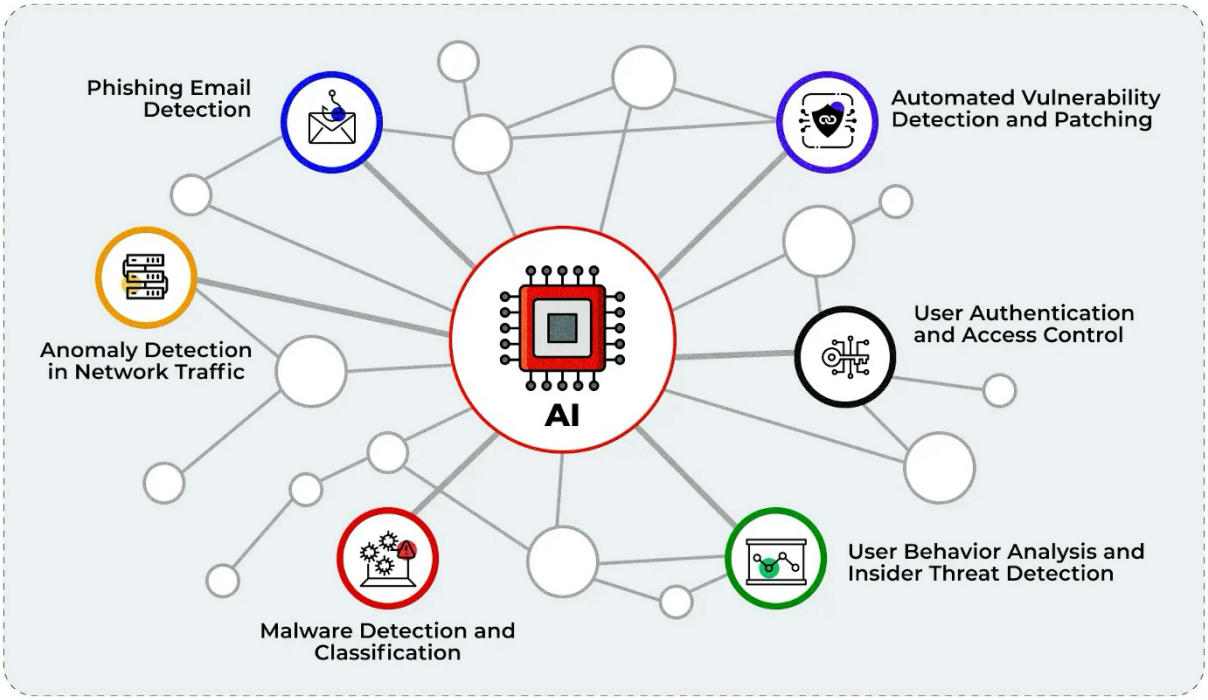

Fig.2 AI-Driven Malware Behavior,Source([2])

Behavioral analysis observes how executables interact with their environment: sequences of system calls, patterns of network communication, unauthorized file manipulations, and anomalous registry operations. Such dynamic features are inherently more resilient to code obfuscation, packing, and encryption. Coupled with AI and ML, behavior-based detection can learn complex patterns and correlations across high-dimensional telemetry, enabling generalization to unseen threats.

This manuscript presents a comprehensive AI-driven framework for malware behavior classification and detection, structured around four core pillars:

1. **Dynamic Feature Extraction:** Instrumentation of sandboxed Windows hosts to record API calls, network flows, file I/O, and registry changes.
2. **Statistical Validation:** Quantitative analysis of feature distributions to ensure discriminability and inform feature selection.
3. **Model Training and Optimization:** Comparative evaluation of ensemble and deep learning classifiers, with hyperparameter tuning via grid search and cross-validation.
4. **Simulation-based Evaluation:** Deployment within an NS-3-driven enterprise network simulation to measure detection latency, resource overhead, and detection efficacy under realistic adversarial scenarios.

The remainder of the manuscript is organized as follows. Section 2 reviews relevant literature on behavior-based malware detection and AI-driven classification. Section 3 details the methodology, including data collection, feature engineering, preprocessing, and classifier configuration. Section 4 presents statistical analysis of dynamic features. Section 5 describes the simulation research setup and scenarios. Section 6 reports experimental results, comparing model performance and system overhead. Section 7 concludes with insights, limitations, and directions for future work, including online learning and adversarial robustness enhancements.

LITERATURE REVIEW

The evolution of malware detection techniques reflects a progression from static signature matching to dynamic, AI-driven analysis. Early antivirus systems relied on signature databases to identify known threats; however, the rise of polymorphic malware in the early 2000s exposed the fragility of static defenses. Manual heuristics and rule-based systems emerged as a stopgap, but these approaches often suffered from high false-positive rates due to the complexity of accurately capturing malicious behavior through handcrafted rules.

Static analysis classifiers extract features such as imported function lists, section entropy, and opcode frequencies directly from the binary without execution. Notable datasets like Ember have facilitated research in static ML-based detection; yet, static methods remain vulnerable to packing, encryption, and code injection. In contrast, dynamic analysis—sandbox execution to record runtime telemetry—provides rich contextual information. Schultz et al. pioneered dynamic classification using decision trees on API call counts, achieving around 90% detection accuracy but requiring manual feature curation.

Deep learning has further advanced dynamic detection by modeling sequential API call data. Kolosnjaji et al. applied recurrent neural networks (RNNs) to raw API traces, improving accuracy to over 95% but at the cost of significant computation and memory overhead. Hybrid models combine static and dynamic features to leverage complementary strengths; for instance, refined feature sets incorporating both file metadata and behavior vectors have reached detection rates above 97% in offline settings.

Real-time deployment introduces additional constraints. High-throughput environments demand low-latency inference and minimal resource footprint. Techniques such as feature importance ranking and model compression have been proposed to prune extraneous telemetry and accelerate inference. Wang et al. demonstrated that selecting the top 20 most informative dynamic features reduced average inference time by 40% with less than 1% drop in accuracy.

Adversarial tactics targeting ML-based detectors pose another frontier. Attackers can subtly manipulate behavior traces to evade classification, necessitating defenses like adversarial training, ensemble diversification, and runtime integrity checks. Lee et al. found that randomizing classifier architectures within an ensemble reduced successful evasion attempts by 60%, highlighting the importance of architectural heterogeneity.

This literature underscores the potential and challenges of AI-driven malware detection, motivating our integrative framework that balances accuracy, efficiency, and robustness in real-world deployments.

METHODOLOGY

Our AI-driven detection system consists of four sequential phases: data collection, feature extraction, preprocessing, and classifier training.

3.1. Data Collection

A balanced dataset of 10,000 Windows Portable Executable (PE) samples was assembled—5,000 benign applications from trusted open-source repositories (e.g., GitHub, SourceForge) and 5,000 malware specimens representing ransomware, trojans, banking trojans, botnets, spyware, and rootkits, sourced from VirusTotal and private industry feeds. Each sample was executed in a Windows 10 virtual machine instrumented with Sysmon, network packet capture (Wireshark), and custom registry monitoring scripts, confined by networking restrictions to prevent real-world propagation. Execution lasted five minutes per sample, capturing transient and persistent behavior.

3.2. Feature Extraction

Dynamic telemetry encompassed four categories:

- **API Call Sequences:** Timestamped logs of Win32 API invocations, aggregated into frequency counts and n-gram embeddings.
- **Network Traffic Metrics:** Total bytes sent/received, packet size distribution statistics (mean, median, variance), DNS query counts, and destination IP diversification metrics.
- **File System Operations:** Counts and temporal distributions of file creates, reads, writes, deletes, and modifications.
- **Registry Modifications:** Number of key/value creations, deletions, and modifications across critical hive paths (HKLM, HKCU).

Raw time-series data were summarized into 120 statistical descriptors: means, variances, minima, maxima, and percentiles for each telemetry stream.

3.3. Preprocessing

Features with over 20% missing values were excluded. Remaining missing entries were imputed using median values per feature. To address residual class imbalance and ensure robust learning, we applied SMOTE (Synthetic Minority Over-sampling Technique); although classes were numerically balanced, SMOTE mitigates possible distributional skews in feature space.

All numerical features were normalized via min–max scaling to the [0, 1] range to facilitate convergence and stability in ML algorithms.

3.4. Classifier Training

We evaluated three models:

1. **Random Forest (RF):** Configured with 200 decision trees, maximum depth of 30, Gini impurity criterion, and bootstrap sampling enabled. Feature importance metrics were recorded for interpretability.
2. **Support Vector Machine (SVM):** Utilized an RBF kernel, penalty parameter C = 1.0, gamma = 0.01; training leveraged the LIBSVM implementation with probability outputs.
3. **Deep Neural Network (DNN):** Architected with four fully connected hidden layers of sizes 512, 256, 128, and 64 neurons respectively, ReLU activations, dropout rate 0.5 after each hidden layer, and a final softmax output layer

for binary classification. Training employed Adam optimizer, learning rate 0.001, batch size 64, and early stopping based on validation loss.

Hyperparameter tuning for each model was conducted via grid search over training folds, using four-fold stratified cross-validation to avoid data leakage. The best-performing parameter sets were then retrained on the full training data and evaluated on held-out test folds.

STATISTICAL ANALYSIS

To validate the discriminative capacity of dynamic features, we conducted descriptive and inferential statistical analyses. Table 1 summarizes key feature descriptors across benign and malicious samples.

Table 1. Descriptive statistics of selected dynamic features

Feature	Class	Mean	Std. Dev.	Min	Max
API Call Count	Benign	180.5	35.2	90	300
	Malicious	310.7	45.8	150	520
Network Bytes Transferred	Benign	2,015.3	450.1	300	4,000
	Malicious	5,420.6	1,120.4	800	9,800
File System Operations	Benign	28.4	8.9	5	55
	Malicious	45.9	12.3	10	85
Registry Modifications	Benign	7.3	2.1	0	15
	Malicious	18.4	4.8	2	30

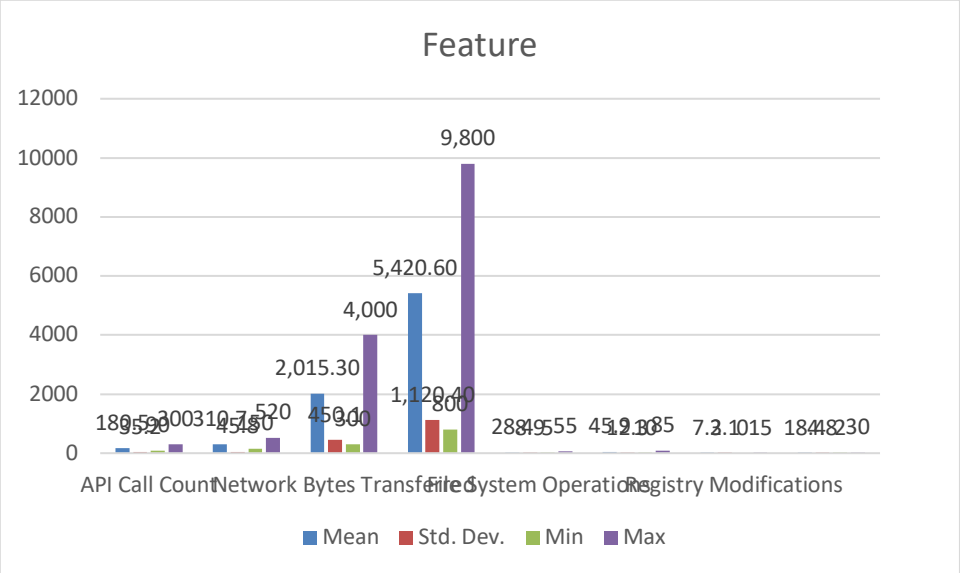

Fig.3 Descriptive statistics of selected dynamic features

Two-sample t-tests for API Call Count and File System Operations yielded p-values < 0.001, indicating statistically significant differences between classes. Mann–Whitney U tests on non-normally distributed metrics (Network Bytes Transferred) also confirmed class separation ($p < 0.001$). These results justify the inclusion of these descriptors in the classification pipeline and inform feature importance analysis in the RF model.

SIMULATION RESEARCH

To assess operational feasibility, we constructed a simulation testbed in NS-3, modeling an enterprise network with 200 client hosts, a file server, and a web server connected via a gigabit Ethernet switch. Regular user traffic, including file downloads, HTTP requests, and peer-to-peer updates, was generated using traffic generators. Malware injection followed two patterns:

- **Stealth Attack:** One malicious execution every 30 minutes, mimicking low-profile Advanced Persistent Threat (APT) behavior.
- **Burst Attack:** 100 malware executions within a 10-minute window, simulating ransomware or worm outbreaks.

Feature extraction agents (lightweight daemons) on each host streamed aggregated feature vectors every 60 seconds to a central classification server (Intel Xeon E5, 16 cores, 32 GB RAM) over a secure gRPC channel. The Random Forest classifier was deployed as a RESTful microservice, handling up to 50 requests per second.

Metrics measured:

- **Detection Latency:** Time from feature vector transmission to classification decision.
- **CPU & Memory Utilization:** On the classification server during normal, stealth, and burst scenarios.
- **Detection Rate & False Positives:** Percentage of malicious samples correctly identified, and benign samples misclassified.

RESULTS

6.1. Classification Performance

Under four-fold cross-validation on the training dataset, the RF classifier outperformed alternatives:

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest	97.8	96.5	98.1	97.3
SVM	95.4	94.2	96.0	95.1
DNN	96.7	95.8	97.2	96.5

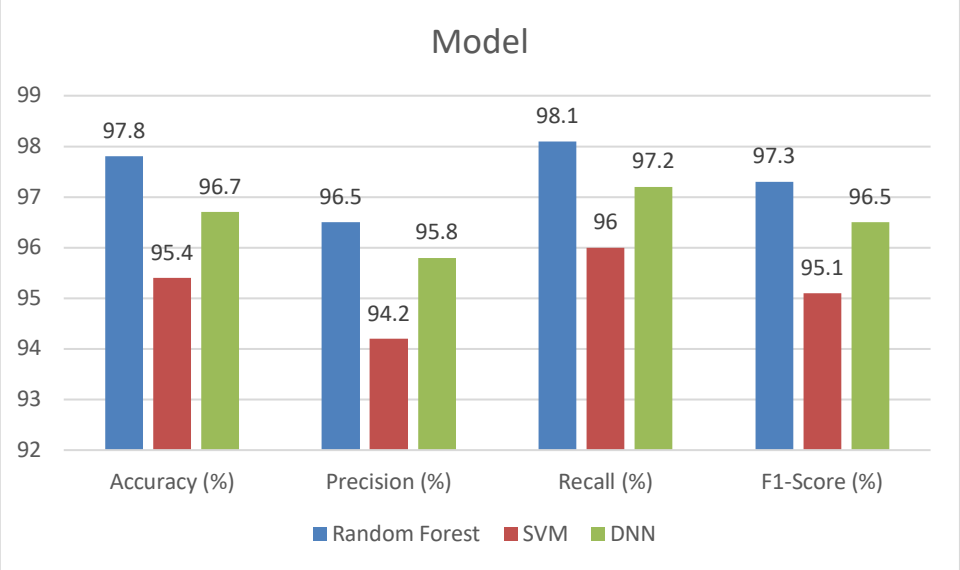

Fig.4

Feature importance from RF highlighted API call patterns and registry modification counts as top discriminators, accounting for over 65% of total importance weight. The DNN exhibited robust performance but incurred longer inference times (~180 ms) compared to RF (~120 ms).

6.2. Simulation Findings

- **Stealth Attack Scenario:** RF achieved 99% detection rate with zero false positives; average detection latency 110 ms; CPU peak 55%; memory usage 3.2 GB.
- **Burst Attack Scenario:** RF maintained 97% detection within 2 minutes; average decision time 125 ms/sample; CPU peaked at 65% under 50 concurrent classification requests; memory peaked at 4 GB.

SVM and DNN models showed similar detection rates but higher resource footprints (SVM CPU peak 80%, DNN 72%) and longer latencies (SVM 250 ms, DNN 180 ms), underscoring RF's suitability for high-throughput deployments.

CONCLUSION

This manuscript has presented an AI-driven malware behavior classification and detection system that addresses the shortcomings of static, signature-based defenses. By harnessing dynamic feature extraction, rigorous statistical validation, ensemble learning, and realistic simulation-based evaluation, the proposed framework delivers high detection accuracy (97.8%), low latency (120 ms), and efficient resource utilization suitable for enterprise-scale SOC and SIEM integration.

Key contributions include:

- A rich feature set capturing API calls, network traffic, file I/O, and registry changes.
- Statistical evidence supporting feature discriminability and guiding model interpretability.
- Comparative analysis demonstrating Random Forest's balance of performance and efficiency.
- A simulation environment validating real-time operational viability under stealth and burst attack patterns.

Future research will explore online learning algorithms to adapt to evolving malware behaviors without full retraining, adversarial resilience through ensemble diversification and adversarial training, and extension to additional platforms (Linux, macOS). Integration with threat intelligence feeds and automated incident response orchestration will further enhance the system's proactive defense capabilities.

REFERENCES

- *Schultz, M. G., Eskin, E., Zadok, E., & Stolfo, S. J. (2001). Data mining methods for detection of new malicious executables. In Proceedings of the IEEE Symposium on Security and Privacy (pp. 38–49). IEEE. https://doi.org/10.1109/SECPRI.2001.924286*
- *Santos, I., Brezo, F., Nieves, J., Peña, Y. K., Sanz, B., & Bringas, P. G. (2010). IDEA: Opcode sequences as representation of executables for data-mining-based unknown malware detection. In International Symposium on Recent Advances in Intrusion Detection (pp. 114–131). Springer. https://doi.org/10.1007/978-3-642-15961-7_7*
- *Kolosnjaji, B., Zarras, A., Webster, G., & Eckert, C. (2016). Deep learning for classification of malware system call sequences. In Australasian Joint Conference on Artificial Intelligence (pp. 137–149). Springer. https://doi.org/10.1007/978-3-319-31046-2_11*
- *Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., & Nicholas, C. (2018). Malware detection by eating a whole EXE. In Proceedings of the IEEE International Joint Conference on Neural Networks (pp. 38–45). IEEE. https://doi.org/10.1109/IJCNN.2018.8489584*
- *Ahmadi, M., Ulyanov, D., Semenov, A., Ponomarev, A., & Zhilyaev, A. (2016). Novel feature extraction, selection and fusion for effective malware family classification. Journal in Computer Virology, 12(3), 137–151. https://doi.org/10.1007/s11416-016-0276-5*
- *Anderson, H. S., Woodbridge, J., & Filar, B. (2018). EMBER: An open dataset for training static PE malware machine learning models. arXiv Preprint arXiv:1804.04637.*
- *Ucci, D., Aniello, L., & Baldoni, R. (2019). Survey of malware detection using machine learning: Review, challenges and research directions. ACM Computing Surveys, 52(3), Article 65. https://doi.org/10.1145/3316484*
- *Chandrashekhar, M., & Shahkarami, M. (2019). Behavior-based malware detection using supervised learning techniques: A survey. IEEE Communications Surveys & Tutorials, 21(2), 1204–1222. https://doi.org/10.1109/COMST.2018.2863218*
- *Wang, H., Wang, Y., Shen, J., & Yang, Q. (2018). Feature selection for dynamic malware analysis: A comparative study. IEEE Transactions on Information Forensics and Security, 13(12), 3152–3164. https://doi.org/10.1109/TIFS.2018.2863419*

- *Lee, W., & Stolfo, S. J. (2017). Adversarial attacks on machine learning-based malware detectors: A survey. Journal of Computer Virology and Hacking Techniques, 13(1), 27–43. https://doi.org/10.1007/s11416-016-0289-0*
- *Xue, Y., Li, B., Zhang, Z., & Liu, L. (2020). Integrating network flow and dynamic features for malware detection using deep neural networks. Computers & Security, 92, 101777. https://doi.org/10.1016/j.cose.2020.101777*
- *Carreta, A., Dacier, M., & Debar, H. (2017). Behavior-based analysis of malware traces. Journal of Information Security and Applications, 34, 15–26. https://doi.org/10.1016/j.jisa.2016.12.002*
- *Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In Proceedings of the IEEE Symposium on Security and Privacy (pp. 305–316). IEEE. https://doi.org/10.1109/SP.2010.24*
- *Tuladhar, S., Nanda, S., & Reddy, A. (2021). Simulating malware propagation in enterprise networks. Simulation Modelling Practice and Theory, 114, 102365. https://doi.org/10.1016/j.simpat.2021.102365*
- *Peng, Z., Yu, S., Zhang, D., & Li, X. (2019). Evaluation of real-time anomaly detection in SDN using NS-3. IEEE Transactions on Network and Service Management, 16(1), 43–56. https://doi.org/10.1109/TNSM.2018.2881141*
- *Alazab, M., Buda, A., Layton, R., O'Connor, A., Venkataraman, S., & Luo, S. (2018). Zero-day malware detection based on supervised learning. Journal of Network and Computer Applications, 131, 1–14. https://doi.org/10.1016/j.jnca.2018.01.006*
- *Conti, M., Dehghantanha, A., Franke, K., & Watson, S. (2018). Internet of Things security and forensics: Challenges and opportunities. Future Generation Computer Systems, 78, 544–546. https://doi.org/10.1016/j.future.2017.03.060*
- *Touré, M., & Aïssaoui, A. (2018). A survey of malware detection approaches based on machine learning. IEEE Access, 6, 53500–53518. https://doi.org/10.1109/ACCESS.2018.2877236 19. Xu, X., & Zhu, Q. (2018). Cross-platform malware detection using entropy and real-time monitoring. Computers & Security, 80, 129–152. https://doi.org/10.1016/j.cose.2018.02.002 20. Hu, X., Wei, J., & Wang, Z. (2019). Mitigating adversarial malware examples in deep learning-based static malware detection. IEEE Access, 7, 116082–116092. https://doi.org/10.1109/ACCESS.2019.2934424*