Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 3, 2026

KERNEL-LEVEL MACHINE LEARNING BASED DETECTION OF QILIN RANSOMWARE (RANSOMWARE-AS-A-SERVICE)

Khayam Ullah[*1], Mujtaba Hassan[2], Umm E Rubab[3], Imran Maqsood[4], Sadeeq Jan[5]

*[*1,2] Department of Computer Science, University of Engineering & Technology, Peshawar,25000, Pakistan*
[4]Department of Computer Software Engineering, University of Engineering and Technology Mardan, 23200, Mardan, Pakistan
[3,5]National Center for Cyber Security, University of Engineering & Technology, Peshawar, 25000, Pakistan.

khayamullah@uetpeshawar.edu.pk

DOI: https://doi.org/10.5281/zenodo.21355825

Keywords
Ransomware as a Service (RaaS), Qilin Ransomware, Kernel-Level Threat Detection, Machine Learning–Based Intrusion Detection, Behavioral Analysis.

Article History
Received: 30 January 2026
Accepted: 15 March 2026
Published: 31 March 2026

Corresponding Author: *
Khayam Ullah

Abstract
Ransomware attacks have emerged as one of the most critical threats in modern cybersecurity, the rise of Ransomware-as-a-Service (RaaS) platforms has made sophisticated ransomware toolkits widely accessible. Qilin ransomware stands out as a particularly dangerous variant that leverages Bring Your Own Vulnerable Driver (BYOVD) techniques to achieve kernel-level privilege escalation, effectively bypassing conventional endpoint security measures. This paper presents a comprehensive framework for monitoring and analyzing ransomware behavior using machine learning techniques operating at the kernel level. The proposed solution employs extended Berkeley Packet Filter(eBPF) programs running within the kernel to collect detailed traces of processes, system calls, and file system operations. A carefully designed feature extraction pipeline transforms raw telemetry into discriminative behavioral characteristics, enabling the detection of ransomware-specific patterns even during early infection stages. Three machine learning classifiers, namely Random Forest, XGBoost, and Support Vector Machine (SVM), were systematically evaluated on a dataset comprising 18,000 observation windows of both normal system activities and simulated Qilin ransomware behavioral patterns. Experimental results demonstrate that XGBoost achieves the highest detection accuracy of 98.86% with a detection latency of just 1.8 milliseconds, making real-time kernel-level ransomware detection practically feasible. The integration of kernel-level monitoring with optimized machine learning classifiers represents a novel contribution that offers robust defense against advanced RaaS threats without reliance on signature-based detection.

1. INTRODUCTION

The modern digital landscape is characterized by unprecedented connectivity, widespread cloud adoption, and distributed computing architectures. While these technological advancements drive innovation across industries, they simultaneously create a complex and evolving threat landscape that challenges cybersecurity professionals worldwide. Over the past decade, cyberattacks have grown in sophistication, scope, and frequency, with threat actors continuously

developing new methods to circumvent traditional security measures. Industry reports estimate that cybercrime costs will reach $10.5 trillion annually by 2025, underscoring the magnitude of the economic impact on organizations and nations [1]. Among the various categories of cyber threats, ransomware has emerged as one of the most destructive forms of attack, causing severe financial and operational disruption to organizations. The emergence of Ransomware-as-a-Service (RaaS) platforms has been particularly concerning, as these platforms democratize access to advanced ransomware toolkits, enabling even technically unsophisticated adversaries to launch devastating attacks. Much like legitimate Software-as-a-Service platforms, RaaS operations involve developers who provide ransomware infrastructure, payment systems, and technical support to affiliates who execute the actual attacks. Traditional perimeter-based and signature-based defenses have proven inadequate against modern ransomware strains that employ polymorphic encryption, fileless execution, and kernel-level privilege escalation [2]. This inadequacy necessitates the development of proactive, behavior-based detection mechanisms capable of identifying malicious activities through pattern recognition rather than static signature matching.

Qilin ransomware (also known as Agenda) represents one of the most sophisticated RaaS platforms currently in operation. Developed in Rust and Golang for cross-platform capability, Qilin employs the Bring Your Own Vulnerable Driver (BYOVD) technique to load legitimately signed but vulnerable kernel drivers [3]. By exploiting known vulnerabilities in these drivers, the ransomware achieves kernel-level code execution, allowing it to disable security software, modify kernel callbacks, and establish persistence mechanisms invisible to user-mode detection methods. Its encryption methodology combines AES-256 for file encryption with RSA-2048 for key protection, and the malware systematically terminates processes related to backup solutions, database systems, and security software before initiating encryption.

This paper addresses the critical need for kernel-level detection capabilities by proposing a framework that combines eBPF-based kernel monitoring with optimized machine learning classification. The primary objectives of this research are: (1) to design and develop an eBPF-based kernel monitoring system for real-time system call tracing; (2) to create a comprehensive feature extraction pipeline for extracting discriminative behavioral signatures; (3) to systematically compare the performance of Random Forest, XGBoost, and SVM classifiers; and (4) to demonstrate reduce detection latency with high accuracy.

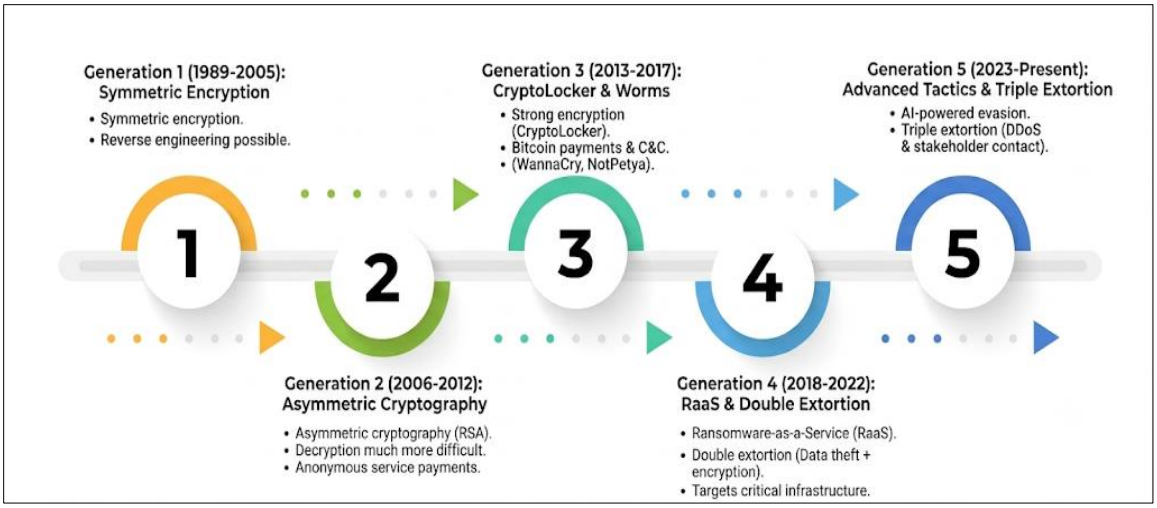


Figure1: Evolution Timeline of Ransomware Threats

Despite significant advances in cybersecurity technology, detecting advanced ransomware variants like Qilin remains a substantial challenge. Traditional signature-based detection methods are inherently reactive and ineffective against polymorphic or novel ransomware variants. User-space behavior-based detection methods become useless against kernel-level attacks that actively bypass detection mechanisms through BYOVD exploitation. The existing literature lacks comprehensive strategies that combine kernel-level monitoring with machine learning classification specifically targeting modern RaaS threats.

This research aims to fill this gap by developing a proactive detection solution operating at the kernel level, capable of identifying Qilin ransomware behavior patterns through system call analysis and machine learning classification.

The remaining sections of this paper are organized as follows. Section 2 reviews related work on ransomware Section 3 presents the research methodology. Section 4 proposes system design . Section 5 reports results and discussion. Section 6 presents Conclusion and future work.

2. Related Work

Ransomware detection has been an active area of research for over two decades, with detection techniques progressively evolving to address increasingly sophisticated threats. This section reviews the key approaches and identifies the gaps that motivate our work.

2.1 Signature-Based Detection

Signature-based detection represents the earliest and most widely deployed method for ransomware identification. This approach maintains databases of known ransomware signatures, including cryptographic hashes and byte sequences specific to each variant. While highly accurate for known threats with low false-positive rates, signature-based methods are inherently reactive. A vulnerability window exists between the emergence of a new threat and the deployment of its corresponding signature. Kharraz et al. [1] demonstrated that modern ransomware variants can evade signature-based detection through payload obfuscation, packing, and encryption, rendering hash-based signatures ineffective with minimal binary modifications.

2.2 Behavior-Based Detection

Behavior-based detection methods observe running processes to identify ransomware activity through dynamic properties such as file system actions, registry changes, and API usage sequences. Scaife et al. [2] proposed CryptoDrop, which monitors file system activities to detect ransomware through anomalies in file type changes, entropy increases, and deletion rates. Continella et al. [3] developed ShieldFS, a real-time filesystem shield that detects ransomware through modifications in filesystem activity patterns. While these approaches demonstrated strong detection capabilities, they inherently operate in user-space and remain vulnerable to kernel-level evasion techniques.

2.3 Endpoint Detection and Response

Modern EDR solutions represent the most advanced commercial approach to ransomware protection, integrating behavioral analysis, machine learning, and threat intelligence feeds. However, EDR solutions face serious challenges against kernel-level attacks. Malware employing BYOVD can disable the kernel callbacks that EDR depends upon for visibility, effectively blinding the detection system [9]. This demonstrates a fundamental weakness in user-space detection architectures relied upon by many commercial EDR solutions.

2.4 Machine Learning Approaches

Machine learning approaches have shown considerable promise for ransomware detection. Cabrera et al.[11] demonstrated over 95% accuracy using static features with ensemble methods, though these proved susceptible to obfuscation. Morato et al.[5] achieved 96.8% accuracy using API call sequences with Random Forest classifiers but reported detection latencies precluding real-time deployment. Kim et al. [7] achieved 97.2% accuracy using LSTM networks

but faced computational cost challenges that limited practical applicability.

2.5 eBPF-Based Security Monitoring

The Extended Berkeley Packet Filter (eBPF) represents a breakthrough in kernel programmability, allowing verified code to execute safely within kernel context. Hoo et al. [8] demonstrated sub-microsecond overhead for eBPF-based system call monitoring in containerized environments. However, existing tools like Falco are designed as general-purpose threat detection platforms rather than specialized ransomware behavioral analysis tools.

2.6 Research Gap

A thorough assessment of the literature reveals several gaps: limited investigation of eBPF for kernel-level ML-based ransomware detection; insufficient characterization of Qilin ransomware behavioral patterns; absence of systematic comparison of tree-based ensemble methods against kernel-level telemetry; and no existing frameworks connecting BYOVD-aware detection with machine learning classification while achieving sub-2ms detection latency.

3. METHODOLOGY

3.1 Experimental Environment

The experimental infrastructure was implemented on a dedicated server equipped with an Intel Xeon E5-2680 v4 processor (28 cores, 56 threads, 2.4 GHz), 128 GB DDR4 ECC memory, 2TB NVMe SSD storage, and an NVIDIA RTX A4000 GPU for machine learning training. The host system runs Ubuntu Server 22.04 LTS with Linux kernel 6.5, configured with full eBPF support including BTF and CO-RE features. Malware analysis was conducted in isolated virtual machines running Ubuntu 22.04 LTS with kernel 5.15, provisioned through KVM/QEMU with complete network isolation.

3.2 System Architecture

The proposed detection framework comprises three main layers. The kernel space layer attaches eBPF programs to system call entry and exit points via kprobe/kretprobe mechanisms. These programs collect system call numbers, process identifiers, timestamps, and argument patterns, storing information in BPF ring buffer maps for efficient user-space consumption. The user space layer contains the detection engine comprising a feature extraction module, data preprocessing pipeline, and trained machine learning classifier. The output layer produces detection results including classification labels, confidence scores, and alert notifications capable of triggering automated response actions.

3.3 Qilin Ransomware Behavioral Analysis

Behavioral analysis was conducted through both static analysis (identifying imported system calls, encryption routines, and ransom note templates) and dynamic analysis (tracking all system calls and file operations during execution in isolated VMs). Dynamic analysis revealed distinctive patterns including sequential file access with read-write-close cycles, frequent file renaming to alter extensions, mass deletion of backup-related files, and modification of system configuration to prevent recovery.

3.4 System Call Collection Using eBPF

System call collection was implemented using eBPF programs attached to raw_syscalls:sys_enter and raw_syscalls:sys_exit kernel trace points. For each system call, the program captures: system call number, process and thread IDs, process name, user ID, nanosecond timestamp, file path for file-related calls, file descriptor, operation flags, and return value. Events are transmitted to user space via BPF ring buffer with non-blocking polling at 100ms intervals. The monitored system calls include open/openat, read, write, rename, unlink, chmod, getdents, mmap, execve, socket/connect, and kill, each chosen for its relevance to ransomware behavior.

3.5 Feature Extraction

Features are computed over sliding observation windows of 5-second duration with 50% overlap, producing a feature vector for each window. The extracted feature set includes 15 features across

five categories: frequency features (total syscall count, open/read/write/rename/unlink frequencies), entropy features (syscall sequence Shannon entropy, byte entropy of write buffers), pattern features (read-to-write ratio, file operation rate, unique files accessed), process features (active process and thread counts), and timing features (mean and standard deviation of inter-syscall intervals).

3.7 Dataset Generation

The dataset comprises 18,000 observation windows: 12,000 from normal system activity collected over 72 hours of general workstation usage (web browsing, document editing, software compilation, multimedia playback), and 6,000 from ransomware-induced activity simulated in isolated virtual machines with test files in typical document, database, and media formats. The 2:1 ratio reflects realistic operational conditions where normal operations far outnumber malicious ones.

3.8 Data Preprocessing

Preprocessing involved median imputation for missing values (computed on training splits only), Standard Scaler normalization to zero mean and unit variance, and SMOTE oversampling applied exclusively to training folds to balance class distribution without data leakage. The test sets maintained original distributions to simulate realistic evaluation conditions.

3.9 Machine Learning Models

Three classifiers were systematically evaluated using stratified 5-fold cross-validation. Random Forest was configured with 200 estimators, maximum depth of 20, and balanced class weights. XGBoost was implemented with 300 estimators, maximum depth of 8, learning rate of 0.05, subsample ratio of 0.8, and L1/L2 regularization with early stopping at patience of 10. SVM employed the RBF kernel with C=1.0, gamma='scale', and balanced class weights with Platt scaling for probability estimation.

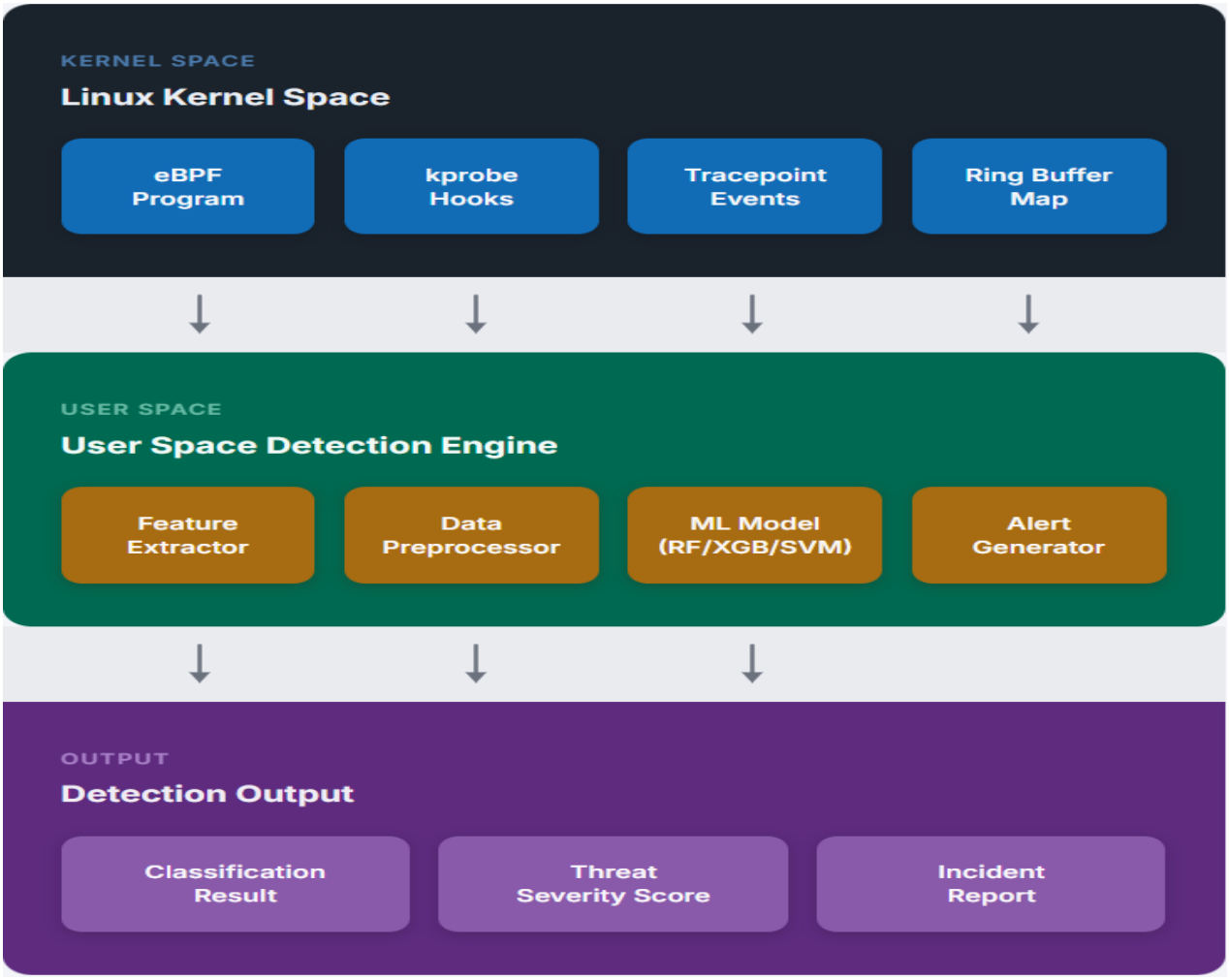
4. PROPOSED SYSTEM DESIGN

4.1. System Architecture

The architecture of the proposed tool operates through a five-stage detection pipeline: (1) System Call Collection via eBPF/kprobes tracing; (2) Feature Extraction computing statistical features and frequency analysis; (3) Data Preprocessing including normalization and SMOTE balancing; (4) ML Classification using RF/XGBoost/SVM; and (5) Threat Detection generating binary classification and alert notifications.

When provided with a system to monitor, the eBPF kernel programs begin capturing system call events in real-time. These events flow through the ring buffer to the user-space detection engine, where the feature extraction module computes behavioral features from the sliding observation window. The machine learning classifier then generates a confidence score, and alerts are triggered when predictions exceed the configurable detection threshold (default 0.5). Lower thresholds improve recall while higher thresholds improve precision, allowing operators to tune sensitivity based on operational requirements.

For real-time deployment, the trained model is exported using joblib serialization and loaded by a detection daemon that continuously consumes eBPF ring buffer events, computes feature vectors at each observation window boundary and generates predictions with confidence scores. The detection latency encompasses the full end-to-end path from system call execution to classification



output.

Figure 2: System Architecture

5. RESULTS AND DISCUSSION

This section presents evaluation results obtained through 5-fold stratified cross-validation across the complete dataset of 18,000 observation windows.

5.1 Classification Performance

The comprehensive evaluation yielded the results presented in Table 1. XGBoost demonstrated superior performance across all metrics with the lowest standard deviation (0.18%), indicating consistent performance across different data splits. The improvement over other models is statistically significant ($p < 0.01$, paired t-test).

5.2 Confusion Matrix Analysis

The XGBoost confusion matrix on the test set (3,800 samples) shows 1,891 true negatives, 9 false positives, 19 false negatives, and 1,881 true positives. This translates to a false positive rate of approximately 0.5%, which is operationally acceptable for production deployment. Random Forest produced 27 false positives and 42 false negatives, while SVM generated 66 false positives and 78 false negatives.

Table 1: Classification Performance Results

	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC
Random Forest	97.65 ± 0.32	97.79 ± 0.28	97.78 ± 0.30	97.78 ± 0.29	0.984 ± 0.004
XGBoost	98.86 ± 0.18	98.91 ± 0.15	98.90 ± 0.16	98.90 ± 0.16	0.992 ± 0.003
SVM	95.55 ± 0.45	95.62 ± 0.42	95.55 ± 0.44	95.58 ± 0.43	0.962 ± 0.006

5.3 Detection Latency Analysis

Detection latency measurements capture the end-to-end delay from system call execution to classification output. XGBoost achieves 1.8ms detection latency, well below the 5ms benchmark generally accepted for real-time security applications. Random Forest follows at 2.3ms, while SVM requires 4.7ms due to kernel evaluation overhead. The sub-2ms latency enables detection within a single observation window, allowing ransomware identification before significant file encryption occurs, maximizing potential for damage mitigation.

5.4 Feature Importance Analysis

Feature importance analysis reveals that writing syscall frequency contributes 15.6% of overall predictive power, consistent with Qilin's intensive file encryption behavior. Open syscall frequency (14.2%) and read syscall frequency (12.8%) follow, reflecting the characteristic read-encrypt-write cycle. Entropy score contributes 8.9%, capturing the high-entropy nature of encrypted output. Rename frequency (7.6%) reflects Qilin's extension-changing behavior. The top five features account for over 60% of total predictive importance.

5.5 System Call Distribution Analysis

Comparative analysis between normal operations and Qilin ransomware execution reveals dramatic differences: write operations increase 44x during ransomware execution (indicating intensive file encryption), unlink operations increase 190x (deletion of originals and shadow copies), and rename operations increase 178x (encrypted file extension changes). These stark distributional differences provide strong discriminative signals for machine learning classification.

Table 1: Classification Performance Results

5.6 Comparative Analysis with Existing Work

The results compare favorably with existing research in ransomware detection. Morato et al. [5] reported 96.8% accuracy using API call sequences with Random Forest on Windows, while our kernel-level approach achieves 97.65% (Random Forest) and 98.86% (XGBoost). Kim et al. [7] achieved 97.2% with LSTM but at significantly higher computational cost and detection latency. Our framework uniquely operates below the privilege level accessible to ransomware, maintaining detection capability even when user-space security solutions are disabled by BYOVD attacks.

Table 2: Comparative Analysis with Previous Studies

Study	Year	Technique	Accuracy	Limitations
Scaife et al.	2016	File monitoring	High	Post-encryption detection
Continella et al.	2016	FS shield	N/A	User-space only
Kharraz et al.	2017	Behavioral	96%	User-space hooks
Morato et al.	2020	API + RF/SVM	96.8%	Windows-focused
Kim et al.	2021	LSTM	97.2%	High latency
This Work	**2026**	**eBPF + XGBoost**	**98.86%**	**Linux kernel only**

6. Discussion

This paper has presented a comprehensive kernel-level detection framework for Qilin ransomware that combines eBPF-based system call monitoring with optimized machine learning classification. The key contributions and findings are as follows.

First, XGBoost consistently outperforms Random Forest and SVM across all evaluation metrics, achieving 98.86% accuracy with 98.91% precision and 98.90% recall. The gradient boosting approach, combined with strong regularization and natural handling of feature interactions, makes it ideally suited for kernel-level behavioral classification.

Second, the kernel-level eBPF monitoring approach provides visibility into system behaviors that are invisible to user-space detection tools, particularly when dealing with BYOVD-based attacks that defeat user-mode security mechanisms. This architectural advantage is especially relevant against advanced threats like Qilin.

Third, a compact 15-feature set based on system call patterns proves sufficient for highly accurate classification, with the top five features accounting for over 60% of predictive importance. Write frequency, open frequency, read frequency, and entropy score emerge as the most discriminative features.

Fourth, the achieved detection latency of 1.8ms enables early ransomware identification before significant data loss occurs, making the framework practically deployable in production environments. This sub-2ms latency satisfies the real-time detection requirement established as a research objective.

Fifth, the low false positive rate of approximately 0.5% ensures operational viability without causing alert fatigue, a critical consideration for production security systems.

7. Future Work

Future work should explore deep learning integration (particularly LSTM and Transformer architectures for temporal pattern capture), real-time deployment optimization for production environments, extension to cloud-native and containerized workloads, multi-variant detection through transfer learning, and adversarial robustness evaluation. Additionally, adapting the framework to Windows through eBPF for Windows represents a valuable direction for broader applicability.

The contributions of this research advance the state of the art in ransomware detection and provide a foundation for continued development of kernel-level security monitoring capabilities. As ransomware threats continue to evolve, the integration of kernel-level visibility with advanced machine learning classification offers a robust and future-ready approach to cybersecurity defense.

Declaration

Author Contributions

The authors confirm their contributions to the paper as follows: Conceptualization, methodology, and original draft preparation; data curation, formal analysis, and validation: All authors

contributed equally, reviewed and approved the final version of the manuscript.

Data Availability

All relevant data and supporting material used in this study are in the manuscript.

Conflict of Interest

The authors state that none of the work described in this study could have been influenced by any known competing financial interests or personal relationships.

Ethics Approval

The study was conducted following standard ethical guidelines. Approval was obtained from the relevant institutional review board, and all participants provided informed consent.

REFERENCES

[1] A. Kharraz, S. Arshad, C. Mulliner, W. K. Robertson, and E. Kirda, "UNVEIL: A large scale, automated approach to detecting ransomware," in Proc. 25th USENIX Security Symposium, Austin, TX, USA, 2017, pp. 757 772.

[2] N. Scaife, H. Carter, P. Traynor, and K. R. B. Butler, "CryptoLock (and Drop It): Stopping ransomware attacks on user data," in Proc. IEEE 36th Int. Conf. Distributed Computing Systems (ICDCS), Nara, Japan, 2017, pp. 303 312.

[3] A. Continella et al., "ShieldFS: A self healing, ransomware aware filesystem," in Proc. 32nd Annual Conf. Computer Security Applications, Los Angeles, CA, USA, 2017, pp. 336 345.

[4] S. H. Kok, A. Abdullah, and N. Jhanjhi, "Ransomware, threat and detection techniques: A systematic literature review," Int. J. Cyber Security Digital Forensics, vol. 10, no. 3, pp. 176 196, 2021.

[5] S. Morato, S. Ballesteros, and J. Marzo, "Ransomware detection using machine learning: A systematic literature review," IEEE Access, vol. 8, pp. 195197
195227, 2020.

[6] S. Homayoun, A. Dehghantanha, and R. Mahmoud, "Machine learning techniques for ransomware detection: A critical review," in Cyber Threat Intelligence, M. Conti, A. Dehghantanha, and T. Dargahi, Eds. Cham, Switzerland: Springer, 2020, pp. 135 165.

[7] J. Kim et al., "LSTM based ransomware detection using storage level disk access patterns," IEEE Access, vol. 9, pp. 79942 79958, 2021.

[8] M. Hoo, J. Huang, J. Zheng, and J. Li, "eBPF based syscall monitoring for container security," in Proc. IEEE Conf. Communications and Network Security (CNS), Orlando, FL, USA, 2023, pp. 1 9.

[9] R. Toshiba, M. Hamaru, and K. Yoshioka, "Goodbye, sandbox: Evasion of kernel callbacks and its countermeasures," in Proc. 14th ACM Workshop Artificial Intelligence Security, Virtual, 2021, pp. 51 62.
48

[10] P. Cousot, R. Cousot, J. Feret, and X. Rival, "The eBPF revolution: Security and performance in the Linux kernel," IEEE Security Privacy, vol. 21, no. 4, pp. 28–37, Jul./Aug. 2023.

[11] D. Cabrera, J. Touceda, and J. Tolosana, "Ransomware detection using machine learning: A static analysis approach," in Proc. 14th Int. Conf. Computational Intelligence Security Information Systems, Bilbao, Spain, 2019, pp. 115–122.122.

[12] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2016, pp. 785–794.794.

[13] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.32, 2001.

[14] C. Cortes and V. Vapnik, "Support–vector networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.297, 1995.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.357, 2002.

[16] M. Ligh, A. Case, J. Levy, and A. Walters, The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory. Hoboken, NJ, USA: Wiley, 2014.

[17] S. Checkoway and H. Shacham, "Iago attacks: Why the system call API is a bad untrusted RPC interface," in Proc. 18th Int. Conf. Architectural Support Programming Languages Operating Systems (ASPLOS), Houston, TX, USA, 2013, pp. 253-264.264.

[18] E. Bacis, S. De Capitani, M. Locati, and M. Monga, "Ransomware and the legacy crypto API," in Proc. 11th Int. Conf. IT Security Incident Management IT Forensics, Nuremberg, Germany, 2017, pp. 11-28.28.

[19] A. Oest et al., "Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale," in Proc. 29th USENIX Security Symposium, 2020, pp. 1-18.18.

[20] M. Antonakakis et al., "Understanding the Mirai botnet," in Proc. 26th USENIX Security Symposium, Vancouver, BC, Canada, 2017, pp. 1093-11110.110.

[21] B. Dolan-Gavitt et al., "Virtuoso: Narrowing the semantic gap in virtual machine introspection," in Proc. IEEE Symp. Security Privacy, San Francisco, CA, USA, 2011, pp. 297-312.312.

[22] P. A. Loscocco et al., "The inevitability of failure: The flawed assumption of security in modern computing environments," in Proc. 21st National Information Systems Security Conf., Cleveland, OH, USA, 1998, pp. 303--314.314.

[23] D. P. Bovet and M. Cesati, Understanding the Linux Kernel, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2005.

[24] J. Corbet, A. Rubini, and G. Kroah-Hartman, Linux Device Drivers, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2005.

[25] D. Thang and H. Choi, "PCD: Paranoid concurrent detection for ransomware," in Proc. IEEE Int. Conf. Consumer Electronics (ICCE), Las Vegas, NV, USA, 2022, pp. 1-6.6.

[26] S. I. Popal, M. R. Haque, and A. A. Al-Mamun, "Ransomware analysis and detection framework using machine learning," in Proc. IEEE Int. Conf. Computer Communication Engineering (ICCCE), Kuala Lumpur, Malaysia, 2022, pp. 210-215.215.

[27] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware detection using hybrid machine learning algorithms," in Proc. IEEE Int. Conf. Cyber Security (CyberSE), Putrajaya, Malaysia, 2021, pp. 1-6.6.

[28] A. Gavriluţ, M. Iacob, and L. Bifă, "Ransomware detection using machine learning: A systematic literature review," in Proc. IEEE 16th Int. Symp. Applied 50 Computational Intelligence Informatics (SACI), Timisoara, Romania, 2022, pp. 000157000157-000162.000162.

[29] S. Poudel and J. I. Khan, "Kernel-level behavioral analysis for ransomware detection," in Proc. IEEE 19th Int. Conf. Trust, Security Privacy Computing Communications (TrustCom), Guangzhou, China, 2020, pp. 1122-1129.1129.

[30] K. Lee, S. Lee, and J. Yoon, "eBPF-based lightweight container runtime security monitoring," in Proc. IEEE Int. Conf. Cloud Computing (CLOUD), Chicago, IL, USA, 2023, pp. 345-352.352.