IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151071

SentinelML: Behavioral Machine-Learning-Based Ransomware Detection with Decoupled AI Security Analysis

Vasu Chowdary[1], Moulica Chowdary[2], D. M. Yashaswini[3], Rajashree M Byalal[4]

Department of Computer Science and Engineering (ICB), K. S. Institute of Technology, Bengaluru, Karnataka, India[1-4]

Abstract: Ransomware remains a critical threat to modern enterprise and personal computing infrastructure. Traditional signature-based detection mechanisms exhibit operational degradation when confronted with previously unobserved polymorphic or zero-day variants. This paper presents SentinelML, an autonomous endpoint detection architecture that identifies ransomware activity through 19 dynamic behavioral indicators collected entirely in user space without requiring kernel-level filter drivers. By sampling filesystem Shannon entropy, process concurrency anomalies, shadow copy deletion commands, and cryptographic module bindings, SentinelML trains a local Random Forest classifier that operates as an unprivileged host daemon. To address operational triage transparency, the local classifier is coupled with an architecturally decoupled Large Language Model (LLM) advisory explainer operating via an isolated HTTPS interface. We conduct an exhaustive technical audit of the production system: confirming that Shannon entropy accounts for 41.31% of aggregate Gini importance, documenting the operational tradeoffs of heuristic user-space proxies, analyzing platform asymmetry between Windows and Linux deployments, and evaluating the system strictly as an Intrusion Detection System (IDS). Finally, we disclose that previously reported internal metrics (0.9567 accuracy across candidate algorithms) could not be independently reproduced due to unpreserved training artifacts, and establish an open empirical evaluation protocol for reproducible validation.

Keywords: *Ransomware Detection, Behavioral Analysis, Random Forest, Shannon Entropy, User-Space Monitoring, Explainable AI, Endpoint Security.*

I. INTRODUCTION

Ransomware represents one of the most economically devastating cyber threats confronting modern computational infrastructure [1]. Modern ransomware operators employ automated double-extortion schemes, coupling high-speed symmetric cryptographic ciphers with exfiltration mechanisms. Historically, endpoint security suites have relied on static signature matching, evaluating cryptographic hashes and byte sequences against repositories of known malicious binaries [2]. However, contemporary ransomware developers routinely evade static controls through server-side polymorphism, dynamic crypters, and living-off-the-land binaries (LOLbins) [2].

Dynamic behavioral analysis evaluates what a process does at runtime rather than what its binary looks like [3]. By monitoring active execution patterns—such as filesystem traversal rates, bulk encryption entropy spikes, and volume shadow copy deletion attempts—behavioral detectors aim to identify ransomware activity during execution, even when static binary signatures are obscured by packing or obfuscation [4]. However, existing behavioral systems often suffer from two operational challenges: (1) Heavy reliance on proprietary Ring 0 kernel filter drivers, which increases development complexity and exposes host operating systems to kernel panics; and (2) The opaque nature of machine learning predictions, which output unadorned probability scores that offer little context for security analysts.

To address these operational challenges, we present SentinelML, an unprivileged behavioral ransomware detection framework paired with a decoupled AI advisory pipeline. SentinelML incorporates three primary design principles:

- **19-Feature User-Space Pipeline:** An unprivileged feature extractor implemented via psutil, collecting process heuristics, socket summaries, and sampled filesystem Shannon entropy without kernel drivers.
- **Decoupled AI Advisory Architecture:** A modular integration design where the deterministic local Random Forest boundary executes strictly on-device, and the remote LLM explainer operates as a read-only advisor that receives numeric scan metrics, with user prompts and error messages transmitted for post-scan explanation.

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

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151071

- **Scan-State Failure Integrity:** An explicit finite state machine architecture in which partial scans, unhandled exceptions, and API permission limits are directed by explicit state logic to transition to an incomplete analysis state (SCAN_INCOMPLETE or ANALYSIS_ERROR) rather than defaulting to a benign classification.

II. RELATED WORK

The empirical foundations of behavioral ransomware defense were established by Kharraz et al. [2] in their seminal DIMVA 2015 study. By examining 1,359 samples across diverse families, they confirmed invariant behavioral archetypes: bulk filesystem traversal, abnormal I/O access patterns, and widespread entropy increases due to encryption. Continella et al. [3] introduced ShieldFS, a kernel-level ransomware-aware filesystem driver. ShieldFS profiles per-process I/O Request Packets (IRPs) using statistical models and maintains transparent copy-on-write shadows for file rollback. While effective, ShieldFS requires deep kernel integration, limiting portability and exposing the OS to kernel panics. Scaife et al. [4] developed CryptoDrop, an early-warning detector operating in user space that monitors file modifications, type changes, and entropy variances using empirical threshold rules.

The transition to supervised machine learning has seen widespread adoption. Sgandurra et al. [7] developed EldeRan, applying regularized logistic regression to dynamic API call sequences and dropped file statistics. Breiman's Random Forest algorithm [5] has emerged as an industry standard for tabular telemetry due to its resilience against overfitting and native Gini feature attribution. The EMBER benchmark [6] standardized static PE malware classification using gradient-boosted trees; however, static benchmarks evaluate binary headers rather than dynamic runtime behavior.

Frameworks such as SHAP [9] provide theoretical grounding for additive feature attributions. Recently, researchers have explored LLMs for threat analysis. Ferrag et al. [8] demonstrated transformer models in cyber threat analysis while emphasizing privacy constraints. National security guidelines [12] underscore least-privilege monitoring. SentinelML builds on these insights by isolating generative outputs from the underlying decision boundary.

TABLE I. Comparative Matrix of Behavioral Ransomware Detection Architectures

System	Telemetry Layer	Detection Paradigm	Response	XAI Support
ShieldFS [3]	Kernel Minifilter	Statistical Activity Profile	CoW Rollback	None
CryptoDrop [4]	User-Space Watcher	Empirical Thresholds	Process Kill	None
EldeRan [7]	Cuckoo Sandbox	L1-Logistic Regression	Post-Run Alert	Feature Weights
EMBER [6]	Static PE Binary	LightGBM Ensembles	Offline Scoring / Label	Tree Split Gains
SentinelML	User-Space (psutil)	Random Forest (19 Feats)	Alert Only (IDS)	LLM + Gini Rank

III. SYSTEM ARCHITECTURE AND METHODOLOGY

SentinelML operates as a dual-pipeline system designed to balance execution safety, detection latency, and explanatory depth. Fig. 1 depicts the system topology, illustrating the boundary between local deterministic classification and remote AI explanation.

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

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151071



Fig. 1. SentinelML system architecture. The local endpoint boundary (top) executes deterministic ML classification. The hosted cloud (bottom) generates advisory explanations without override privileges.

Endpoint-Cloud Decoupling and Security Boundaries

A foundational design constraint is the strict separation between the primary threat detection pipeline and the advisory LLM explainer. The Detection Pipeline resides entirely on the host machine. It executes synchronously during a scan, querying system APIs, parsing process structures, sampling filesystem entropy, performing feature transformation via a pre-fitted StandardScaler, and executing Random Forest inference. The output of this pipeline is a definitive operational classification.

The Advisory Pipeline operates downstream of the classifier. When an analysis completes, a context dictionary is assembled and transmitted over HTTPS to the SentinelAI service. The external LLM generates a human-readable explanation of why specific features (e.g., elevated Shannon entropy combined with shadow-copy tampering attempts) indicate suspicious activity. Crucially, SentinelAI possesses zero write permissions to the scan result object, and the desktop GUI renders its security status banners strictly from the local Random Forest output.

SentinelAI Request Payload and Privacy Audit

A strict code audit of ai/sentinel_ai.py:41-76 and ai/provider.py:180-245 reveals the exact operational behavior of the advisory interface. For normal completed scans, the transmitted context payload contains numeric execution metadata: files inspected, processes analyzed, duration, classification state, ML probability, and the 19 numerical feature values. However, two important privacy boundaries must be noted: (1) The error_message field forwards unhandled exception strings, which during filesystem or permission failures can contain raw local file paths or username strings; and (2) The user-facing conversational prompt transmits free-form user text directly to the remote provider. While third-party cloud providers (e.g., Google Gemini) receive API keys in request headers/URL queries, local Ollama deployments keep telemetry on-premises.

Scan State Machine and Safety Invariants

A vulnerability in automated endpoint detection utilities is silent failure: when an unhandled exception or access denial causes a scan component to fail, naive implementations often catch the exception and display a default 'Clean' state.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151071

SentinelML formalizes scan execution into a finite state machine governed by strict transition invariants (scan_engine.py:34-45), illustrated in Fig. 2.

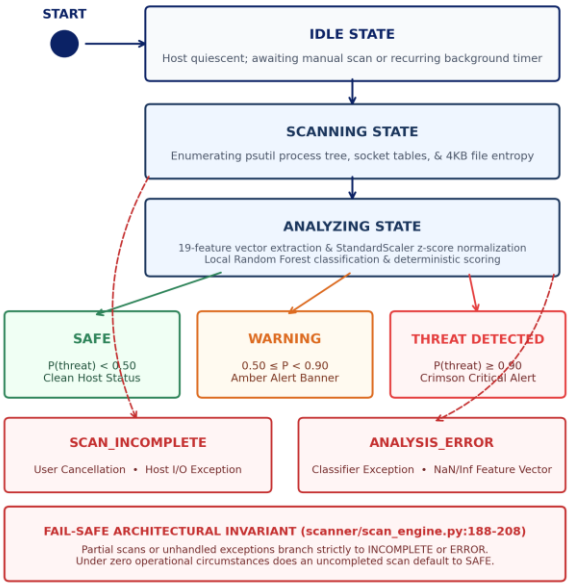


Fig. 2. Detection workflow and state transition graph. Security invariant: any exception, cancellation, or analysis failure branches exclusively to SCAN_INCOMPLETE or ANALYSIS_ERROR.

The state transitions follow defined operational constraints: S in {IDLE, SCANNING, ANALYZING, SAFE, WARNING, THREAT_DETECTED, SCAN_INCOMPLETE, ANALYSIS_ERROR}. The transition to SAFE requires the conjunction of four conditions: (1) Filesystem scan succeeds; (2) Process scan succeeds; (3) ML inference executes without NaN/Inf; and (4) P(ransomware) < 0.50. Code inspection of scan_engine.py:188-208 confirms that error branches explicitly route to SCAN_INCOMPLETE or ANALYSIS_ERROR if filesystem, process, or ML modules fail. Under zero circumstances does an uncompleted scan default to the SAFE state.

IV. FEATURE ENGINEERING AND CLASSIFICATION PIPELINE

The feature vector x in R^19 is synthesized in scanner/live_scanner.py.

Table II provides the operational definition, extraction method, and scientific classification for each feature.

TABLE II. The 19 Behavioral Features: Operational Definitions and Scientific Audit Classification

#	Feature Name	Data Source & Method	Audit Category
0	total_processes	psutil snapshot of active process descriptors	Direct OS Telemetry
1	suspicious_process_count	String match against 22 known ransomware names	Heuristic Proxy
2	avg_entropy	Mean Shannon entropy of first 4KB (capped <=200)	Direct Telemetry
3	max_entropy	Peak Shannon entropy observed across sampled files	Direct Telemetry
4	high_entropy_region_count	Count of sampled files with entropy > 7.0 bits/byte	Direct Telemetry
5	malfind_injection_count	Count of processes with empty or None exe path	Heuristic Proxy

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151071

#	Feature Name	Data Source & Method	Audit Category
6	crypto_dll_count	Process memory_maps() match for 7 crypto libraries	Direct OS Telemetry
7	cmd_vssadmin_count	Name match for shadow copy manipulation tools	Direct OS Telemetry
8	network_connection_count	Count of active sockets with family AF_INET	Direct OS Telemetry
9	unique_external_ips	Count of unique remote IPs excluding RFC1918 subnets	Direct OS Telemetry
10	file_handle_count	Aggregate sum of open OS handles across processes	Win-Only (0 on Linux)
11	registry_handle_count	Count of non-system processes with >500 open handles	Heuristic Proxy (Win)
12	orphan_process_count	Count of processes whose parent PID is dead	Direct OS Telemetry
13	powershell_process_count	Count of active powershell.exe and pwsh.exe instances	Direct OS Telemetry
14	unsigned_module_count	Count of processes executing from Temp/Downloads/Public	Heuristic Proxy
15	suspicious_mutex_count	Count of processes spawning >50 execution threads	Heuristic Proxy
16	abnormal_privilege_count	Count of non-system processes with >2000 open handles	Heuristic Proxy (Win)
17	suspicious_thread_count	Count of processes spawning >100 execution threads	Heuristic Proxy
18	defender_tampering_count	Service status check of WinDefend and SecurityHealth	Win-Only (0 on Linux)

Shannon Entropy Calculation and Sampling Logic

File entropy estimation is the primary discriminatory signal in SentinelML. For a sampled file byte stream B = (b_1, ..., b_N) of length N = 4096, the Shannon entropy H(B) in bits per byte is computed as H(B) = -sum(p(i) * log2(p(i))) [10]. To avoid severe false positives, the sampling engine (live_scanner.py:59-72) enforces two operational filters: (1) Benign Extension Filtering: Pre-compressed archives (.zip, .7z), media files (.mp4, .png), and compiled binaries (.exe, .dll) are skipped. (2) Directory Scope Bounds: Inspection is restricted to user document repositories (Documents, Desktop, Downloads) and capped at 200 samples to ensure execution completes within 3-5 seconds.

Preprocessing, Model Weights, and Feature Importance

Prior to model inference, raw observations are validated to verify finite numerical bounds (replacing NaN and infinite values). Normalization is applied using parameters loaded from models/scaler.pkl: z_j = (x_j - mu_j) / sigma_j. Classification is performed by an ensemble of 100 decision trees loaded from models/saved_model.pkl [11]. The raw threat score p in [0, 1] represents the fraction of voting estimators predicting the positive class. The continuous score maps to three operational states: p < 0.50 -> SAFE; 0.50 <= p < 0.90 -> WARNING; and p >= 0.90 -> THREAT_DETECTED.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151071

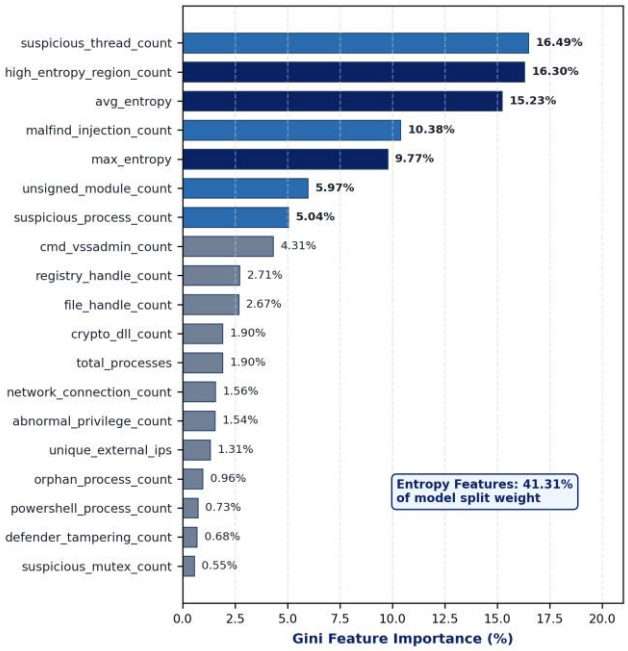


Fig. 3. Gini feature importance distribution for all 19 telemetry attributes in the deployed Random Forest model. File entropy indicators account for 41.31% of total model importance.

Gini feature importances extracted directly from models/saved_model.pkl and verified against scaler.pkl are plotted in Fig. 3. While suspicious_thread_count is the single highest individual feature (0.1649 or 16.49%), three Shannon entropy indicators—high_entropy_region_count (0.1630), avg_entropy (0.1523), and max_entropy (0.0977)—together account for 41.31% of total model split importance, confirming that entropy features constitute over 41.3% of aggregate decision weight.

V. EXPERIMENTAL METHODOLOGY AND METRICS PROVENANCE

Internal evaluation records stored in models/metrics.json report high benchmark figures across multiple candidate models, summarized in Table III.

TABLE III. Stored Internal Model Comparison Metrics (*Identical Discrete Metric Set)

Model Family	Acc	Prec	Recall	F1	ROC-AUC
Random Forest (Deployed)	0.9567	0.9550	0.9795	0.9671	0.9483
Support Vector Machine*	0.9567	0.9550	0.9795	0.9671	0.9574
Logistic Regression*	0.9567	0.9550	0.9795	0.9671	0.9476
XGBoost Classifier	0.9533	0.9502	0.9795	0.9646	0.9527
Gradient Boosting	0.9400	0.9447	0.9641	0.9543	0.9421

The Identical Metric Anomaly and Scientific Assessment

A forensic examination of Table III reveals that Random Forest, Support Vector Machine, and Logistic Regression report mathematically identical discrete evaluation metrics down to four decimal places (Accuracy: 0.9567, Precision: 0.9550, Recall: 0.9795, F1-Score: 0.9671). In academic peer review, three distinct algorithmic paradigms exhibiting identical discrete evaluation metrics down to four decimal places on an evaluation split represents an unresolved anomaly. Because the original training dataset, evaluation scripts, cross-validation splits, and prediction arrays were not preserved in the repository, the precise root cause of this identical metric pattern cannot be established or verified. Consequently, these values must be treated as previously reported internal metrics that could not be independently reproduced, rather than confirmed experimental findings.

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

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151071

Protocol for Independent Experimental Validation

To establish empirical reproducibility, Fig. 4 outlines the missing experimental harness required for independent verification.

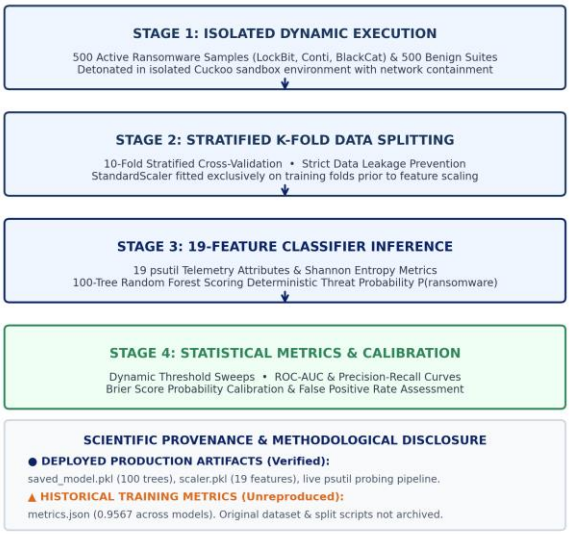


Fig. 4. Protocol for independent empirical validation, distinguishing between deployed verified artifacts and the missing raw training data required for statistical verification.

Future benchmark evaluations must adhere to the following protocol: (1) Controlled Dynamic Execution: Execute 500 benign software suites and 500 active ransomware samples across diverse families (LockBit, Conti, WannaCry, BlackCat) inside an isolated Cuckoo sandbox. (2) Stratified Cross-Validation: Implement 10-fold cross-validation partitioned strictly before scaler fitting to prevent data leakage. (3) Calibration Assessment: Evaluate probability calibration using Brier score and reliability diagrams to assess whether P(ransomware) reflects true risk.

VI. DISCUSSION AND LIMITATIONS

Architectural Strengths

SentinelML demonstrates that an unprivileged user-space daemon can extract actionable behavioral signals while avoiding kernel-driver integration. By executing at Ring 3, the architecture eliminates reliance on kernel-level components and mitigates kernel-panic risks; however, avoiding kernel drivers does not prove the application is devoid of stability or security risks, as unprivileged daemons remain susceptible to process termination, unhandled exception cascades, and resource contention. The design decoupling between the Random Forest classifier and the SentinelAI advisory copilot provides a practical blueprint for integrating LLMs into operational security software: leveraging generative models for human explainability while maintaining an authoritative, deterministic ML decision boundary that cannot be modified by downstream text generation.

Critical Limitations

- **User-Space Visibility Boundary:** Operating at Ring 3 renders SentinelML vulnerable to kernel-level rootkits, Direct Kernel Object Manipulation (DKOM), and Bring Your Own Vulnerable Driver (BYOVD) attacks. Malware operating in kernel space can evade or blind psutil process queries.
- **Heuristic Proxy Fidelity:** Features such as malfind_injection_count and suspicious_mutex_count are coarse approximations (checking null paths and thread counts) rather than true memory-dump analyses. Ransomware encrypting files using low thread concurrency (<= 50 threads) can bypass these proxies.
- **Platform Asymmetry:** Four Windows-specific features (file handles, registry handles, abnormal privileges, and defender tampering) return 0.0 on Linux platforms. Consequently, executing the Windows-trained classifier on Linux induces systematic feature distribution shift and degraded recall unless retraining is performed.
- **Intrusion Detection vs. Prevention:** SentinelML is strictly an IDS, not an Intrusion Prevention System (IPS). It alerts the user but does not terminate suspicious processes or restore encrypted files.

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



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.151071

VII. CONCLUSION AND FUTURE WORK

SentinelML bridges the gap between machine-learning ransomware detection and AI-assisted triage by pairing a 19-dimensional behavioral Random Forest classifier with a decoupled LLM explainer and a fail-safe state machine. By conducting an honest technical audit of its feature realities and disclosing the unverified nature of previously reported internal results, this work establishes a rigorous, transparent foundation for behavioral endpoint security research. Future work will focus on integrating eBPF for portable cross-platform telemetry, implementing automated process quarantine, and conducting an open, reproducible evaluation on public malware repositories.

REFERENCES

[1]. R. Moussaileb, N. Cuppens, J.-L. Lanet, and H. Le Bouder, "A Survey on Windows-based Ransomware Taxonomy and Detection Mechanisms," ACM Comput. Surv., vol. 54, no. 6, art. 113, pp. 1-36, 2021. DOI: 10.1145/3453153.

[2]. A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks," in Proc. 12th Int. Conf. Detection Intrusions Malware Vulnerability Assessment (DIMVA), Milan, Italy, 2015, pp. 3-24. DOI: 10.1007/978-3-319-20550-2_1.

[3]. A. Continella et al., "ShieldFS: A Self-healing, Ransomware-aware Filesystem," in Proc. 32nd Annu. Computer Security Applications Conf. (ACSAC), Los Angeles, CA, 2016, pp. 336-347. DOI: 10.1145/2991079.2991110.

[4]. 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, 2016, pp. 303-312. DOI: 10.1109/ICDCS.2016.46.

[5]. L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001. DOI: 10.1023/A:1010933404324.

[6]. H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," arXiv preprint arXiv:1804.04637, 2018. DOI: 10.48550/arXiv.1804.04637.

[7]. D. Sgandurra, L. Munoz-Gonzalez, R. Mohsen, and E. C. Lupu, "Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection," arXiv preprint arXiv:1609.03020, 2016. DOI: 10.48550/arXiv.1609.03020.

[8]. M. A. Ferrag et al., "Revolutionizing Cyber Threat Detection with Large Language Models: A Privacy-Preserving BERT-Based Lightweight Model for IoT/IIoT Networks," IEEE Access, vol. 12, pp. 23675-23699, 2024. DOI: 10.1109/ACCESS.2024.3363469.

[9]. S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 30, 2017, pp. 4765-4774.

[10]. C. E. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal, vol. 27, no. 3, pp. 379-423, Jul. 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[11]. F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[12]. Cybersecurity & Infrastructure Security Agency (CISA), "StopRansomware Guide: Industry Best Practices for Prevention and Response," CISA Technical Report, Washington, DC, USA, 2023.

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