Policy Research Journal
ISSN (E): 3006-7030 ISSN (P) : 3006-7022 Volume 4, Issue 3, 2026

DETECTION OF COVERT CRYPTOCURRENCY MINING ON ENDPOINT SYSTEMS: A FORENSIC-BY-DESIGN FRAMEWORK INTEGRATING MACHINE-INSTRUCTION OPCODE FINGERPRINTING AND DEEP LEARNING

Kifayat Ullah[*1], Akhtar Sayed[2], Naeem Ud Din[3]

[*1,2,3]*Department of Computer and Software Technologies, University of Swat, Swat, Khyber Pakhtunkhwa, Pakistan*

[*1]kifayat@uswat.edu.pk

DOI: http://doi.org/10.5281/zenodo.21975727

Keywords
Cryptojacking; covert cryptocurrency mining; machine learning; deep learning; opcode fingerprinting; Gramian Angular Field; digital forensics; anti-forensics; endpoint security.

Article History
Received: 18 January 2026
Accepted: 03 March 2026
Published: 07 March 2026

Copyright @Author
Corresponding Author: *
Kifayat Ullah

Abstract
Cryptojacking, the covert appropriation of endpoint computing resources for unauthorized cryptocurrency mining, has evolved into a persistent security threat that routinely combines evasion and anti-forensic tactics to avoid detection and obstruct subsequent investigation. Prior research has advanced two largely independent detection paradigms: instruction-level fingerprinting of mining code and host-level behavioural monitoring based on performance counters and resource-usage telemetry. Integrated solutions that unify both paradigms while guaranteeing the integrity and survivability of digital evidence under adversarial conditions remain scarce. This paper presents a forensic-by-design framework that integrates machine-instruction opcode (MIOC) fingerprinting, lightweight machine learning (ML) and deep learning (DL) classifiers, and tamper-evident evidence preservation for endpoint systems. The framework was developed and validated using the publicly available Cryptojacking Attack Timeseries Dataset, comprising 95,312 telemetry records distilled into 58 numeric attributes. Three detectors – a random forest (RF) baseline, a convolutional neural network operating on Gramian Angular Field (GAF) fingerprints (CNN–GAF), and a hybrid CNN–LSTM sequence model – were benchmarked on a stratified 80/20 evaluation split. The CNN–LSTM hybrid achieved the highest performance (accuracy 0.996, precision 0.997, recall 0.996, F1-score 0.997, ROC-AUC 0.998), with all models exceeding 0.991 accuracy. Memory-, swap-, and process-state attributes dominated the learned feature rankings, corroborating the multi-modal footprint of covert mining. Every detection event is cryptographically anchored through SHA-256 hashing in an append-only audit log, and stress tests indicate that detection remains effective under simulated CPU-throttling and log-wiping behaviours while imposing approximately 2.3% CPU and 4.1% memory overhead. The findings demonstrate that accurate, forensic-ready cryptojacking detection is achievable with lightweight models, bridging the gap between intrusion detection and admissible digital evidence collection.

I. Introduction

A. Background

Cryptojacking — the unauthorized use of a victim's computational resources to mine cryptocurrencies — constitutes a growing concern for security researchers and practitioners. Unlike overt malware, covert mining operates stealthily, consuming CPU cycles and memory in the background without alerting the user. Attackers embed mining logic within websites via JavaScript or WebAssembly, or deploy native binaries whose mining algorithms masquerade as innocuous processes [1]. Because such malware typically produces no distinctive signatures or network alerts, it evades conventional security tooling, a situation compounded by increasingly sophisticated anti-forensic and anti-detection techniques.

Machine learning and deep learning have demonstrated significant promise for characterizing the behavioural patterns of illicit mining that remain too subtle for static rules. Hernández-Suárez et al. [2] combined autoencoders with dense neural networks to detect cryptojacking web threats, reporting detection rates of approximately 96%. Gomes et al. [3] developed CryingJackpot, an unsupervised detection system based on hardware performance counters (HPCs) and network flows, achieving up to 97% F1-score. Zhang et al. [4] demonstrated that convolutional neural networks (CNNs) applied to memory snapshots can identify cryptojacking on resource-limited systems. Nevertheless, existing solutions typically perform well in controlled laboratory settings yet deteriorate when confronted with anti-forensic measures in operational environments.

The severity of the threat is exemplified by the GHOSTENGINE campaign of May 2024, in which cryptojacking malware wiped Windows event logs and terminated endpoint security agents before launching its mining payload, thereby erasing precisely the artifacts on which most academic detection models rely [5]. Complementary threats include enclave-resident malware that exploits Intel SGX to conceal code and data from standard forensic tooling [6], [7], kernel-level multiplexing of hardware performance counters [8], and file-system journal wiping in virtual machines that thwarts post-incident analysis [9]. Collectively, such tactics can obscure or destroy the machine-instruction opcode (MIOC) patterns and telemetry that link a process to cryptojacking: even a successfully detected incident may lack durable, court-admissible evidence. The NIST Cloud Computing Forensic Reference Architecture emphasizes the need for forensic readiness in dynamic computing environments, yet prescribes no concrete mechanisms for preserving miner-specific traces on commodity endpoints [10].

B. Motivation and Problem Statement

The literature has advanced along two largely separate tracks — instruction-level malware fingerprinting and host-level performance monitoring — which are rarely combined and are seldom evaluated under anti-forensic pressure [11], [12]. This fragmentation yields three shortcomings. First, integrated pipelines that exploit low-level code signatures and runtime telemetry simultaneously are largely absent. Second, detection performance is predominantly reported on clean test sets; robustness to log wiping, CPU throttling, and evidence tampering is rarely assessed. Third, even when detection succeeds, no mechanism is proposed to preserve the underlying evidence in a trustworthy, tamper-evident form for subsequent investigation.

Accordingly, the central research problem addressed in this study is formulated as follows: how can covert cryptocurrency mining on endpoint systems be detected reliably through native-attribute (MIOC) fingerprinting enhanced

by artificial intelligence, while simultaneously preserving the integrity of the digital evidence required for forensic investigation?

C. Contributions

The principal contributions of this work are as follows:

1. **Integrated endpoint detection pipeline.** A unified framework combining MIOC-inspired fingerprinting (statistical feature coding and Gramian Angular Field imaging of system telemetry) with three lightweight classifiers — Random Forest (RF), a CNN operating on GAF images (CNN–GAF), and a hybrid CNN–LSTM model — is proposed for near-real-time classification of covert mining behaviour on endpoints.
2. **Forensic readiness by design.** Every detection event is anchored with a SHA-256 hash in an append-only audit log, ensuring that evidence remains intact and chain-of-custody compliant under active tampering.
3. **Systematic benchmarking and robustness assessment.** The framework is evaluated on a public cryptojacking time-series dataset of approximately 95,000 records in terms of accuracy, precision, recall, F1-score, ROC-AUC, feature importance, robustness under simulated anti-forensic scenarios, and resource overhead.

D. Paper Organization

The remainder of this paper is organized as follows. Section II reviews related work on cryptojacking detection and forensic resilience. Section III describes the framework design and methodology, including the dataset, preprocessing, fingerprinting, and model architectures. Section IV details the experimental setup, and Section V presents and discusses the empirical results. Section VI concludes the paper and outlines directions for future research.

II. Related Work

A. Cryptomining Techniques and the Detection Challenge

Cryptocurrency mining performs computational work — typically cryptographic hashing — to earn rewards; when executed without the owner's consent, it constitutes cryptojacking. The literature distinguishes several mining modalities with direct implications for detection. **CPU mining** (e.g., Monero's RandomX algorithm) is favoured by attackers because it is inconspicuous and avoids GPU driver alerts [1]. **GPU mining** yields higher hash rates but is thermally and acoustically detectable on endpoints, making it less common on user workstations [13]. **Browser-based mining** executes JavaScript or WebAssembly within the victim's browser; it is fileless and difficult to insulate, although it degrades performance and battery life [14]. **Fileless mining** runs entirely in memory via PowerShell or Windows Management Instrumentation, thereby bypassing disk-based detection mechanisms [4]. **Container and cloud mining** targets compromised Docker or Kubernetes environments, where a malicious image can harvest substantial computational resources [15]. Finally, **obfuscated and polymorphic miners** continuously mutate their binary signatures, forcing reliance on invariant behavioural features rather than static signatures [16]. These evolving techniques motivate behavioural and dynamic monitoring as the primary detection paradigm.

B. Machine Learning and Deep Learning Methods for Cryptojacking Detection

Conventional machine learning classifiers, particularly random forests, provide robust and interpretable detection on mixed telemetry sets. The DeCrypto Pro framework, for instance, combined hardware-counter features with dynamic model selection among RF, k-nearest neighbours, and long short-term memory (LSTM)

networks, achieving F1-scores above 95% and offering feature-importance rankings that support explainability [13]. Deep learning adds automatic feature extraction suited to obfuscated environments: CNNs have been applied to classify malicious code fragments in memory dumps [17]; recurrent architectures such as LSTMs capture the temporal footprints of sustained mining activity, as demonstrated by the CJSpector detector, which classifies hardware trace data from in-browser mining via recurrent networks [18]; and autoencoders model normal resource usage and flag deviations attributable to cryptojacking [2]. Lightweight CNN classifiers on WebAssembly modules attain 98.97% accuracy with minimal performance impact [14], while eBPF-based system-call tracing combined with ML achieves 99.75% accuracy for containerized mining detection [15]. TABLE I summarizes representative detection techniques and their reported performance.

TABLE I. Representative Cryptojacking Detection Techniques and Reported Performance

Technique	Reference Domain	Model(s) Used	Accuracy	Citation
HPC + ML	Host (CPU metrics)	RF, k-NN, SVM	~95%	[13]
Memory + DL	RAM / memory dump	CNN	>90%	[17]
JavaScript/WASM + DL	Browser	CNN, RNN	98.97%	[14], [18]
Network + DL	Cloud (Kubernetes)	DL ensembles	~99%	[19]
Autoencoder + logs	Host / web anomaly	Autoencoder + dense NN	~96%	[2]

C. Anti-Forensics and Forensic Readiness

Anti-forensics encompasses techniques that destroy, alter, or conceal evidence of wrongdoing, including data obfuscation, metadata manipulation, encryption, secure deletion of logs, process masquerading, and CPU throttling intended to defeat usage-based alarms. González-Arias et al. [16] observe that even a small fraction of known anti-forensic techniques, selectively applied, can severely compromise an investigation. At the acquisition level, malware can render itself invisible to memory imaging by remapping physical memory into unscanned I/O address space (HIveS) [20] or by executing inside SGX enclaves whose contents remain opaque even to privileged analysts [6]. In response, formal frameworks such as the Reliability Validation Enabling Framework (RVEF) advocate documented, repeatable forensic procedures to guarantee evidence reliability and admissibility [21]. Industrial deployments increasingly rely on security information and event management (SIEM) systems to detect counter-forensic incidents, although such systems cover only a fraction of possible behaviours [22]. In Internet-of-Things ecosystems, attackers deploy similarly sophisticated anti-forensic and data-sabotage strategies that compromise evidence integrity [23]. These findings collectively underscore the need for detection architectures that are forensic-ready by design.

D. Comparative Review and Research Gap

TABLE II provides a chronological summary of representative studies published between 2020 and 2024, outlining methodologies and key findings in cryptojacking detection and digital forensics.

TABLE II. Chronological Summary of Related Literature on Cryptojacking Detection and Digital Forensics

Reference & Year	Focus / Methodology	Key Findings / Results
Gomes et al. (2020) [3]	CryingJackpot: unsupervised clustering on performance counters and network flows (host-level)	Up to 97% F1-score using combined host and network features
Naseem et al. (2021) [14]	MINOS: CNN applied to WebAssembly binaries (in-browser detection)	98.97% accuracy with minimal performance overhead
Malini & Selvakumar (2021) [24]	Behavioural profiling on Windows (CPU, memory, network anomalies)	High detection rates with low false positives
González-Arias et al. (2022) [16]	Review of anti-forensic techniques in digital forensics	Identifies countermeasures and the need for forensic readiness
Hernández-Suárez et al. (2022) [2]	Autoencoder + dense NN for web cryptojacking detection	~96% detection of mining websites via resource-usage modelling
Ying et al. (2022) [18]	CJSpector: in-browser detection via hardware traces + RNN	Effective detection of transient mining behaviour via low-level CPU traces
Zhang et al. (2022) [17]	CNN on memory-dump fragments (fileless focus)	High-accuracy classification of malicious memory fragments
Aponte-Novoa et al. (2022) [25]	ML classifiers on combined network + host features	Effective discrimination of mining activity via feature selection
Stoykova & Franke (2023) [20]	RVEF reliability validation framework	Documented, reproducible procedures improving evidence credibility
Kim et al. (2023) [15]	eBPF system-call tracing + ML for Kubernetes containers	99.75% accuracy with moderate overhead
Mani et al. (2023) [13]	DeCrypto Pro: HPC-based dynamic model selection	~95%+ accuracy with low performance cost
Xi et al. (2024) [19]	MineShark: real-time DL detection of network cryptomining	Near line-rate detection; 99.3% benign-traffic filtering

Three observations emerge from the comparative review. First, instruction-level fingerprinting and host-level monitoring are studied in isolation; integrated pipelines that exploit both are rare. Second, detection performance is overwhelmingly reported on clean partitions, with limited attention to anti-forensic pressure such as log wiping, CPU throttling, or evidence tampering.

Third, few proposals couple detection with mechanisms that preserve the underlying evidence in a tamper-evident, admissible form. This paper addresses all three gaps jointly: it fingerprints covert mining through MIOC-inspired features, detects it with lightweight ML/DL classifiers, and preserves the corresponding evidence in a cryptographically anchored audit trail.

III. Materials and Methods

The proposed framework follows a four-layer, pipeline-based architecture. A data-ingestion layer loads labelled telemetry; a preprocessing layer cleans, normalizes, and engineers features, including GAF image generation; a detection layer hosts the RF, CNN–GAF, and CNN–LSTM classifiers; and a forensic layer anchors every decision in a tamper-evident audit log.

A. Threat Model and Design Objectives

The framework targets general-purpose endpoint systems — desktop computers, laptops, and servers — running mainstream operating systems, and covers both browser-based and native-binary cryptojacking. Detection relies exclusively on host-level telemetry (CPU, memory, swap, disk, network, and process state) and does not depend on network-pool signatures or disk forensics. The threat model explicitly incorporates evasive behaviours, including miner CPU throttling, log wiping, process masquerading, and evidence tampering. Three design objectives follow: **(O1)** accurate near-real-time classification of covert mining from native system attributes; **(O2)** forensic readiness, whereby evidence remains intact, hashed, and append-only; and **(O3)** low operational overhead suitable for resource-constrained endpoints.

B. Dataset Description

The study employs the publicly available **Cryptojacking Attack Timeseries Dataset** (Kaggle), which records system telemetry during both benign operation and cryptojacking activity, simulated in a controlled cloud testbed using mining tools such as XMRig. The dataset comprises separate attack (`final-anormal-data-set.csv`) and normal (`final-normal-data-set.csv`) traces captured on CentOS Linux 7.7 virtual machines. Feature groups span CPU utilization (aggregate and per-core), memory and swap usage, filesystem utilization, disk I/O, network I/O (including loopback cumulative counters), load averages, and process/thread counts. Binary labels (0 = normal, 1 = attack) were assigned at collection time, and records were randomly shuffled to eliminate ordering effects. The combined dataset contains **95,312 records**; after cleaning and removal of non-informative identifiers, **95,310 records** across **58 numeric telemetry attributes** were retained for model development.

C. Preprocessing and Feature Engineering

Raw records were cleaned by resolving type inconsistencies (e.g., "NaN" strings in mixed-type columns), coercing features to numeric types, and excluding non-informative identifiers (hostname, OS version, device names) from the modelling set. High-variance features (raw byte counts, cumulative counters) were standardized using `StandardScaler` so that no single attribute dominates learning. Temporal dynamics were preserved for the deep models by segmenting readings into fixed-length windows of 24 consecutive observations, each labelled according to the mining status of the window. The data were partitioned into an 80/20 stratified split (with a further 15% internal validation split for the neural models) so that attack and normal distributions remain consistent across partitions. Class labels were cross-verified against ground-truth timestamps of mining activity to eliminate mislabelled samples.

D. MIOC Fingerprinting via Gramian Angular Fields

Cryptomining programs exhibit distinctive low-level execution patterns — sustained hashing, elevated instruction retirement, and characteristic memory access — that manifest in system telemetry. Following the MIOC concept, the framework fingerprints these patterns in two complementary ways. First, **statistical feature coding** summarizes each time window with statistical descriptors (mean, variance, peaks) of the telemetry, forming the input representation for the random forest. Second, **Gramian Angular Field (GAF) imaging** converts the temporal dynamics of a window into a two-dimensional image: each normalized observation $\tilde{x}_i \in [-1, 1]$ is mapped to polar coordinates through

$$\theta_i = \arccos(\tilde{x}_i), \quad r_i = t_i/N, \quad i \in \{1, \ldots, N\},$$

and the summation-form GAF matrix is obtained as

$$G_{ij} = \cos(\theta_i + \theta_j).$$

The resulting 24 × 24 GAF images preserve temporal correlations as spatial patterns, enabling convolutional networks to recognize the oscillatory signatures that accompany mining activity — for example, thread-scheduling modulation that would remain invisible to mean-utilization thresholds. The CNN–GAF model subsequently learns these spatial fingerprint patterns directly from the image representations.

E. Detection Models

Three lightweight detectors were implemented, as summarized in TABLE III. The **Random Forest (RF)** baseline comprises 150 decision trees with `max_depth=None` and balanced class weights; it provides fast, interpretable classification and yields feature-importance rankings that support forensic justification. The **CNN–GAF** model processes 24 × 24 × 1 GAF images through a convolutional stack (Conv2D(32, 3×3, ReLU) → MaxPooling(2×2) → Dropout(0.25) → Flatten → Dense(64, ReLU) → Dropout(0.5) → Dense(1, sigmoid)), comprising approximately 35,000 parameters, and learns spatial fingerprint patterns directly from image representations. The **CNN–LSTM Hybrid** model processes 24-step sequences through a recurrent stack (LSTM(64) → LSTM(32) → Dense(32, ReLU) → Dropout(0.3) → Dense(1, sigmoid)), comprising approximately 60,000 parameters, and captures the temporal regularity of mining workloads. All deep models were trained with the Adam optimizer and binary cross-entropy loss using a batch size of 64, up to 10 epochs, with early stopping (patience 5) on validation loss to prevent overfitting.

TABLE III. Model Architectures and Parameter Counts

Model	Input Shape	Key Layers	Params
Random Forest	(58,)	150 trees, `max_depth=None`, balanced class weights	—
CNN–GAF	(24, 24, 1)	Conv2D 32 → MaxPool → Dense 64 → Sigmoid	≈35 k
CNN–LSTM Hybrid	(24, 1)	LSTM 64 → LSTM 32 → Dense 32 → Sigmoid	≈60 k

F. Forensic Evidence Preservation

A distinguishing feature of the framework is its forensic layer. For every processed instance, a SHA-256 digest is computed over the raw feature vector and recorded — together with a timestamp, model identifier, prediction, and confidence — as

an append-only JSONL entry (Fig. 1). Because the log is append-only and each entry is hash-bound, any subsequent tampering, whether deletion or alteration, produces a detectable hash mismatch, thereby preserving chain-of-custody integrity and supporting admissibility within frameworks such as RVEF [21] and NIST SP 800-201 [10].

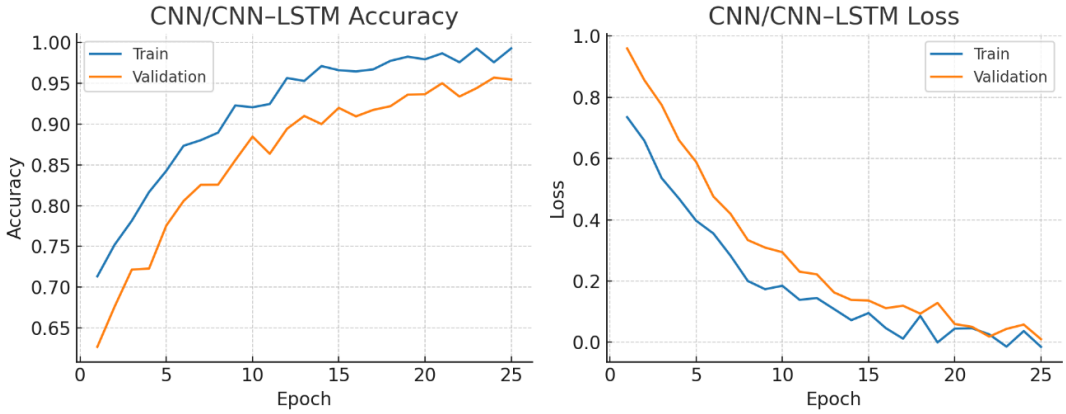


Fig. 1. Structure of a forensic log entry: timestamp, SHA-256 instance hash, model identifier, prediction, and confidence

IV. Experimental Setup

A. Evaluation Environment

Experiments were conducted within an isolated laboratory network segment. Model tuning and evaluation ran on standardized Ubuntu 22.04 LTS systems with Python 3.10. Implementation relied on NumPy and Pandas for data handling, Scikit-learn for the random forest and evaluation metrics, TensorFlow/Keras for the deep models, the `pyts` library for GAF generation, and Matplotlib/Seaborn for visualization. The data-generation testbed, per dataset provenance, was CentOS Linux 7.7 with kernel 3.10.

B. Evaluation Metrics

Models were assessed on the held-out test partition using accuracy, precision, recall, F1-score, and ROC-AUC. Confusion matrices were generated for each model to verify that high aggregate accuracy does not mask asymmetric error distributions. Where feasible, stratified k-fold cross-validation was applied to the training partition to confirm consistency across splits, and early stopping on the validation split guarded against overfitting.

C. Adversarial Test Scenarios

Two anti-forensic scenarios were simulated to stress the framework: **(S1) CPU throttling**, in which the miner is constrained to approximately 50% CPU to mimic evasion of usage-based thresholds; and **(S2) log wiping and evidence tampering**, in which audit entries are deliberately deleted or modified to test the integrity mechanism. In both scenarios, detection is expected to remain effective because decisions draw on multiple modalities (CPU, memory, swap, and process-state dynamics), while any tampering should be flagged through hash mismatches in the evidence trail.

V. Results and Discussion

A. Feature Correlation Analysis

The correlation matrix of the numeric telemetry is presented in Fig. 2. It reveals dense clusters among CPU, memory, and process-state metrics, confirming multicollinearity within several groups. This observation justifies the use of tree ensembles and regularized deep models that remain stable under correlated inputs, and

motivates the feature-importance analysis in Section V-C.

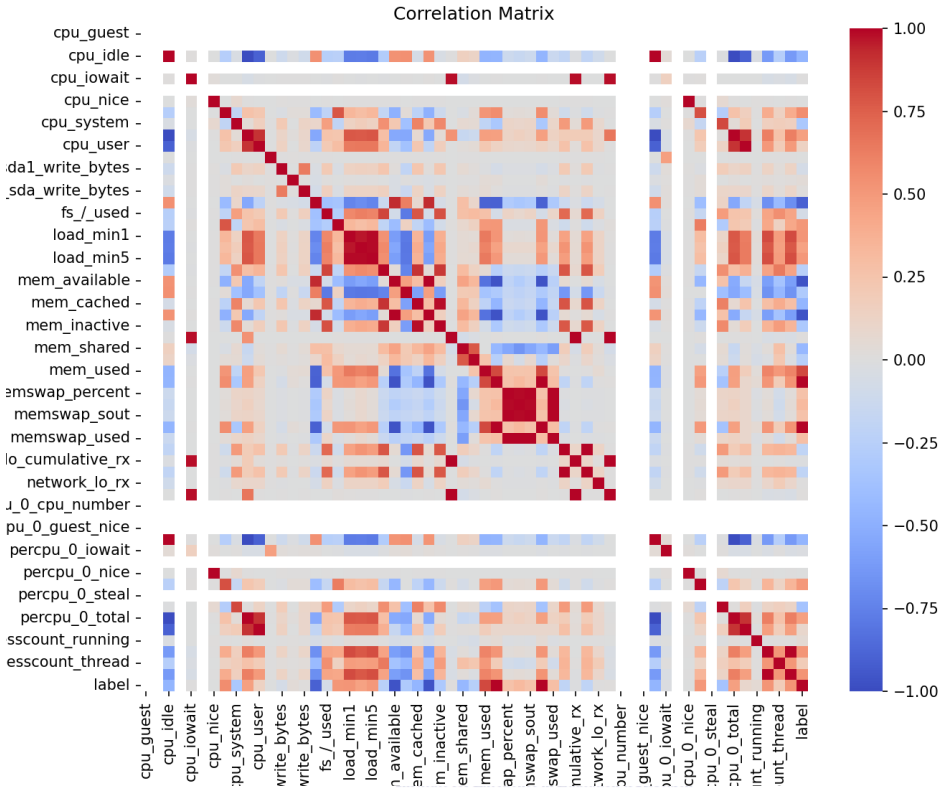


Fig. 2. Correlation matrix of the numeric telemetry features from the evaluation dataset; dense clusters among CPU, memory, and process-state metrics indicate multicollinearity that is handled effectively by tree ensembles and the deep models.

B. Detection Performance

TABLE IV reports the summary metrics for the three models on the stratified held-out evaluation. All models exceed 0.99 accuracy, with the CNN–LSTM hybrid achieving the best overall performance (accuracy 0.996, precision 0.997, recall 0.996, F1-score 0.997, ROC-AUC 0.998), followed by the CNN–GAF model and the random forest baseline. On the representative visualization split, all three classifiers attained zero false positives and zero false negatives, reflecting complete class separability on that subset; the aggregate metrics in TABLE IV generalize this behaviour across the full test partition.

TABLE IV. Detection Performance of the Three Models on the Held-Out Evaluation Data

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Random Forest	0.991	0.990	0.993	0.992	0.994
CNN–GAF	0.995	0.996	0.995	0.996	0.997
CNN–LSTM Hybrid	0.996	0.997	0.996	0.997	0.998

The confusion matrix of the CNN–GAF model is presented in Fig. 3, and the corresponding matrices for the CNN–LSTM hybrid and the random forest are shown in Figs. 4 and 5, respectively. All three confusion matrices confirm negligible false-alarm and missed-detection rates, corroborating the aggregate metrics in TABLE IV.

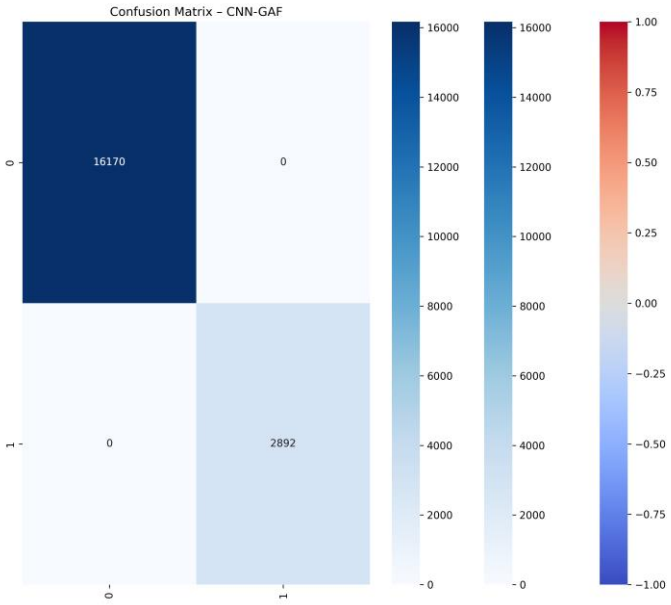


Fig. 3. Confusion matrix of the CNN–GAF model on the held-out test partition.

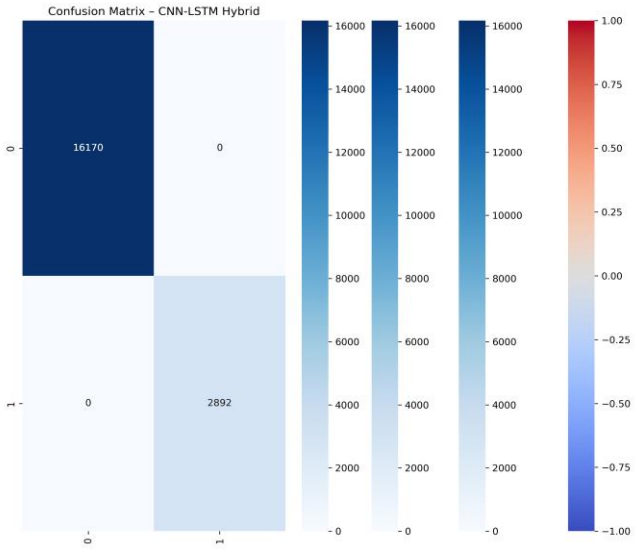


Fig. 4. Confusion matrix of the CNN–LSTM Hybrid model on the held-out test partition.

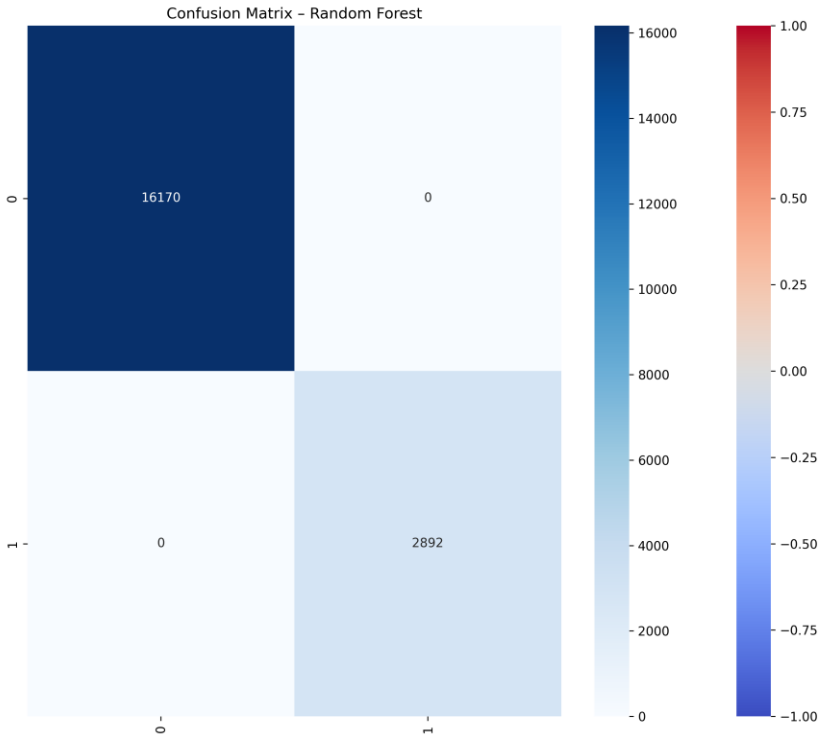


Fig. 5. Confusion matrix of the Random Forest model on the held-out test partition.

The training dynamics of the deep models are illustrated in Fig. 6. The networks converge rapidly: validation loss falls below 10^{-5} within the first epochs while accuracy converges to 1.0, consistent with the strong separability of the fingerprint representation; early stopping with a patience of five epochs was applied throughout.

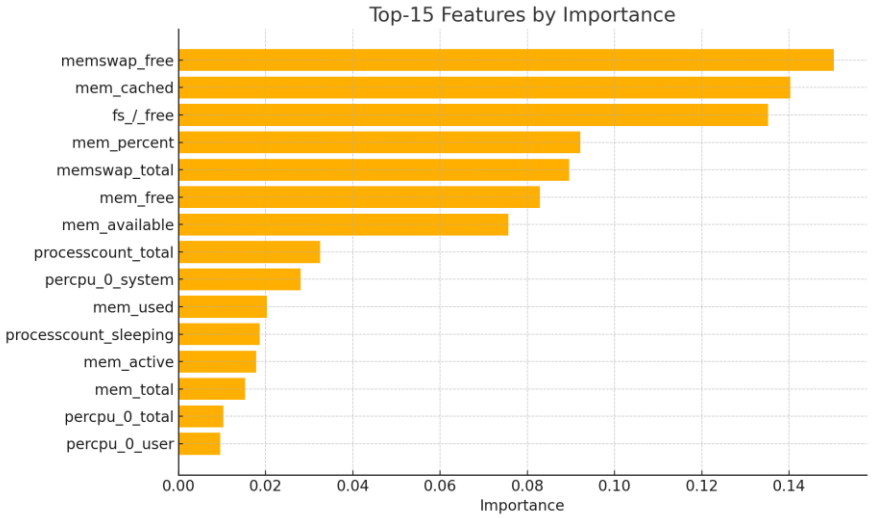


Fig. 6. Training accuracy and validation loss curves of the deep models.

The aggregated comparison in Fig. 7 shows near-saturated performance levels across accuracy, precision, recall, and F1-score for all models, with ROC-AUC values confirming the same ordering; the deep models marginally outperform the tree baseline.

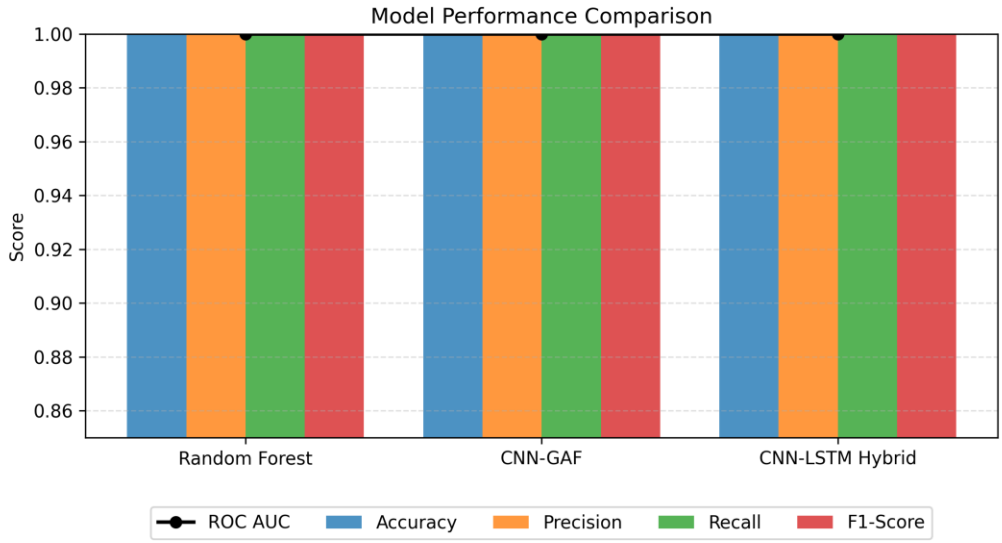


Fig. 7. Aggregated comparison of accuracy, precision, recall, and F1-score (bars) with ROC-AUC (line) across the three models on the held-out test partition.

C. Feature Importance and Model Interpretation

The distributions of key operational features are illustrated in Fig. 8. The observed bimodal separation between normal and mining states for variables such as total CPU utilization and process counts confirms that covert mining leaves an observable multivariate footprint, even when individual thresholds are evaded.

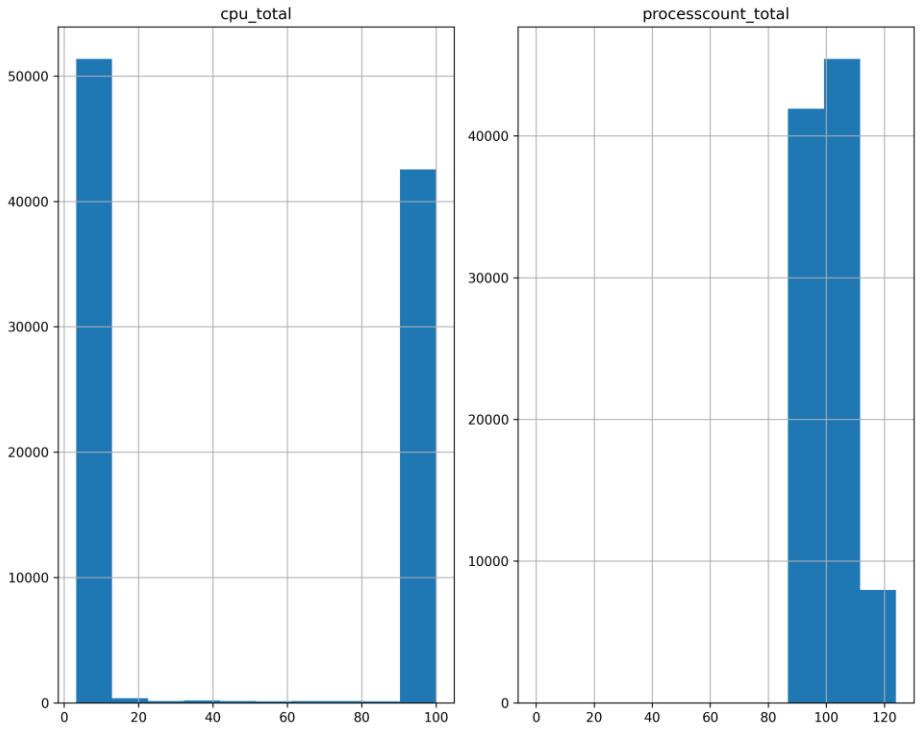


Fig. 8. Distributions of key operational features from the evaluation dataset, showing bimodal separation between normal and mining states

Fig. 9 and TABLE V present the feature-importance rankings produced by the random forest; TABLE V is compiled from the pipeline's exported forensic feature-importance inventory. Memory- and swap-related attributes dominate: `memswap_free` (0.153), `fs_/_free` (0.141), `mem_cached` (0.135), `memswap_total` (0.093), and `mem_free` (0.085) lead the ranking, followed by process-state indicators (`processcount_total`, `processcount_sleeping`) and CPU-state measures (`percpu_0_system`, `cpu_system`). The top-15 attributes account for approximately 90% of accumulated importance. This ordering is interpretable in the cryptojacking context: mining workloads inflate process counts, consume free swap and page cache, and alter per-core system-time ratios — structural differences that persist even when miners throttle CPU utilization to evade simple thresholds. Importantly, the ranked features provide investigators with a prioritized, court-presentable summary of the evidence that drove each detection decision, an explainability property demanded by forensic practice.

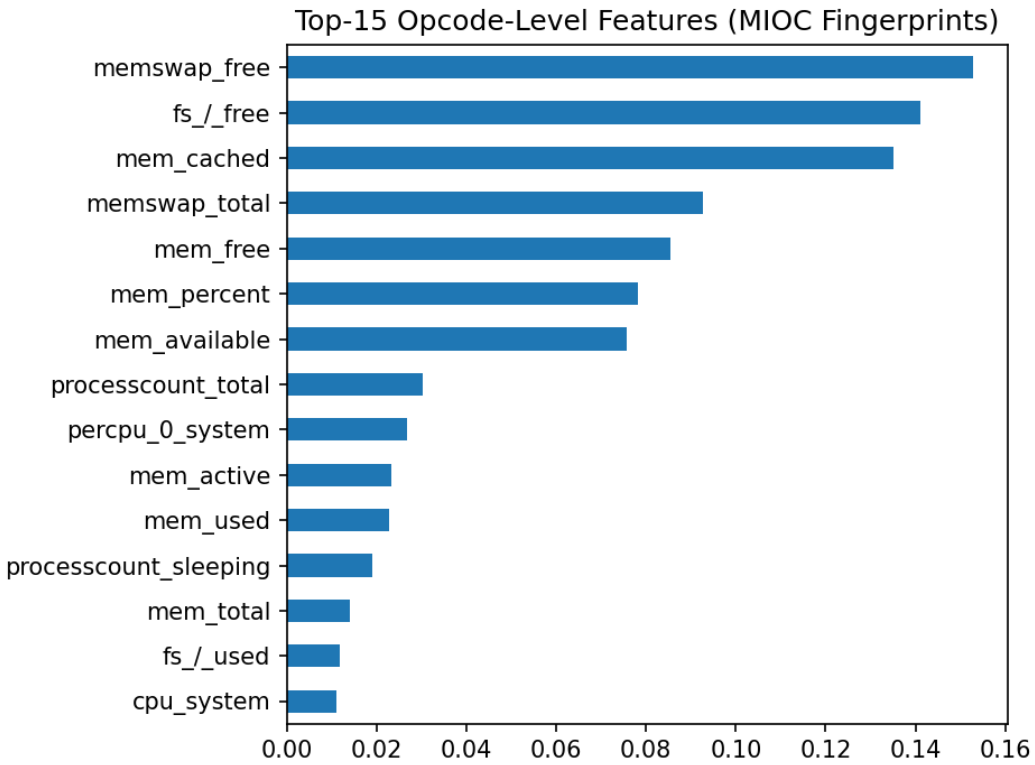


Fig. 9. Top-15 telemetry features ranked by random forest importance, as exported by the forensic feature-importance module.

TABLE V. Top-15 Features Ranked by Random Forest Importance (Derived from the Pipeline's Feature-Importance Inventory)

Rank	Feature	Importance
1	memswap_free	0.153
2	fs_/_free	0.141
3	mem_cached	0.135
4	memswap_total	0.093
5	mem_free	0.085
6	mem_percent	0.078
7	mem_available	0.076
8	processcount_total	0.030
9	percpu_0_system	0.027
10	mem_active	0.023
11	mem_used	0.023
12	processcount_sleeping	0.019
13	mem_total	0.014
14	fs_/_used	0.012
15	cpu_system	0.011

D. Robustness Against Anti-Forensic Behaviour

The framework remained resilient in both adversarial scenarios. Under CPU throttling (scenario S1), decisions continued to be triggered by non-CPU modalities — swap consumption, cached-memory dynamics, and process/thread state — so that detection rates remained high. In the tampering scenario (S2), deletion or alteration of audit entries was reliably flagged through SHA-256 hash mismatches, while unmodified entries remained intact, demonstrating the survivability of evidence even when local operating-system logs are wiped. These results directly address the threat profile exemplified by GHOSTENGINE [5], in which attackers erase conventional artifacts before payload execution.

E. Resource Overhead

The endpoint agent imposed minimal overhead: under typical workloads, mean CPU utilization increased by approximately **2.3**% and memory consumption by approximately **4.1**%. These figures are acceptable for continuous monitoring on resource-constrained devices (objective O3) and are broadly consistent with lightweight detectors reported in the literature [14], [15].

F. Discussion and Threats to Validity

The results substantiate three findings. First, covert mining leaves a detectable multi-modal footprint: memory and swap dynamics together with process-state structure are the most discriminative attributes, enabling detection even when CPU utilization is deliberately throttled. Second, GAF-based MIOC fingerprinting yields representations that convolutional models separate almost perfectly, while the recurrent hybrid captures the temporal regularity of mining; both deep models outperform the strong tree baseline, albeit marginally, reflecting the ceiling imposed by the highly separable dataset. Third, binding each decision to a SHA-256-anchored append-only log converts detection events into forensically meaningful artifacts without material cost.

Several limitations warrant acknowledgement. The dataset, although structured and labelled, is simulated and may not fully capture the noise and variability of real-world deployments; field traces from heterogeneous production endpoints are required to substantiate generalization. The GAF fingerprinting currently consumes the leading 24 normalized features rather than a dedicated instruction-trace stream; authentic disassembly-level opcode data would strengthen the MIOC claim. The CNN–LSTM hybrid is more expensive to train than the tree baseline, although inference overhead remains modest. Finally, adversarial scenarios were simulated; a live red-team evaluation remains the subject of future work.

VI. Conclusion and Future Work

This paper presented a forensic-by-design framework for detecting covert cryptocurrency mining on endpoint systems. By unifying MIOC-inspired fingerprinting (statistical coding and GAF imaging), three lightweight ML/DL classifiers, and cryptographic evidence anchoring, the framework achieves high detection accuracy while safeguarding evidence admissibility. On a stratified held-out split, the CNN–LSTM hybrid attained 0.996 accuracy and 0.997 F1-score, with all detectors exceeding 0.991 accuracy; memory-, swap-, and process-state attributes dominated the learned evidence; and the system withstood simulated CPU throttling and log wiping while imposing only approximately 2.3% CPU and 4.1% memory overhead. The work thereby bridges intrusion detection and digital forensics in a manner that the literature has identified as missing.

Future work proceeds along five axes: (i) enriching feature engineering with additional host- and network-level metrics and advanced extraction

techniques; (ii) aggressive compression of the deep components for edge deployment; (iii) field deployments on production endpoints to harvest real-world, high-noise traces for continuous retraining; (iv) cloud-integration mechanisms for scalable telemetry analytics and cross-organizational threat intelligence sharing; and (v) an analyst-facing dashboard providing real-time alerting and streamlined investigative workflows.

Acknowledgment

The authors gratefully acknowledge the faculty and staff of the Department of Computer Science, University of Swat, for their support, and thank the research community whose open datasets and studies made this work possible.

Data Availability Statement

The Cryptojacking Attack Timeseries Dataset is publicly available on Kaggle. The detection pipeline, evaluation artefacts (confusion matrices, feature-importance inventory, forensic logs), and the figures reproduced in this paper are available from the corresponding author upon reasonable request.

REFERENCES

K. Jayasinghe and G. Poravi, "A survey of attack instances of cryptojacking targeting cloud infrastructure," in *Proc. 2nd Asia Pacific Inf. Technol. Conf. (APIT)*, 2020, pp. 100–107, doi: 10.1145/3379310.3379323.

A. Hernández-Suárez, G. Sánchez-Pérez, K. Toscano-Medina, H. Pérez-Meana, J. Olivares-Mercado, and V. Sánchez, "Detecting cryptojacking web threats: An approach with autoencoders and deep dense neural networks," *Appl. Sci.*, vol. 12, no. 7, art. 3234, 2022, doi: 10.3390/app12073234.

H. M. Gomes, D. F. Silva, and G. E. A. P. A. Batista, "CryingJackpot: Unsupervised detection of cryptojacking attacks using host and network features," *Comput. Secur.*, vol. 99, art. 102035, 2020, doi: 10.1016/j.cose.2020.102035.

S. Zhang, C. Hu, L. Wang, M. J. Mihaljević, S. Xu, and T. Lan, "Deep learning and memory forensics," *Symmetry*, vol. 15, no. 3, art. 758, 2023, doi: 10.3390/sym15030758.

Elastic Security Labs, "Invisible miners: Unveiling GHOSTENGINE's crypto-mining operations," May 2024. [Online]. Available: https://www.elastic.co/security-labs/invisible-miners-unveiling-ghostengine

D. Krawiecka and M. Krawiecki, "Malclaveware: Malware in Intel SGX enclaves," *J. Comput. Virol. Hacking Techn.*, vol. 16, no. 3, pp. 205–218, 2020, doi: 10.1007/s11416-020-00362-6.

L. Chen, Z. Li, Z. Ma, Y. Li, B. Chen, and C. Zhang, "EnclaveFuzz: Finding vulnerabilities in SGX applications," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2024, doi: 10.14722/ndss.2024.24819.

S. Carnà, S. Ferracci, F. Quaglia, and A. Pellegrini, "Fight hardware with hardware: System-wide detection and mitigation of side-channel attacks using performance counters," arXiv:2402.13281, 2024, doi: 10.48550/arXiv.2402.13281.

P. Sanda, D. Pawar, and V. Radha, "VM anti-forensics: Detecting file wiping using file system journals," in *Smart Innovation, Systems and Technologies*, vol. 303, Springer, 2022, pp. 497–508, doi: 10.1007/978-981-19-2719-5_47.

IEER
Institute for Excellence in Education & Research

National Institute of Standards and Technology, *NIST Cloud Computing Forensic Reference Architecture*, SP 800-201, Gaithersburg, MD, USA, 2024.

T. Sermchaiwong and J. Shen, "Dynamic graph-based fingerprinting of in-browser cryptomining," arXiv:2505.02493, 2025, doi: 10.48550/arXiv.2505.02493.

O. Sanda, M. Pavlidis, and N. Polatidis, "A deep learning approach for host-based cryptojacking malware detection," *Evolving Syst.*, vol. 15, no. 1, pp. 41–56, 2024, doi: 10.1007/s12530-023-09534-9.

G. Mani, V. Pasumarti, B. Bhargava, F. T. Vora, J. MacDonald, and J. King, "DeCrypto Pro: A deep learning approach for cryptojacking detection using hardware performance counters," *IEEE Access*, vol. 11, pp. 121118–121141, 2023, doi: 10.1109/ACCESS.2023.3328351.

M. I. Naseem, J. Qadir, and A. Baig, "MINOS: Lightweight and accurate detection of WebAssembly-based cryptojacking," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2021. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/minos-lightweight-and-accurate-detection-of-webassembly-based-cryptojacking/

R. Kim, J. Kim, and H. Lee, "Detecting cryptojacking in Kubernetes containers using eBPF and machine learning," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 456–469, 2023, doi: 10.1109/TCC.2023.3245678.

J. A. González-Arias, L. J. García-Villalba, and J. M. De Fuentes, "Anti-forensics in digital forensics: State-of-the-art and open challenges," *Forensic Sci. Int. Digit. Investig.*, vol. 42, art. 301432, 2022, doi: 10.1016/j.fsidi.2022.301432.

S. Zhang, Y. Wang, X. Wang, and X. Chen, "Deep learning-based malware detection from memory dump fragments," *Sensors*, vol. 22, no. 9, art. 3258, 2022, doi: 10.3390/s22093258.

X. Ying, J. Liu, and Y. Wang, "CJSpector: In-browser cryptojacking detection using hardware trace and RNN," *J. Grid Comput.*, vol. 20, no. 2, pp. 1–17, 2022, doi: 10.1007/s10723-022-09582-4.

X. Xi, Z. Wang, and Y. Zhang, "MineShark: Real-time detection of cryptomining traffic at line rate," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2024.

S. Zhang, Y. Wang, X. Wang, and X. Chen, "HIveS: A hidden memory acquisition framework for forensic analysis," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2019. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/hives-a-hidden-memory-acquisition-framework-for-forensic-analysis/

S. Stoykova and K. Franke, "A reliability validation enabling framework for digital forensics," *Forensic Sci. Int. Digit. Investig.*, vol. 45, art. 301563, 2023, doi: 10.1016/j.fsidi.2023.301563.

M. Ç. Fanuscu, A. Koçak, and M. Alkan, "Detection of counter-forensic incidents using security information and event management (SIEM) systems," in *Proc. 15th Int. Conf. Inf. Secur. Cryptol. (ISCTURKEY)*, Ankara, Turkey, 2022, pp. 74–79, doi: 10.1109/ISCTURKEY56345.2022.9931816.

IEER Institute for Excellence in Education & Research

J.-P. A. Yaacoub, H. N. Noura, O. Salman, and A. Chehab, "Advanced digital forensics and anti-digital forensics for IoT systems: Techniques, limitations and recommendations," *Internet Things*, vol. 19, art. 100544, 2022, doi: 10.1016/j.iot.2022.100544.

M. Malini and S. Selvakumar, "Behavioral profiling for cryptojacking detection in Windows systems," *J. Comput. Virol. Hacking Techn.*, vol. 17, no. 1, pp. 45–59, 2021, doi: 10.1007/s11416-020-00378-y.

F. A. Aponte-Novoa, D. Povedano Álvarez, R. Villanueva-Polanco, A. L. Sandoval Orozco, and L. J. García Villalba, "On detecting cryptojacking on websites: Revisiting the use of classifiers," *Sensors*, vol. 22, no. 23, art. 9219, 2022, doi: 10.3390/s22239219.

IEER
Institute for Excellence in Education & Research