

DroidRan: hardware behavior dataset for multi-stage android ransomware detection

Kalaivani Selvaraj, Manmeet Mahinderjit Singh

School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia

Article Info

Article history:

Received Nov 30, 2025

Revised Aug 25, 2026

Accepted Sep 4, 2026

Keywords:

Android ransomware

Attack lifecycle

Benchmark dataset

Hardware telemetry

MITRE ATT&CK

Ransomware detection

ABSTRACT

Android ransomware increasingly exploits unpatched kernel vulnerabilities, still conventional detection methods rely on static application features, system logs, or network flows that polymorphic variants readily obfuscate. Hardware-level behavioral telemetry offers a tamper-resistant alternative, but no public dataset links such telemetry to the ransomware lifecycle. This paper presents DroidRan, a benchmark dataset of 22 dynamic hardware-behavior features spanning six resource categories, collected from an Android device carrying the unpatched CVE-2024-36971 vulnerability while 12 ransomware families were executed through three squatting attack vectors. Feature variations are mapped to four MITRE ATT&CK mobile stages: initial access (IA), privilege escalation (PE), command and control (CC), and impact (Im). A gated recurrent unit-long short-term memory (GRU-LSTM) framework combining parallel stage-specific branches, a cluster-based graph Bayesian network with correlation attention, and a dynamic threat intelligence module is proposed to exploit these traces. The Friedman test confirms that all 22 features discriminate benign from ransomware behavior, and cross-dataset benchmarking against CICMaldroid2017, CICAndMal2020, and HelDroid with Wilcoxon post-hoc analysis ranks DroidRan first for both accuracy and F1-score. The proposed framework outperforms four recent deep learning baselines on DroidRan, supporting stage-aware and obfuscation-resilient ransomware detection.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Manmeet Mahinderjit Singh

School of Computer Sciences, Universiti Sains Malaysia

Gelugor 11800, Penang, Malaysia

Email: manmeet@usm.my

1. INTRODUCTION

Mobile computing has expanded rapidly, and Android-based smart devices are now embedded in healthcare, finance, and critical infrastructure. This ubiquity has increased the attack surface for cybercriminals and ransomware is currently the most economically harmful type of malware against platform [1]. Ransom demands in the early days of Android ransomware were just to unlock the screen of the device, but modern ransomware variants use robust cryptography to encrypt user contents and require ransom in well-known but untraceable cryptocurrencies, resulting in significant economic and information losses [2], [3]. In 2026, ransomware attacks worldwide increased by more than 50% compared with the levels reported in 2024, with mobile endpoints becoming an increasingly significant target within the attack landscape [1].

Most of deployed countermeasures rely on analyzing software artefacts: either statically inspecting application structure and manifests, or dynamically inspecting API calls and network traffic [4]. These methods work well if they are applied to known families and will fail badly if they are applied to new variants [5]. There are polymorphic transformations and code obfuscation systematically change code signatures and make it

difficult to be analyzed statically [6]. Dynamic analyses are more resilient but require a controlled sandbox and are limited by observation time and path coverage and can be identified and thwarted by ransomware that can detect and suppress a virtualized environment [7], [8]. The software level indicators are also directly manipulable: an author can alter the patterns of the API calls, encrypt the communication of the commands, and add anti-analysis code explicitly to evade these indicators [9]. Thus, a feature space that is more difficult to manipulate is required.

Such space exists at the hardware level with behavioral analysis. Rather than examining the application's stated purpose, it performs the measurable impact of executing the application on the resources at the device: processor utilization, memory usage, battery voltage and temperature, network throughput, graphics activity, and clock frequency [10], [11]. The amount of these quantities is far more difficult for a malware author to obfuscate than software quantities, as they are a consequence of the computation and not a declaration of it [12]. Each of these resource usage perturbations (file encryption, data exfiltration, and C2 communication) can be measured and are not seen in benign application use and are present under software level obfuscation which makes them appealing for distinguishing zero-day and polymorphic variants [13], [14]. The lack in the research community is not the idea, but the data: there is no public Android corpus that has hardware telemetry systematically collected throughout attack phases and ransomware families.

The field has come a long way with the establishment of Android malware and ransomware benchmarks like CICMaldroid2017, CICAndMal2020, and HelDroid [15]–[17]. They can expose attributes at the software layer, like permissions, API calls, network flows, and system calls, but don't include hardware telemetry of the ransomware phases. Ransomware can be broken down into the following components: initial access (IA), privilege escalation (PE), command and control (CC), and impact (Im); the MITRE ATT&CK mobile (Android) matrix [5] is a formalization of this decomposition. Variation in hardware features can be matched with these phases allowing for stage-aware detection, which is important for it allows for intervention before the damage is irretrievably lost due to encryption [14], [18]. There is no existing dataset that provides alignment.

Table 1 makes the gap concrete. Existing benchmarks are limited to three measurable dimensions. First, feature modality: they expose only software-level attributes and contribute zero hardware-telemetry features. Second, lifecycle granularity: none labels samples by ransomware stage, so the number of MITRE ATT&CK-mapped stages available for supervised training is zero. Third, evasion resilience: because their features are software-derived, they are directly susceptible to obfuscation and polymorphism. Alongside the dataset gap, current deep learning detectors share a structural limitation. DBN-GRU [19] performs unsupervised pre-training with restricted Boltzmann machines, which is computationally costly and does not weight features by importance. iCNN-LSTM+ [20] stacks convolutional and recurrent layers, introducing an information bottleneck, and selects features with the Pearson correlation coefficient, which presumes a linear feature-label relationship. MHA-LSTM+EGTO [21] couples multi-head attention with an LSTM but inherits both an expensive optimization loop and the vanishing-gradient behavior of long sequences. SA-CGRU [22] fuses self-attention with a convolutional GRU but depends on sensitive focal-loss hyperparameters. Each was designed for binary or general anomaly classification; none identifies the executing attack stage or attributes a sample to a family from hardware behavior [23], [24].

Table 1. Comparison of DroidRan with widely used Android malware and ransomware datasets

Dataset	Feature modality	Total features	Hardware features	MITRE stages labelled	Ransomware families	Obfuscation resilience
CICMaldroid2017	Application and API level	84	0	0	Mixed malware	Low
CICAndMal2020	Network flow and permissions	141	0	0	Mixed malware	Low
HelDroid	Static indicators	35	0	0	Ransomware	Low
CICIoMT2024	Network and flow level	45	0	0	IoMT malware	Low
DroidRan (this work)	On-device hardware telemetry	22	22	4	12	High

This paper addresses both gaps. The first contribution is DroidRan, a publicly released benchmark of hardware telemetry captured from an Android device during execution of 12 ransomware families delivered through three squatting vectors, comprising 22 features drawn from six resource categories and mapped to the four MITRE ATT&CK lifecycle stages [25]. The second is a gated recurrent unit–long short-term memory (GRU-LSTM) detection framework that exploits these traces through four parallel stage-specific GRU branches, a cluster-based graph Bayesian network with correlation-weighted attention, and a dual-branch module pairing a pre-trained LSTM with a dynamic threat intelligence store for family attribution. The dataset

is validated with the Friedman test and a Bonferroni-corrected Wilcoxon signed-rank post-hoc test against three established benchmarks, and the framework is evaluated against four recent baselines. The remainder of the paper is organized as follows. Section 2 describes the experimental setup, dataset generation, and the proposed framework. Section 3 reports the statistical analysis, cross-dataset benchmarking, and comparative model evaluation. Finally, section 4 provides conclusion.

2. METHOD

This section describes the DroidRan dataset generated and operations of the proposed GRU-LSTM framework. Subsections 2.1 to 2.4 cover the experimental environment, the MITRE ATT&CK stage labelling procedure, the dataset composition, and the preprocessing pipeline. Subsection 2.5 describes the proposed GRU-LSTM detection framework, and subsection 2.6 covers the implementation configuration and evaluation criteria used throughout section 3.

2.1. Experimental setup

The environment was set up to mimic a squatting attack against an Android device that is running an older version of the kernel where the CVE-2024-36971 vulnerability has not been patched. The actual real-time setup can be seen in Figure 1. Three types of squats were performed: application-based, GUI-based, and email based, using watering-hole, cross-site scripting (XSS), and spear phishing, respectively.

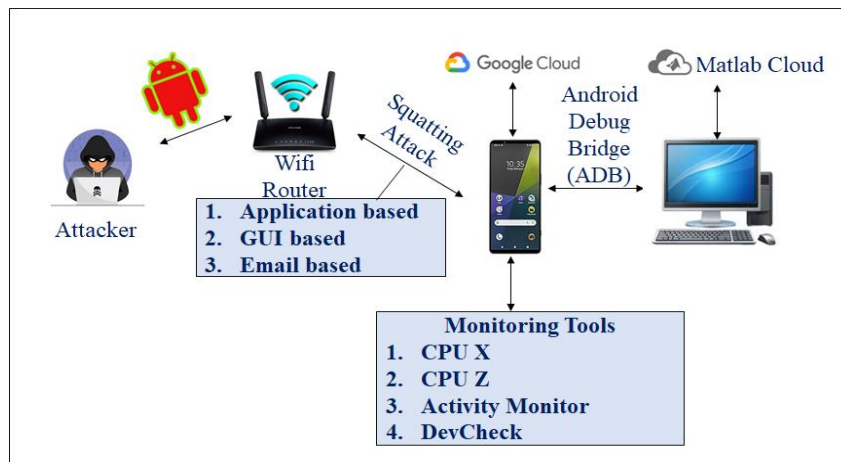


Figure 1. Real-time experimental setup for DroidRan dataset generation

The simulation environment consists of four components: ransomware APK samples, Wi-Fi router, an Android device with monitoring applications, and a MATLAB cloud workspace running in a Windows 10 workstation with an Intel Core i7 processor and 16 GB of RAM. Public repositories such as VirusTotal, MITRE, AndroZoo, and Malware Bazaar were used to collect APK samples. Twelve families were used: CovidLocker, CryCryptor, DoubleLocker, FakeApp Locker, FileCoder, Jisut, Koler, LockerGoga, RafelRAT, SimpLocker, SLocker, and SpyNote. Ransomware activity was started through the bypass of the notification alert, with the CVE-2024-36971 patch not applied. Variation in hardware features were collected using CPU-X, CPU-Z, Activity Monitor, and DevCheck and compiled into a single CSV corpus. All activities were conducted on an isolated network segment, not connected to any production infrastructure, and outbound C2 traffic was limited to this segment.

The three operational flows are summarized in Figure 2. In Figure 2(a) the attacker inflicts a malicious script on a legitimate website, and when the victim accesses the contaminated site, the session is hijacked, session cookies are stolen and the device is locked until payment is made. In Figure 2(b) the adversary compromises a web application used by the target group, so that visiting the application silently downloads and installs a Trojan APK before the session is seized. In Figure 2(c) the adversary impersonates a trusted entity and delivers a malicious link by email; following the link surrenders the victim authentication tokens. Table 2 lists the propagation behavior of each family together with the hardware resources it stresses most.

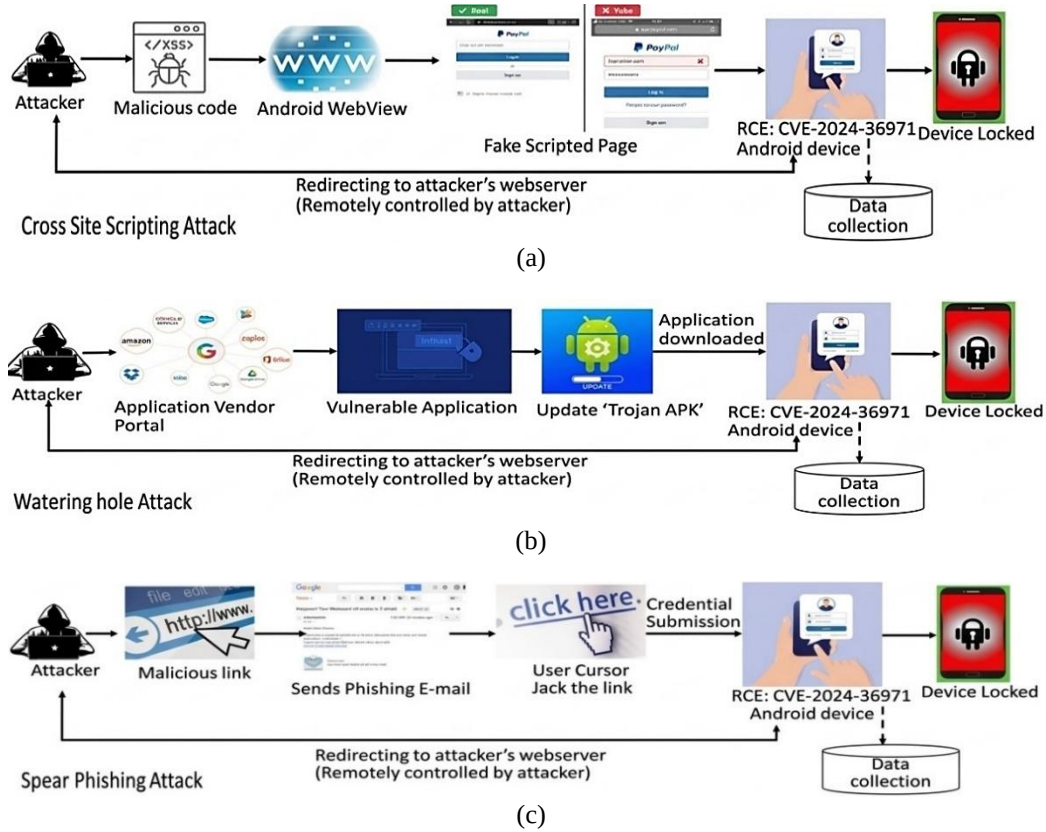


Figure 2. Squatting attack simulation; (a) cross-site scripting, (b) watering-hole, and (c) spear phishing

Table 2. Android ransomware families, propagation behavior, and dominant hardware indicators

Family	Propagation and runtime behavior	CPU	RAM	Battery	Clock	Network	GPU
CovidLock	Screen lock via fake application; UI loop drives CPU	▲	●	●	▲	○	●
CryCryptor	AES file encryption; memory spikes during processing	▲	▲	●	▲	▲	▲
DoubleLocker	Encrypts files and changes device PIN; GPU animation loop	●	▲	▲	●	●	●
FakeApp	Fake UI overlay; memory held for real-time interception	●	●	○	●	○	○
Locker	Full storage encryption; voltage rises on disk activity	▲	●	●	▲	●	▲
FileCoder.C	Spreads via third-party stores; memory load on payload run	●	○	○	○	○	○
Jisut	Browser-based ransom UI; issues malicious web requests	○	○	○	○	▲	○
Koler	Encrypts configuration and documents; CPU heats on loops	●	▲	▲	●	●	●
LockerGoga	Exfiltrates to C2 server; high network and memory use	●	●	●	●	●	▲
RafelRAT	AES on SD card contents; raises I/O latency and CPU use	▲	▲	●	▲	●	▲
SimpLocker	Fake UI animation and file locking; memory and GPU pressure	▲	●	●	▲	○	▲
SLocker	Data theft; background execution leaks memory, spikes network	●	●	●	●	●	●
SpyNote							

Note: ● very high; ▲ high; ● moderate; and ○ low.

2.2. MITRE ATT&CK stage labelling

Each telemetry window is labelled with a lifecycle stage by mapping the adversary action active during that window to the MITRE ATT&CK mobile (Android) matrix. During controlled execution, the action in progress at each instant, namely, initial compromise, permission elevation, C2 communication, or encryption and locking, was recorded and mapped to the tactic and technique identifiers in Table 3. The telemetry captured during that action inherits the corresponding stage label, and a pre-attack window is labelled benign for

reference. Across the collected traces the most frequently exercised techniques are T1660 (phishing) under IA and T1626/.001 (abuse elevation control mechanism, device administrator permissions) under PE.

Table 3. MITRE ATT&CK labelling for squatting-based attacks

Tactic and stage label	Description	Technique used	MITRE technique ID
IA, TA0027	Initial compromise of the device	Phishing	T1161, T1456, T1664, T1461, T1660
PE, TA0029	Escalating privileges to gain control	Abuse elevation control mechanism / device administrator permissions	T1626/.001, T1404
CC, TA0037	Communication with adversary servers	Remote access software	T1437, T1616, T1637/.001, T1663
Im, TA0034	Final malicious action such as encryption or destruction	Endpoint denial of service	T1640, T1662, T1471, T1641, T1642, T1582

2.3. DroidRan dataset description

Collection Ran for six months, from 1 November 2024 to 30 April 2025, between 08:00 and 18:00 daily, and occupied approximately 320 GB of storage. Each family is profiled across the four attack stages together with a pre-attack baseline. Table 4 reports the acquisition schedule and record counts. A total of 305,703 raw hardware-telemetry readings were captured, of which 48,294 correspond to benign behavior and 257,409 to ransomware activity.

Table 4. Data acquisition for each ransomware family (collection window 08:00–18:00)

Family	Collection period	Normal	IA (≤ 15 s)	PE (≤ 60 s)	CC (≤ 300 s)	Im (≤ 900 s)	Total
CovidLock	1–15 Nov 2024	703	1,753	1,015	1,257	1,034	5,762
CryCryptor	16 Nov–2 Dec 2024	956	2,382	1,378	1,707	1,405	7,828
DoubleLocker	3–18 Dec 2024	1,438	3,585	2,076	2,571	2,115	11,785
FakeApp Locker	19 Dec 2024–2 Jan 2025	1,074	2,225	1,365	1,753	1,359	7,776
FileCoder.C	3–19 Jan 2025	950	2,280	1,342	1,674	1,447	7,693
Jisut	20 Jan–2 Feb 2025	547	1,365	790	979	806	4,487
Koler	3–18 Feb 2025	351	874	505	627	516	2,873
LockerGoga	19 Feb–6 Mar 2025	936	2,143	1,386	1,723	1,425	7,613
RafelRAT	7–21 Mar 2025	1,003	2,280	1,412	1,776	1,144	7,615
SimpLocker	22 Mar–4 Apr 2025	950	2,225	1,259	1,792	1,455	7,681
SLocker	5–17 Apr 2025	1,438	3,393	1,924	2,571	2,305	11,631
SpyNote	18–29 Apr 2025	1,327	3,585	2,076	2,645	2,115	11,748

The dataset comprises 22 hardware features organized as sub-features of six resource categories. Table 5 lists them. These categories were selected because they are the resources most directly stressed across the ransomware lifecycle and, unlike software-level indicators, cannot be trivially masked by obfuscation or anti-analysis logic. Encryption and screen-locking loops directly reflect in the hardware resources of drive clock, thermal, and GPU frequency variation. In-memory payload staging and interception overlays appear in RAM status, communication and exfiltration appear in network throughput and sustained disk activity manifests as battery voltage and temperature shift. All 22 quantities are exposed through standard non-rooted Android diagnostic interfaces, so collection reproduces on commodity devices. Two candidate families of metrics were deliberately excluded: per-application software counters (API and system call traces), because these are precisely the signals polymorphic ransomware evades, and low-level hardware performance counters, because they require root privilege or vendor-specific kernel access and are therefore not portable.

Table 5. DroidRan feature description

No.	Resource category	Sub-features	Data type	Unit
1	RAM status	Total RAM; free RAM; available internal memory; total internal memory; available storage	Float, integer	GB, MB
2	Network traffic	Received; transmitted	Float	KB/s
3	Battery performance	Voltage; temperature	Integer, float	mV, °C
4	CPU utilization	Thermal sensor 0; thermal sensor 1; thermal sensor 2	Float	°C
5	GPU usage	Temperature; frequency	Float, integer	°C, MHz
6	Clock speed	Clock frequency of CPU cores 0 to 7 (eight sub-features)	Float	GHz

Raw records were preprocessed into a normalized, class-balanced corpus of 1,480 baseline samples and 5,920 post-attack samples per family (1,480 per stage), giving 7,400 samples per family and 88,800 in total. Samples are organized into family-specific directories with stage-wise separation to support reproducible experimentation on stage-aware detection, multi-class classification, and behavioral drift analysis.

2.4. Sampling and preprocessing

All telemetry was acquired at a fixed one-second interval applied uniformly to every feature and every stage; only the collection duration varied by stage, as reported in Table 4. The one-second rate was chosen for three reasons. It provides sufficient temporal granularity to capture the transient anomalies of the time-critical early stages, such as processor and memory surges during payload injection, which intervals of 5 s or 10 s would miss. The monitoring tools have a one-second refresh natively through the Android Debug Bridge, allowing reproducible acquisition. The sampling rates of 100 and 500 milliseconds were considered and discarded, since the overhead of polling the hardware would interfere with the hardware's behavior. The four-step pipeline was used to remove noise from the raw sensor readings due to background processes, transient operating-system events, and momentary dropouts.

The steps undertaken were first cleaning: missing and dropout entries (maximum 0.5% of the data) were recovered and missing data by linear interpolation of immediate temporal neighbours, and rows with persistent sensor failure or physically implausible data were excluded. Second denoising: each series was decomposed using discrete wavelet transform at three levels, soft thresholded with universal threshold, and reconstructed. The wavelet transform was preferred to a moving average because it suppresses jitter without smearing the sharp transients that characterize encryption bursts. Third, outlier treatment and scaling: residual outliers were identified per feature with the interquartile range rule and winsorized rather than discarded, so that genuine ransomware-induced spikes were retained, and values were min-max normalized to [0, 1]. Fourth, class balancing across stages produced the final 88,800-sample corpus. For the cross-dataset study only, principal component analysis was applied to the external datasets, retaining the components explaining at least 95% of variance, so that dimensionality is comparable across corpora while DroidRan retains all 22 features.

2.5. Proposed gated recurrent unit–long short-term memory framework

The framework pursues three objectives simultaneously: identify the active MITRE ATT&CK stage, determine the hardware feature sequence order characteristic of that stage, and attribute the sample to a ransomware family. It operates in three stages, shown in Figure 3: dataset segmentation with parallel GRU processing; a cluster-based graph Bayesian network with correlation-weighted attention; and dual-branch processing combining a pre-trained LSTM with a dynamic threat intelligence store.

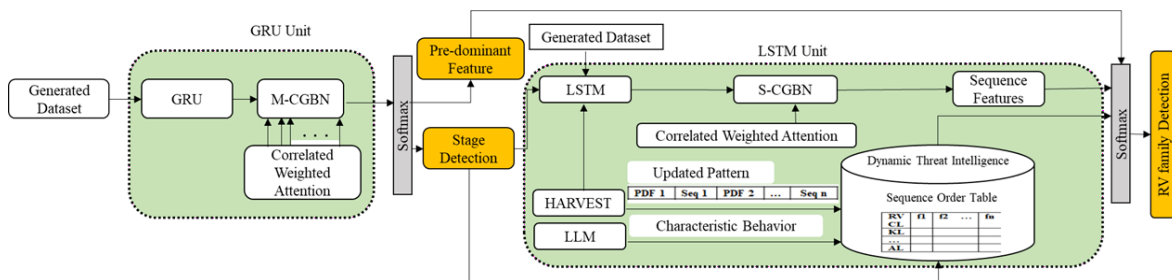


Figure 3. Proposed GRU-LSTM framework for ransomware stage detection and family classification

2.5.1. Dataset segmentation and parallel gated recurrent unit processing

Each generated datasheet is segmented into four sub-datasheets aligned with the MITRE ATT&CK stages, and the sub-datasheets are processed by four parallel GRU branches, one per stage. Within each branch, feature variation is analyzed against a predefined threshold. Conventional selection retains a feature purely by counting threshold crossings, which discards information carried by variation that never crosses the boundary. Each branch therefore additionally captures within-state fluctuation, that is, variation occurring inside a state, and admits it as evidence during selection. These preserves feature information that threshold-only selection would lose and that can be decisive for distinguishing low-footprint variants.

2.5.2. Cluster-based graph Bayesian network with correlation-weighted attention

Features selected by each branch pass to a cluster-based graph Bayesian network in which a dual-layer graph is constructed: an upper layer from features above the threshold and a lower layer from features

below it. Each selected feature is a node, every node connects to all remaining features, and network formation is carried out separately within four clusters, each mapped to one attack stage. Edge weights are then updated from the pairwise correlation between the features they join, under a correlation attention rule: an edge whose correlation difference lies close to the threshold is assigned unit weight and incremented for each successive difference near the threshold; a difference above the threshold adds 0.5; and a difference below it subtracts 0.25. The resulting weights are passed to a SoftMax. The connecting link with maximum edge weight is identified, and of the two nodes it joins, the one with the greater feature weight is taken as the predominant feature. Because each cluster corresponds to one stage, the SoftMax output identifies the active MITRE ATT&CK stage.

2.5.3. Dual-branch processing

The detected stage is passed to two parallel branches. Branch 1 uses an LSTM pre-trained with the hardware-aware ransomware variant evaluation and stage-wise tracking (HARVEST) framework [13], which profiles variant behavior through RAM status, network traffic, battery performance, processor utilization, graphics activity, and clock variation, and comprises a tunable Q-factor wavelet transform for preprocessing, Ran neighborhood rough set theory for feature selection, and graph Bayesian networks for feature optimization. HARVEST captures fluctuation across oscillations and yields a behavioral pattern per variant, ordered as a sequence that reflects the characteristic progression of each attack phase. Given the stage from the SoftMax and the generated datasheet as a secondary input, the pre-trained LSTM retains only the feature vectors associated with the detected stage. These filtered features pass through a graph Bayesian network analogous to the GRU module, differing in that only the selected stage contributes to network formation rather than all stages in cluster form. The same correlation attention rule computes edge weights, and features are then ordered by descending edge weight to produce the feature sequence order.

Branch 2 keeps dynamic threat intelligence where it stored hardware behavior profiles for the 12 known families. It provides the stage-specific feature weights and sequence orders for all 12 families are assembled into a sequence table with separate entries for weights and orders. That table and the sequence order produced by Branch 1 are compared through a SoftMax; where the match exceeds 50%, the sample is attributed to the corresponding family.

2.6. Implementation details and evaluation criteria

All experiments use 10-fold stratified cross-validation, preserving per-class and per-stage proportions in every fold. Within each training fold, an inner 90/10 split governs hyperparameter selection and early stopping, so reported metrics are never tuned on the evaluation fold, and splits are performed at family and stage level to prevent leakage of adjacent time-series windows between partitions. The framework and all baselines were implemented in Python with TensorFlow/Keras and trained on the workstation described in subsection 2.1 with GPU acceleration. Each parallel GRU branch uses 64 hidden units and the pre-trained LSTM branch uses 128. Models are optimized with Adam at an initial learning rate of 1×10^{-3} under cosine decay, with $\beta_1=0.9$, $\beta_2=0.999$, batch size 64, dropout 0.3, and a maximum of 100 epochs with early stopping at a patience of 10 on inner-validation loss. Categorical cross-entropy is used for stage and family classification, the correlation-attention threshold is the per-cluster median edge weight, and the family-match threshold is 50% sequence-order similarity. All random seeds are fixed.

Because the classification problem is multi-class and imbalanced, accuracy alone is an insufficient summary. Performance is therefore reported as weighted-average precision, recall, F1-score, and accuracy, together with false positive rate (FPR), false negative rate (FNR), specificity, and the area under the receiver operating characteristic curve (AUC-ROC). Writing TP, TN, FP, and FN for the true positive, true negative, false positive, and false negative counts of the confusion matrix, the metrics are defined in (1) to (7):

$$\text{Precision} = TP / (TP + FP) \quad (1)$$

$$\text{Recall} = TP / (TP + FN) \quad (2)$$

$$\text{F1-score} = 2(\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (3)$$

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (4)$$

$$\text{FPR} = FP / (FP + TN) \quad (5)$$

$$\text{FNR} = FN / (FN + TP) \quad (6)$$

$$\text{Specificity} = TN / (TN + FP) \quad (7)$$

Statistical validation uses the Friedman test, a non-parametric alternative to repeated-measures analysis of variance that operates on ranks rather than raw values and requires one scalar performance value per group per fold, satisfied here by the weighted F1-score and accuracy. Where the Friedman test rejects the null hypothesis of equal performance at $p < 0.05$, the Wilcoxon signed-rank post-hoc test is applied to pairwise comparisons with a Bonferroni adjustment controlling the family-wise error rate.

3. RESULTS AND DISCUSSION

This section performs two complementary analyses of the dataset and framework. Subsections 3.1 and 3.2 evaluate the statistical constitution of DroidRan and its benchmarking ability compared to CICMaldroid2017, CICAndMal2020, and HelDroid. Subsections 3.3 and 3.4 provide a comparison of the proposed model with four recent deep learning models on DroidRan, and a per-family analysis of behaviour. The implications and limitations of deployment are discussed in subsection 3.5.

3.1. Statistical significance of dataset features

Each hardware feature was evaluated prior to the model training to determine its ability to differentiate between benign and ransomware behavior. The null hypothesis is that a feature is not different between the two behavior groups, the alternative is that a feature is different significantly. All 22 features are reported in Table 6.

Each of the 22 features is significant at <0.05 level, and most have p-values below the reporting resolution, meaning that the benign and ransomware induced behavior varies significantly on each of the measured quantities. The most discriminative telemetry is the set of processor clock features (f15–f22, $\chi^2 > 2974$), available storage (f4, $\chi^2 = 5948$), and GPU frequency (f14, $\chi^2 = 3756.937$).

Table 6. Friedman test results for feature significance between normal and ransomware behavior

ID	Feature	Category	χ^2	p-value	ID	Feature	Category	χ^2	p-value
f1	Total RAM	RAM	1767.901	<0.001	f12	Thermal sensor 2	Thermal	1494.813	<0.001
f2	Free RAM	RAM	824.267	4.25×10^{-177}	f13	GPU temperature	GPU	1498.652	<0.001
f3	Available internal memory	RAM	207.72	8.22×10^{-44}	f14	GPU frequency	GPU	3756.937	<0.001
f4	Available storage	RAM	5948	<0.001	f15	CPU 0 clock	Clock	2974.58	<0.001
f5	Total internal memory	RAM	5948	<0.001	f16	CPU 1 clock	Clock	2975.017	<0.001
f6	Received	Network	300.188	9.87×10^{-64}	f17	CPU 2 clock	Clock	2976.921	<0.001
f7	Transmitted	Network	2976.23	<0.001	f18	CPU 3 clock	Clock	2974.652	<0.001
f8	Battery voltage	Battery	591.459	1.09×10^{-126}	f19	CPU 4 clock	Clock	2974.399	<0.001
f9	Battery temperature	Battery	1870.238	<0.001	f20	CPU 5 clock	Clock	2977.718	<0.001
f10	Thermal sensor 0	Thermal	1467.3	<0.001	f21	CPU 6 clock	Clock	2975.536	<0.001
f11	Thermal sensor 1	Thermal	1488.281	<0.001	f22	CPU 7 clock	Clock	2974.175	<0.001

3.2. Cross-dataset benchmarking

To get a fair comparison of DroidRan, the proposed framework was trained and evaluated independently on each of four datasets following the same 10-fold stratified cross validation with 15 significant features which was selected uniformly across the external datasets, while in the case of DroidRan all the 22 features were retained. The result is given in Table 7.

Table 7. Detection performance of the proposed GRU-LSTM model across four datasets

Dataset	Total features	Significant features	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Specificity (%)	AUC-ROC (%)
CICMaldroid2017	84	15	90.38	90.84	91.14	90.99	91.23	90.88
CICAndMal2020	141	15	96.37	95.98	96.06	96.01	96.96	96.87
HelDroid	35	15	94.93	94.52	94.63	94.58	95.04	95.43
DroidRan	22	22	97.72	97.53	99.19	98.35	98.64	98.97

DroidRan achieves best performance on all three metrics with a success rate of 97.72%, an F1-score of 98.35%, and AUC-ROC of 98.97% while CICAndMal2020 needs 141 features to achieve 96.37%. In

DroidRan: hardware behavior dataset for multi-stage android ransomware detection (Kalaivani Selvaraj)

comparison to software telemetry, therefore, hardware telemetry provides a more compact but more discriminating representation.

The Friedman test ranks DroidRan first for both accuracy and F1-score, followed by CICAndMal2020, CICMaldroid2017, and HelDroid. HelDroid ranks lowest, consistent with its smaller feature set and its reliance on static software indicators, whereas DroidRan measures behavioral effects that ransomware cannot readily suppress. Because the Friedman test rejects the null hypothesis, a Wilcoxon signed-rank post-hoc test with Bonferroni correction was applied pairwise against DroidRan; Table 8 reports the result.

Table 8. Wilcoxon signed-rank post-hoc test for cross-dataset comparison

Comparison	Friedman rank (accuracy)	Post-hoc statistic (accuracy)	Significant	Friedman rank (F1)	Post-hoc statistic (F1)	Significant
CICMaldroid2017 versus DroidRan	3.1053	1.706	No	3.0526	1.6533	No
CICAndMal2020 versus DroidRan	2.1316	0.7323	No	2.1316	0.7323	No
HelDroid versus DroidRan	4.8421	3.4428	No	4.8421	3.4428	No
DroidRan (reference)	1.5	—	—	1.5	—	—

3.3. Comparative evaluation of detection models

The proposed framework was compared against DBN-GRU [20], iCNN-LSTM+ [21], MHA-LSTM+EGTO [22], and SA-CGRU [23], all trained and evaluated on DroidRan under the same 10-fold stratified cross-validation protocol. Table 9 reports detection performance together with the Friedman ranks.

The proposed framework leads on every metric, reaching 99.372% accuracy, 99.012% precision, 98.995% recall, and 98.996% F1-score, with the lowest FPR (0.251%) and FNR (1.005%). Among the baselines, SA-CGRU is strongest, followed by iCNN-LSTM+, MHA-LSTM+EGTO, and DBN-GRU. The advantage follows from the framework capturing both short-term and long-term temporal dependence in the telemetry sequence, which is what separates stage-specific behavioral patterns from one another.

Table 9. Comparative evaluation of detection models on DroidRan with Friedman ranks

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	FPR (%)	FNR (%)	AUC	Rank (acc.)	Rank (F1)
DBN-GRU [20]	95.862	95.784	94.983	95.107	1.255	5.017	0.964	3.12	3.9
iCNN-LSTM+ [21]	97.117	97.022	96.989	96.985	0.753	3.011	0.984	2.3	3.12
MHA-LSTM+EGTO [22]	96.489	96.049	95.985	95.991	1.004	4.015	0.978	2.7	3.15
SA-CGRU [23]	97.744	98.015	97.992	97.988	0.502	2.008	0.99	2	2.5
GRU-LSTM (proposed)	99.372	99.012	98.995	98.996	0.251	1.005	0.995	1.5	1.72

The Friedman test returns $p=0.0025$ for both accuracy and F1-score, well below $\alpha=0.05$, and so rejects the hypothesis that the five models perform equally. Repeating the test across all seven metrics confirms significant differences throughout, with $p=0.002505$ for accuracy, 0.003606 for precision, 0.002812 for recall, 0.002527 for F1-score, 0.002618 for FPR and FNR, and 3.95×10^{-5} for AUC, the last of these the strongest.

Following rejection of the Friedman null hypothesis, the Wilcoxon signed-rank post-hoc test was applied pairwise against the proposed model with a Bonferroni correction over $m=4$ comparisons, giving a corrected threshold of $\alpha/m=0.0125$. The corrected results are reported in Table 10 and show that the proposed framework performs substantially better than all the four baselines. The largest signed-rank difference is against iCNN-LSTM+ [21] and the smallest is against the MHA-LSTM+EGTO [22].

Table 10. Bonferroni-corrected Wilcoxon signed-rank post-hoc test against the proposed model

Comparison	p-value (acc.)	Corrected p (acc.)	Signed rank (acc.)	Significant	p-value (F1)	Corrected p (F1)	Signed rank (F1)	Significant
DBN-GRU versus proposed	0.00246	0.02254	27.5	Yes	0.00231	0.02269	25	Yes
iCNN-LSTM+ versus proposed	0.00437	0.04361	44.3	Yes	0.00437	0.04363	46	Yes
MHA-LSTM+EGTO versus proposed	0.00278	0.02222	22.5	Yes	0.00232	0.02268	22.5	Yes
SA-CGRU versus proposed	0.00189	0.03311	35.5	Yes	0.00142	0.03357	34	Yes

3.4. Per-family performance and error analysis

To avoid masking unequal behavior, Table 11 presents accuracy and F1-score measured for each family and the major source of residual errors. The performance levels reported are for each family, and the range of 97.6% to 99.6% does not represent any selection of favourable cases. The easiest ones are families with a sharp signature with a high amplitude, like LockerGoga or FileCoder.C is designed to perform continuous encryption, thereby raising the processor temperature and battery voltage, while DoubleLocker fuses encryption with a lock screen featuring graphics and provides a PIN change capability, thus creating a multi-resource pattern that is highly separable. The most difficult is that of Jisut, which only has a light memory footprint and shares in common with benign background traffic. Koler is difficult to detect, especially when accessing the website initially, because the browser-based UI and SMS traffic appears as normal browsing, and SpyNote generates long-running, benign-looking memory leaks. Most residual errors are adjacent-stage confusions, typically PE against CC during short encryption bursts, rather than benign-against-ransomware errors, which is consistent with the low aggregate FNR and FPR in Table 9.

Table 11. Per-family detection performance and dominant error source for the proposed model

Family	Accuracy (%)	F1-score (%)	Primary source of residual error
CovidLock	99.6	99.5	Very low; strong processor and UI-loop signature
CryCryptor	99.1	99	Occasional PE/CC confusion during short AES bursts
DoubleLocker	99.5	99.4	Very low; distinct GPU activity with PIN change
FakeApp Locker	98.2	98	Overlay UI gives a RAM-only signal with weak CC cues
FileCoder.C	99.3	99.2	Low; clear voltage and processor rise on full-disk encryption
Jisut	97.6	97.3	Hardest; low overall footprint resembles benign applications
Koler	97.9	97.7	Web UI and SMS traffic overlap benign browsing at IA
LockerGoga	99.5	99.4	Very low; sustained high processor temperature
RafelRAT	98.8	98.6	C2 exfiltration overlaps network-heavy benign applications
SimpLocker	99.2	99.1	Low; SD card I/O latency is discriminative
SLocker	99	98.9	Low; graphics and animation pressure is distinctive
SpyNote	98.7	98.5	Background memory leaks overlap long-running benign applications

3.5. Discussion, deployment considerations, and limitations

Three findings follow from the above analysis. First, every hardware feature in DroidRan carries discriminative information, which supports hardware telemetry as a behavioral profiling substrate rather than a supplementary signal. Second, DroidRan ranks first among the four datasets on both accuracy and F1-score while using the fewest features, which supports its use as a benchmark particularly for ransomware that software-level analysis would miss. Third, the proposed framework leads on all seven metrics with statistically significant margins over four recent baselines.

These gains carry explicit costs. The four-branch parallel design enables stage-specific non-linear feature learning, but it raises parameter count and memory footprint relative to a single-branch model, trading computation for detection granularity. Retaining all 22 features on DroidRan, rather than reducing dimensionality as on the external datasets, trades slightly higher input dimensionality for stronger discrimination, and remains far more compact than the 141 features of CICAndMal2020. In deployment, the framework is intended as the detection engine of an on-device or companion Android security application: all 22 features are exposed through standard non-rooted diagnostic interfaces at one-second resolution, so a lightweight background service can stream the same telemetry from which the dataset was built. Because detection is stage-aware, such an application can escalate gradually, raising an alert and quarantine prompt at IA or PE rather than reacting only after Im. On resource-constrained handsets the four-branch model can run in a companion service while the device streams telemetry, or a distilled single-branch variant can run locally at reduced granularity.

Several limitations qualify for these results. The dynamic threat intelligence module relies on pre-defined profiles for 12 families and must be extended as new families appear, and the 50% sequence-similarity criterion, adequate here, may require adjustment on corpora with greater inter-family similarity. DroidRan was collected on a single Android device carrying one unpatched vulnerability, so hardware diversity is limited and absolute telemetry magnitudes are device-specific; cross-device and cross-version validation is required before device-agnostic thresholds can be claimed. All telemetry was captured on physical hardware rather than an emulator, which improves fidelity, but the daytime collection window and background application profile form are part of the measurement context and may shift the benign baseline under other usage patterns. Finally, although hardware telemetry is harder to obfuscate than software indicators, an adversary aware of the detector could pace or throttle encryption to flatten the signature, and robustness against such evasion has not yet been quantified.

4. CONCLUSION

This paper addressed two gaps in Android ransomware research: the absence of a hardware telemetry benchmark labelled by lifecycle stage, and the absence of a detector that identifies the active attack stage and attributes the family from hardware behavior. DroidRan answers the first. It records 22 hardware features across six resource categories from an Android device with the unpatched CVE-2024-36971 vulnerability while 12 ransomware families execute through watering-hole, cross-site scripting, and spear-phishing squatting vectors, and it maps the resulting variation to the four MITRE ATT&CK mobile stages. The Friedman test confirms that all 22 features separate benign from ransomware behavior, with processor clock speed, available storage, and GPU frequency the most discriminative. Cross-dataset benchmarking places DroidRan first among four corpora on both accuracy and F1-score while using the fewest features. The proposed GRU-LSTM framework answers the second. Four parallel GRU branches perform stage-specific analysis that preserves within-state fluctuation otherwise lost to threshold-only selection; a cluster-based graph Bayesian network with correlation attention identifies the predominant feature per cluster and thereby the active stage; and a dual-branch module pairs a pre-trained LSTM producing the feature sequence order with a dynamic threat intelligence store that attributes the family by sequence matching. Against four recent baselines on DroidRan, the framework attains the highest accuracy, precision, recall, F1-score, and AUC and the lowest FPR and FNR, with differences significant under a Friedman test and a Bonferroni-corrected Wilcoxon post-hoc test. Three directions follow from the limitations. Cross-device validation should extend collection to multiple chipsets and operating system versions and learn device-normalized relative features, so that thresholds transfer across heterogeneous handsets. Integration work should package the framework as a lightweight background telemetry service with a distilled on-device model, and benchmark its processor, memory, battery, and latency overhead on commodity phones. Adversarial robustness should be evaluated and hardened against evasion strategies that pace malicious activity to flatten hardware signatures, using adversarial training and rate-independent temporal features. Together with periodic expansion of the threat intelligence profiles, these directions move DroidRan from a benchmark toward a deployable, evasion-resilient Android ransomware defense.

ACKNOWLEDGMENTS

The authors sincerely thank Universiti Sains Malaysia for their continuous support throughout this study.

FUNDING INFORMATION

No financial support was received for this study.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Kalaivani Selvaraj	✓	✓	✓		✓	✓		✓	✓		✓			
Manmeet Mahinderjit Singh	✓	✓		✓	✓		✓			✓		✓	✓	

C : C onceptualization	I : I nterpretation	Vi : V isualization
M : M ethodology	R : R esources	Su : S upervision
So : S oftware	D : D ata Curation	P : P roject administration
Va : V alidation	O : Writing - O riginal Draft	Fu : F unding acquisition
Fo : F ormal analysis	E : Writing - Review & E diting	

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The dataset that supports the findings of this study are openly available in Mendeley Data at <http://doi.org/10.17632/xwnm862mbp.1>. The repository contains the raw and preprocessed hardware-telemetry samples, the stage-wise MITRE ATT&CK-labelled organization described in section 2, and the feature documentation. The experiments reported here can be reproduced using the configuration and hyperparameters given in subsection 2.6. Additional families will be released as new versions of the same repository record.

REFERENCES

- [1] V. P. Patil, "A Comprehensive Review of Cybersecurity Threats and Mitigation Strategies," *Int. J. Sci. Res. Eng. Manag.*, vol. 10, no. 01, pp. 1–9, Jan. 2026, doi: 10.55041/IJSREM55849.
- [2] U. Urooj, B. A. S. Al-rimy, A. Zainal, F. A. Ghaleb, and M. A. Rassam, "Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions," *Appl. Sci.*, vol. 12, no. 1, pp. 1–45, Dec. 2021, doi: 10.3390/app12010172.
- [3] A. Vehabovic, N. Ghani, E. Bou-Harb, J. Crichigno, and A. Yayimli, "Ransomware Detection and Classification Strategies," in *2022 IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom)*, IEEE, Jun. 2022, pp. 316–324, doi: 10.1109/BlackSeaCom54372.2022.9858296.
- [4] A. A. Ahmed, A. Shaahid, F. Alnasser, S. Alfaddagh, S. Binagag, and D. Alqahtani, "Android Ransomware Detection Using Supervised Machine Learning Techniques Based on Traffic Analysis," *Sensors*, vol. 24, no. 1, pp. 1–21, Dec. 2023, doi: 10.3390/s24010189.
- [5] L. Albshaier, S. Almarri, and M. M. H. Rahman, "Earlier Decision on Detection of Ransomware Identification: A Comprehensive Systematic Literature Review," *Information*, vol. 15, no. 8, pp. 1–48, Aug. 2024, doi: 10.3390/info15080484.
- [6] M. Ibrahim, B. Issa, and M. B. Jasser, "A Method for Automatic Android Malware Detection Based on Static Analysis and Deep Learning," *IEEE Access*, vol. 10, pp. 117334–117352, 2022, doi: 10.1109/ACCESS.2022.3219047.
- [7] S. Gulmez, A. G. Kakisim, and I. Sogukpinar, "XRan: Explainable deep learning-based ransomware detection using dynamic analysis," *Comput. Secur.*, vol. 139, p. 103703, Apr. 2024, doi: 10.1016/j.cose.2024.103703.
- [8] I. U. Haq, T. A. Khan, and A. Akhunzada, "A Dynamic Robust DL-Based Model for Android Malware Detection," *IEEE Access*, vol. 9, pp. 74510–74521, 2021, doi: 10.1109/ACCESS.2021.3079370.
- [9] G. Ciaramella, G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Explainable Ransomware Detection with Deep Learning Techniques," *J. Comput. Virol. Hacking Tech.*, vol. 20, no. 2, pp. 317–330, Sep. 2023, doi: 10.1007/s11416-023-00501-1.
- [10] G. Kirubavathi and W. R. Anne, "Behavioral based detection of android ransomware using machine learning techniques," *Int. J. Syst. Assur. Eng. Manag.*, vol. 15, no. 9, pp. 4404–4425, Sep. 2024, doi: 10.1007/s13198-024-02439-z.
- [11] M. Sibtain, M. Hussain, Q. Riaz, S. Qadir, N. Riaz, and K.-H. Jung, "Lightweight and Robust Android Ransomware Detection Using Behavioral Analysis and Feature Reduction," *Comput. Mater. Contin.*, vol. 84, no. 3, pp. 5177–5199, 2025, doi: 10.32604/cmc.2025.066198.
- [12] K. Selvaraj, M. M. Singh, and Z. F. Zaaba, "HARVEST: Hardware-based ransomware detection and stage tracking for AIoMT devices," *J. Netw. Comput. Appl.*, vol. 250, p. 104458, Jun. 2026, doi: 10.1016/j.jnca.2026.104458.
- [13] M. A. Hossain, T. Hasan, F. Ahmed, S. H. Cheragee, M. H. Kanchan, and M. A. Haque, "Towards superior android ransomware detection: An ensemble machine learning perspective," *Cyber Secur. Appl.*, vol. 3, pp. 1–12, Dec. 2025, doi: 10.1016/j.csa.2024.100076.
- [14] N. Singh and S. Tripathy, "It's too late if exfiltrate: Early stage Android ransomware detection," *Comput. Secur.*, vol. 141, p. 103819, Jun. 2024, doi: 10.1016/j.cose.2024.103819.
- [15] I. Almomani *et al.*, "Android Ransomware Detection Based on a Hybrid Evolutionary Approach in the Context of Highly Imbalanced Data," *IEEE Access*, vol. 9, pp. 57674–57691, 2021, doi: 10.1109/ACCESS.2021.3071450.
- [16] A. Mahindru and A. L. Sangal, "MLDroid—framework for Android malware detection using machine learning techniques," *Neural Comput. Appl.*, vol. 33, no. 10, pp. 5183–5240, May 2021, doi: 10.1007/s00521-020-05309-4.
- [17] H. H. R. Manzil and S. M. Naik, "Android ransomware detection using a novel hamming distance based feature selection," *J. Comput. Virol. Hacking Tech.*, vol. 20, no. 1, pp. 71–93, Aug. 2023, doi: 10.1007/s11416-023-00495-w.
- [18] J. Hwang, J. Kim, S. Lee, and K. Kim, "Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques," *Wirel. Pers. Commun.*, vol. 112, no. 4, pp. 2597–2609, Jun. 2020, doi: 10.1007/s11277-020-07166-9.
- [19] H. Kauser.Sk and M. Anu.V, "Hybrid deep learning model for accurate and efficient android malware detection using DBN-GRU," *PLoS One*, vol. 20, no. 5, pp. 1–30, May 2025, doi: 10.1371/journal.pone.0310230.
- [20] J. Isphahany, M. R. Islam, M. A. Khan, and M. Z. Islam, "iCNN-LSTM+: A Batch-Based Incremental Ransomware Detection System Using Sysmon," *IEEE Access*, vol. 13, pp. 87978–87998, 2025, doi: 10.1109/ACCESS.2025.3569635.
- [21] S. A. Alzakari *et al.*, "An intelligent ransomware based cyberthreat detection model using multi head attention-based recurrent neural networks with optimization algorithm in IoT environment," *Sci. Rep.*, vol. 15, no. 1, pp. 1–21, Mar. 2025, doi: 10.1038/s41598-025-92711-4.
- [22] Y. Li and Y. Su, "A Network Traffic Anomaly Classification Model Based on Self-Attention Mechanism and Convolutional Gated Recurrent Unit," *IEEE Access*, vol. 13, pp. 134804–134821, 2025, doi: 10.1109/ACCESS.2025.3593613.
- [23] G. M. and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," *Comput. Sci. Rev.*, vol. 47, p. 100529, Feb. 2023, doi: 10.1016/j.cosrev.2022.100529.
- [24] L. da Costa and V. Moia, "A Lightweight and Multi-Stage Approach for Android Malware Detection Using Non-Invasive Machine Learning Techniques," *IEEE Access*, vol. 11, pp. 73127–73144, 2023, doi: 10.1109/ACCESS.2023.3296606.
- [25] A. Al Sharah, Y. A. Alrub, H. A. Owida, E. A. Elsoud, N. Alshdaifat, and H. Khatnaha, "An Adaptive Framework for Classification and Detection of Android Malware," *Int. J. Interact. Mob. Technol. (IJIM)*, vol. 18, no. 21, pp. 59–73, 2024, doi: 10.3991/ijim.v18i21.49669.

BIOGRAPHIES OF AUTHORS

Kalaivani Selvaraj    received her Bachelor of Science (First Class Hons.) degree in Computer Science from Madurai Kamaraj University, India; her Master of Science (First Class Hons.) degree in Statistics with Computer Applications from Annamalai University, India; and her Master of Philosophy (First Class Hons.) degree in Computer Science from Mother Theresa Women's University, India. She is pursuing her Ph.D. in Computer Science under Universiti Sains Malaysia (USM), Malaysia. She was a lecturer in School of Information Security, University of Technology and Applied Sciences-Nizwa, Oman. Her research interests include data security and privacy, cyber security, and artificial intelligence. She can be contacted at email: kalaivani.selvaraj@student.usm.my.



Manmeet Mahinderjit Singh    is an associate professor born in Perak, Malaysia, in 1978. She received her B.Sc. and M.Sc. degrees in Computer Sciences from Universiti Sains Malaysia and her Ph.D. from The University of Queensland, Australia. Her research interests focus on computer and data security, trusted models and systems, and cyber-crime. She is also actively involved in Science, Technology, Engineering and Mathematics (STEM) education and has been appointed as a Master Trainer in computational thinking. She can be contacted at email: manmeet@usm.my.