Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.083615

ARTICLE

McIFAR: A Multi-Contextual Interaction-Based Framework for Detecting Context-Triggered Android Ransomware

Sonam Jain[1], Tanya Gera[2,*], Rupali Gill[1], Afnan Almegren[3], Ateeq Ur Rehman[4,*] and Salil Bharany[1]

[1]Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India
[2]School of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab, India
[3]Department of Applied Linguistics, College of Languages, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia
[4]School of Computing, Gachon University, Seongnam-si, Republic of Korea

*Corresponding Authors: Tanya Gera. Email: tanya.gera.390@gmail.com; Ateeq Ur Rehman. Email: 2024111144@gachon.ac.kr

Received: 07 April 2026; Accepted: 09 June 2026; Published: 13 August 2026

ABSTRACT: Android ransomware has emerged as a major threat to mobile ecosystems. Modern Android ransomware has evolved beyond the reach of traditional signature-based detection, often lying dormant until specific strategic triggers activate its malicious payload. These strategic ransomware variants activate payloads only under specific device states, events, and conditions that are absent in a sandbox testing environment. To address these sophisticated evasion tactics, this article introduces a novel framework, McIFAR (Multi-contextual Interaction-based Detection Framework for Android Ransomware), that leverages in-context emulation within malware sandboxing to elicit dormant behaviours that are missed by conventional testing, thereby transcending the limitations of isolated static or dynamic analysis. A robust two-stage methodology is presented. In the first stage, the Cross-Validation Feature Selection Ensemble (CVFSE) identifies dominant indicators. This is followed by the Contextual Interaction Feature Orchestrator (CIFO), processing dominant features to encode complex behavioral interactions between features and context in the second stage. Unlike existing studies that rely solely on static and dynamic data, this approach prioritizes contextual interaction, thereby significantly enhancing detection accuracy. The experimental results on the KronoDroid dataset demonstrate that McIFAR achieves a 99.48% detection accuracy, outperforming traditional baselines. The statistical analysis using the Friedman and Nemenyi post-hoc tests confirms that the results are both significant and consistent. The future work includes enhancing the framework by incorporating richer contextual scenarios in in-context emulation, along with federated learning and real-time lightweight deployment.

KEYWORDS: Android ransomware; context; detection; feature; interaction; ransomware; machine learning

1 Introduction

In recent years, Android has become the dominant mobile operating system, holding nearly 69% of the global market share [1]. Its open ecosystem, which allows users to install applications from Google Play, third-party stores, and APK websites, has significantly contributed to its popularity. However, this flexibility, combined with the increasing storage of sensitive personal data on smartphones, has also made Android devices a major target for ransomware attacks [2]. According to Kaspersky's Q3 2025 Android threat report, 197,738 malicious samples and approximately 3.47 million mobile attack incidents were detected, including 1564 newly identified ransomware installation packages [3].

Modern Android ransomware has evolved beyond basic locking techniques by employing advanced evasion mechanisms such as code obfuscation, anti-analysis techniques, and conditional execution [4].

Attackers also exploit trusted platforms like Google Play, where users often trust applications despite suspicious permission requests [5,6]. To counter these threats, researchers have proposed static, dynamic, and hybrid analysis methods [7–9]. Although these approaches improve detection performance, they remain ineffective against context-dependent ransomware behavior, in which malicious actions are triggered only under specific conditions, such as user inactivity or device state changes. Consequently, existing frameworks that rely on generic execution environments often fail to capture hidden malicious behavior and realistic ransomware execution patterns. The wide range of contextual triggers mentioned in Table 1 highlights the inherent diversity and context-sensitive activation behaviour of Android ransomware. These observations indicate a critical limitation in current detection frameworks, which predominantly rely on generic execution environments that inadequately capture real-world usage scenarios.

Table 1: Contextual triggers for ransomware [10,11].

Context	Ransomware Feasibility	Reason
Charging (Plugged In)	Likely	Encrypts files while the user is idle.
During Call	Likely	The user is busy on the call and distracted.
After Reboot/Boot Completed	Very Common	Classic ransomware move.
Connected to Wi-Fi	Highly Likely	Network-dependent command-and-control initiation.
Bluetooth Connection	Unlikely	Uncommon activation mechanism for ransomware.
Certain Apps Opened	Likely	Sometimes associated with ransomware.
Headphones Plugged	Rare	This behaviour is not feasible for ransomware operations
Specific Geolocation	Possible	Geo-fenced ransomware behaviour.
Device Unlock	Likely	Execute the attack after the user returns.
Device Idle/Not Moving	Likely	This is a safe time for encryption.
Aeroplane Mode On	Possible	Encryption can continue offline.
Nighttime-Based	Very Likely	Encrypt at night to delay detection.
SIM Card Changed	Rare	Primarily associated with SIM hijacking.
Screen Off/Locked	Likely	Silent encryption during user inactivity.

To address these challenges, there is a growing need for advanced analysis methods capable of reproducing realistic execution environments. In-context emulation has emerged as an effective approach to simulating user interactions, device states, and system conditions that trigger hidden ransomware behaviour, enabling more accurate behavioural analysis and detection. To overcome limitations in existing studies, this work proposes McIFAR, an Android ransomware detection framework based on in-context emulation, hybrid feature selection, and interaction-based feature engineering. The framework performs trigger-dependent contextual behavioural analysis to expose concealed ransomware activities under realistic conditions. In addition, the proposed CVFSE algorithm selects dominant features using a hybrid feature selection strategy. In contrast, interaction-based feature engineering captures complex ransomware behaviours, such as encrypting files only when the device is idle and charging or initiating command-and-control communication during system boot. It highlights the shortcomings of traditional detection approaches, which often overlook delayed, conditional, or covert execution patterns, and presents a simulation-based pipeline to extract and analyse these contextual indicators. This study contributes: (a) To propose a hybrid threat detection framework for Android ransomware, McIFAR, based on intensive trigger-dependent analysis that incorporates in-context emulation within malware sandboxing, thereby improving the quality of behavioral feature extraction and strengthening detection performance. (b) To identify the final feature set, a robust two-stage methodology is proposed that devises a CVFSE algorithm to determine dominant features, followed by a CIFO algorithm to encode complex behavioural interactions between features and context. (c) To ensure a thorough evaluation of the proposed framework, apply statistical

significance tests to validate performance gains and conduct an ablation study to quantify the contribution of individual components, thereby enhancing the robustness and interpretability of the findings.

The remaining manuscript is organised into sections, with Section 2 providing an analysis of related work. Furthermore, the detailed description of the proposed methodology is provided in Section 3. Afterwards, Section 4 presents the results and discussion. Finally, the work is concluded in Section 5.

2 Related Work

The research landscape of Android malware detection includes static, dynamic, and hybrid analysis techniques designed to address increasingly sophisticated ransomware attacks. Existing studies have introduced several analysis tools, such as Droidbox, CopperDroid, MobSF, and CuckooDroid, for comprehensive Android application analysis. However, these tools often require substantial computational resources and remain vulnerable to anti-emulation techniques. To provide a structured overview, prior studies are categorised based on the analysis approach employed. Table 2 summarizes leading feature selection techniques utilized for Android malware detection and classification across static, dynamic, and hybrid analysis methods.

Table 2: Summary of feature selection techniques for Android malware detection and classification.

Reference	Year	S	D	H	Det	Class	Feature Selection Method		
							Filter	Wrapper	Embedded
[12]	2025	✓	×	×	✓	✓	✓	×	✓
[13]	2025	×	×	✓	✓	✓	×	✓	×
[14]	2025	×	✓	×	✓	✓	✓	×	✓
[15]	2024	×	✓	×	✓	×	×	×	×
[16]	2024	✓	×	×	✓	✓	×	×	×
[17]	2024	×	✓	×	✓	✓	×	✓	×
[18]	2024	✓	×	×	✓	×	✓	×	×
[19]	2023	✓	×	×	✓	✓	×	×	×
[20]	2023	×	×	✓	✓	✓	✓	×	×
[21]	2023	×	✓	×	✓	✓	✓	×	×
[22]	2023	×	×	✓	✓	✓	✓	×	×
[23]	2022	×	×	✓	✓	✓	×	✓	×
[24]	2022	×	✓	×	✓	✓	✓	✓	×
[25]	2022	×	×	✓	✓	✓	×	×	×
[26]	2021	×	×	✓	✓	✓	×	✓	×
[27]	2021	×	×	✓	✓	✓	✓	×	×
[28]	2020	×	✓	×	✓	✓	✓	×	✓
[29]	2020	✓	×	×	×	✓	✓	×	×
[30]	2019	✓	×	×	✓	✓	✓	✓	×
[31]	2019	×	✓	×	✓	✓	✓	×	×
[32]	2018	×	✓	×	✓	✓	✓	✓	×
[33]	2018	✓	×	×	✓	✓	×	×	✓

Note: Abbrev: Det-Detection, D-Dynamic, C-Class, H-Hybrid, S-Static.

The literature review indicates that several studies have incorporated feature selection techniques in Android malware detection, primarily combining filter-based and wrapper-based techniques or by employing either approach individually [13,29–33]. Although prior studies integrate filter-based and wrapper-based techniques, to the best of our knowledge, no existing approaches systematically train and evaluate multiple filter-wrapper combinations to determine the optimal discriminative feature subset for Android ransomware detection. A significant number of existing approaches focus on static features [12,15,18] or assume feature independence, thereby failing to capture contextual dependencies and feature interactions. Furthermore, wrapper-based techniques are effective but computationally intensive, whereas filter-based approaches often fail to capture complex feature dependencies. Furthermore, there has been inadequate consideration of diverse execution contexts and the dynamic evolution of ransomware behaviour. Recent deep learning methods [15,16] have demonstrated strong ransomware detection capabilities. However, their decision-making processes are often less transparent. To address these limitations, this work proposes a two-stage approach that integrates dominant feature selection with context-critical feature engineering using in-context emulation within malware sandboxing. Specifically, top feature interactions between contextual and dominant features are captured by incorporating SHAP, which facilitates feature-level interpretation and provides a balance between detection effectiveness, interpretability, and computational efficiency.

3 Methodology

The literature reviewed in Section 2 highlighted the importance of effective feature selection and contextual analysis in improving ransomware detection performance. To address these aspects, the proposed methodology for the McIFAR, a multi-contextual interaction-based detection framework, is shown in Fig. 1.

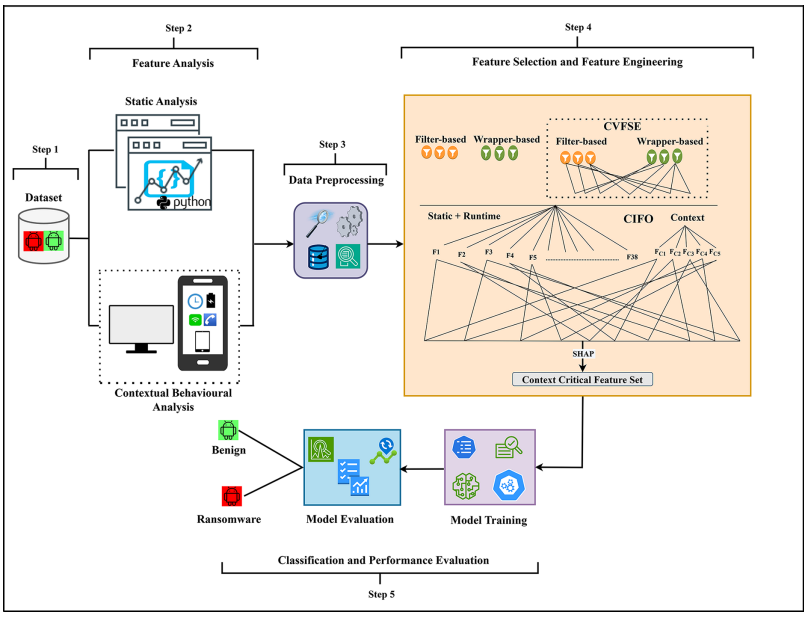

Figure 1: Proposed methodology.

The method comprises five steps. It includes dataset, feature analysis, data preprocessing, feature selection, feature engineering, classification and performance evaluation. The further subsections provide details of each step.

3.1 Dataset

The dataset used in this study is accessed from KronoDroid [34]. This is the biggest dataset, having 209 Android malware families. This is an open-source real-world dataset and has been used in recent ransomware detection research [19,35]. The dataset details are shown in Table 3.

Table 3: Dataset collection.

KronoDroid	
Malware	72,001
Benign	70,127
Ransomware	3668

Initially, 3668 ransomware samples and 3668 benign samples were analysed to consider the class distribution during model training.

3.2 Feature Analysis

Feature analysis is a crucial step in Android ransomware detection, enabling the extraction of meaningful indicators of malicious behaviour from raw application data. This is to analyse both inherent code properties using static analysis and runtime behaviours with contextual triggers called contextual behavioural analysis [36].

3.2.1 Static Analysis

The static analysis is to extract features of the APK from its code without running on an emulator. It is performed using the Androguard tool, which utilizes Python scripts. The tool extracts basic APK information, such as package name, permissions, intents, services, and API calls, from the APK's AndroidManifest.xml file.

3.2.2 Contextual Behavioural Analysis

The behaviour of an APK is analysed after running it in an Android emulator, using the Monkey tool to simulate user interactions and capture strace and logcat logs. A behavioural study of APKs is conducted by simulating different contexts, and their context-critical triggers are illustrated in Fig. 2a–e.

The contexts considered for this study are in the range of C1–C5, as explained in the subfigures.

C1: On-phone-charge attack

The charging state is shown in Fig. 2a and toggled while running the APK on the emulator using the adb commands in the script. adb shell dumpsys battery set ac 1 for charging and adb shell dumpsys battery set ac 0 for discharging. Ransomware is triggered when the device is connected to power, under the assumption that the user is not using the device and that the battery has sufficient power to perform resource-intensive activities such as file encryption. Ransomware, in some cases, waits until this event to minimize user suspicion and prevent battery drain during encryption.

C2: On-phone-call

The state was simulated using the command telnet localhost 5554 → gsm call. Certain ransomware imposes a wait period before a phone call can be initiated or terminated, believing the victim is distracted or compromised. This state is shown in Fig. 2b.

Figure 2: Context-critical triggers.

C3: Wi-Fi connectivity

To simulate this context state, the adb command adb shell svc wifi enable/disable is used. Ransomware waits for a Wi-Fi connection to download encryption keys from a remote command-and-control server and upload the stolen data. This context state is shown in Fig. 2c.

C4: Screen off context

The screen-off context is shown in Fig. 2d. This state is created with the adb shell input keyevent 223 command. Ransomware initiates encryption or payload transfer while the screen is off, considering the device is idle and left unattended. This can be traced in the logcat.txt and stace.txt log files, which are generated by contextual behavior analysis. This minimizes the chance that the user will recognize suspicious activity or a performance slowdown.

C5: Time-based/nighttime

The state is created using adb shell date-s "YYYYMMDD.HHMMSS" The behaviour of an app is studied in specific time windows. These windows are 6 a.m. to 12 p.m., 12 p.m. to 6 p.m., 6 p.m. to 12 a.m., and 12 a.m. to 6 a.m.. The window from 12 a.m. to 6 a.m. is considered an attack window or nighttime attacks when the user is sleeping and not using the phone, as shown in Fig. 2e. The features are extracted from APKs in two forms: static.json and contextualbehavioural.json files from log.txt and starce.txt logs using Python scripts. All the diverse features of each APK are stored as feature vectors with the APK package name

and other metadata. Following the feature analysis step, the derived features are further processed in a data preprocessing stage to enhance data quality and ensure compatibility with ML models.

3.3 Data Preprocessing

The two types of features, static and contextual behavioural, are extracted in a heterogeneous form. The following preprocessing steps are essential to obtain a dataset compatible with an ML classifier. Fig. 3 illustrates the steps used in the data preprocessing phase.

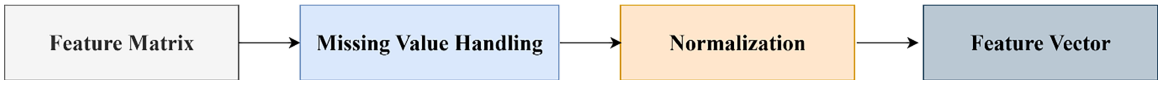

Figure 3: Data preprocessing process flow.

Feature matrix: Initially, from raw logs, both static and runtime features are stored in a feature matrix using one-hot encoding, in which categorical fields (e.g., API calls, System calls, Android permissions, intent filters) are one-hot encoded to produce binary indicator features.

Missing Value handling: There are some empty log files for a few APKs in the dataset. These APKs cannot be installed on the emulator due to a version mismatch. Thus, contextual behavioural analysis was not conducted for those applications. All extracted features are checked for missing or null values. Missing features (e.g., due to unsuccessful dynamic execution or missing logs) are assigned a value of 0 (for absence), and samples with excessive missing data are removed.

Normalisation: Features that are continuous or count-based (e.g., no. of file writes, log segments) are normalised using Min-Max scaling or a StandardScaler to rescale them to [0, 1] or to have a zero mean and unit variance.

Feature Vector Construction: All engineered and extracted features (static and dynamic) are combined into a single flattened feature vector per APK. The existence of a feature within the application is marked as 1, while 0 indicates that the feature is not present. Let's assume there are n features for each application. The feature set is F = F_1, F_2, ..., F_n. A feature vector for each application in the dataset is constructed using the formula in Eq. (1).

$$F_n = \begin{cases} 1 & \text{if present} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

3.4 Feature Selection and Feature Engineering

Feature selection and engineering techniques were applied to identify the most relevant attributes and enhance the dataset's representativeness. This approach ensures that only relevant features are retained while creating additional ones that capture meaningful relationships within the data. Additionally, engineered features improve computational efficiency and enhance model generalization.

3.4.1 Feature Selection

Feature selection was applied to reduce dimensionality and retain the most discriminative ransomware features. Since features selected by different methods may provide complementary information, the proposed CVFSE algorithm combines filter-based and wrapper-based feature selection techniques to perform a more comprehensive exploration of the feature space. Algorithm 1 takes ransomware feature sets from different

samples as input and outputs a dominant feature subset selected through a cross-validation feature selection ensemble approach.

Algorithm 1: Cross-validation feature selection ensemble (CVFSE)

Input: A dataset of distinct ransomware features without context extracted from multiple samples.
Output: Set of dominant features selected, using a cross-validation feature selection ensemble.
Symbols Used:
T_i: Feature selection technique
ST_j: Sub-technique
R_s: Ransomware samples.
Step 1: An assembled set of distinct ransomware features,
$F = \{F_1, F_2, ..., F_x\} \ \forall \ R_s$
Where *x is the number of features*
Step 2: **FOR** i = 1 to P DO
Apply $T_i \ \forall \ R_s$
FOR j = 1 TO Q DO
FC_{ij} = feature selection based on ST_j
END FOR
END FOR
Considering $FC_{1,j}$ = FT, $FS_{2,j}$ = WT
Step 3: Identify each possible integration of wrapper and filter techniques
Pairs = $\{(FT_n \cup WT_m) \mid 1 \leq n \leq y, 1 \leq m \leq z\}$
Total pairs = y × z
Where u indexes the filter methods and v indexes the wrapper methods.
Step 4: Model training with all possible pairs for dominant feature selection

Let F be the total features of all ransomware samples in the dataset. Here, x is 280, $F = \{F_1, F_2, ..., F_x\}$ in the collected dataset. T_i is the feature selection technique that is filter-based and wrapper-based. P is the number of feature selection techniques, and Q is the number of sub-techniques. Considering $FC_{1,j}$ as filter-based techniques and $FS_{2,j}$ as wrapper-based techniques. The next step is to determine the possible integration of each filter-based technique with wrapper-based techniques to collect the total number of combinations. The total number of pairs is the product of the number of filter techniques and the number of wrapper techniques. In the filter-based techniques, three sub-techniques are MI, chi-square, and ANOVA F-score; the later two are applied to MinMax-scaled inputs. The wrapper-based techniques have three sub-techniques: Recursive feature elimination (RFE), Recursive feature elimination with cross-validation (RFECV), and sequential forward selection (SFS). The filter-based and wrapper-based techniques incorporated in the proposed framework are described in detail below. In the proposed CVFSE, several pairs of feature selection sub-techniques, both filter-based and wrapper-based, are examined to select a dominant feature set. The total pairs are y × z, where y = z = 3, hypothesized to provide enough combinations for feature selection. This suggested approach used an incremental feature selection method. The reason for using u = v = 3 is to ensure enough combinations are visible for model training and to identify results based on all feasible pairs.

Filter-Based Techniques: Filter-based techniques provide a quick and efficient way to minimize dimensionality. However, they ignore feature interactions [37]. Three widely used filter-based techniques, which rank features individually in terms of statistical metrics, are as follows:

Mutual Information (**MI**): MI quantifies the strength of the relationship between a given feature f and the target class c. A lower MI value suggests that the two features are largely independent, whereas a higher MI value reflects greater shared information and a stronger association. The definition of MI is that it is zero only when two random variables are completely independent, and it increases significantly when the variables are more highly correlated. In the ransomware dataset, the features are $R_f = \{f_1, f_2, \ldots, f_n\}$, and the main goal is to identify an optimal subset of relevant features, R_F', such that $R_F' \subseteq R_f$ and, for R_F', a classifier attains the best possible classification accuracy. The focus is to select a subset of features that satisfies this for every possible pair (f_i, f_j) R_F'. The feature-feature MI is minimal, and the feature-class MI is maximum.

$MI(f, c)$ is the MI between feature f and target class c, as in Eq. (2).

$$MI(f, c) = \sum_{f \in F} \sum_{c \in C} p(f, c) \log(p(f, c)/p(f)\, p(c)) \tag{2}$$

$p(f, c)$ is the joint probability that feature f will assume with respect to target class c. $p(f)$ is the probability that feature f will assume a specific value, which is independent of c. $p(c)$ is the probability of the target class C, which is independent of f. The features are ranked in ascending order of $MI(f)$ values with respect to target C, and the top features that meet the selection condition are used in the subsequent process. Usually, the stronger the association between c and f, the greater the value of $MI(f)$. Nevertheless, c and f are considered independent (i.e., there is no link) if $MI(f) = 0$ [38].

Chi-Square: This test quantifies the correlation between a given feature and the output class. The proposed approach uses the chi-square test to identify the features that exhibit the highest association with the class. It relies on comparing the expected and observed distributions; a higher chi-square statistic indicates a feature with greater relevance. It determines the statistical relationship between a feature and the target class. It measures if a feature and the target class exhibit statistical dependency, serving as a direct indication of the feature's relevance. The mathematical representation of the chi-square test for each feature f is given in Eq. (3).

$$chi(f, c_i) = n \frac{(fre(f, c_i)\, f(\overline{f}, \overline{c}_i) - fre(\overline{f}, c_i) f(f, \overline{c}_i)^2}{(fre(f, c_i) + fre(f, \overline{c}_i))(fre(f, \overline{c}_i) + fre(\overline{f}, \overline{c}_i))(fre(f, c_i) + fre(\overline{f}, c_i))(fre(f, \overline{c}_i) + freq(\overline{f}, \overline{c_i})} \tag{3}$$

where n is the total number of applications and f is a feature, $fre(f, c_i)$ is the frequency number of applications of the class c_i that uses feature f, $fre(\overline{f}, \overline{c}_i)$ is the frequency number of the applications, not of the class c_i that does not use feature f, $fre(f, \overline{c}_i)$ is the frequency of the applications of the class c_i that does not use feature f, $fre(\overline{f}, c_i)$ is the frequency number of the applications, not of the class c_i that uses feature f [12].

ANOVA F-Score: The analysis of variance (F-value) method was employed on feature sets to identify the most significant features with the highest F-scores. This metric accesses the similarity among significant features and reduces the dimensional disparity between the malware and legitimate applications [39]. In the study by [40], ANOVA effectively mitigates the data imbalance, thereby enhancing the reliability and stability of the proposed approach. The mathematical expression of ANOVA is represented in Eq. (4).

$$\sum_{j=1}^{N} \frac{n_j(\overline{x}_j - \overline{x})^2}{N - 1} \tag{4}$$

where n_j is the number of observations in the j-th group, and N is the total number of groups. $\overline{x}_j$ is the sample mean in the jth group and $\overline{x}$ denotes the overall mean of the samples. The above methods capture diverse features: chi-square tests the association between categorical features, ANOVA identifies variance among

class distributions, and MI evaluates both linear and nonlinear relationships. These 3 filter-based techniques ensure that robust, diverse, and highly relevant features are selected. The number of retained features (k = 50) was determined empirically to balance feature significance and dimensional reduction.

Wrapper-based methods

These are advanced ML-based feature selection methods. In contrast to filter-based techniques, which select features based on statistical measures, wrapper techniques involve the model directly in the feature selection process, repeatedly training and testing it on different combinations of feature sets to select the optimal subset. The three wrapper-based methods used in the proposed approach are as follows:

RFE: RFE evaluates model accuracy as its criterion for selecting features that demonstrate higher predictive capability. The approach involves repeatedly dropping less important features while continuously rebuilding the model with the remaining subset [28]. Let F be the feature set $F = \{f_1 + f_2 + f_3 + \ldots f_n\}$. The RFE procedure involves training a tree-based model M and evaluating the feature contribution value FC_j using the Gini importance score as shown in Eq. (5).

$$Gini = 1 - \sum_{j=1}^{n} P_j^2 \tag{5}$$

where P is the probability of the selected sample belonging to class j in that node. Gini importance serves as a clear model-driven approach for tree-based algorithms to assign W_j during RFE.

1. First, train model M on the current feature subset.
2. The second step is to rank the features according to their absolute weight or importance score W_j, which is calculated in Eq. (6).

$$W_j = \sum_{nodeswhereFC_jused} \Delta Ginni \tag{6}$$

where W_j is the importance score that indicates how influential the feature is FC_j.

3. The third step is to eliminate the features with the lowest importance score W_j.
4. The last step is to repeat until the desired or most influential features remain.

RFE feature selection is widely used in malware detection settings because it is tree-based, naturally handling complex interactions and nonlinear behavior, and provides meaningful importance scores for iterative pruning.

RFECV: It builds on RFE by embedding cross-validation into each elimination stage to identify the most suitable feature subset. The RFECV algorithm eliminates redundant features and those with a low impact on classification results [31]. The model's performance across multiple cross-validation folds automatically guides the selection of the ideal feature count, as in Eq. (7), whereas RFE relies on a predefined number of retained features.

$$Subsetsize\,(K) = argmax_K CV_{score(K)} \tag{7}$$

where the cross-validation score $CV_{score(K)}$ is identified by the ideal feature count depending on model accuracy.

SFS: SFS is a bottom-up feature selection method that takes a null set of features. Let f_s be a null feature set as shown in Eq. (8).

$$f_s = \varnothing \tag{8}$$

For each feature F, initially $F \notin f_s$. Then evaluate model performance, Adding F to f_s. In every step, add the features that give the best improvements in f_s, as in Eq. (9).

$$F^* = arg\, max_{F \notin f_s}(f_s \cup \{F\}) \tag{9}$$

The termination process occurs when the inclusion of a new feature fails to yield performance gains, thereby identifying the suitable feature subset.

3.4.2 Feature Engineering

To identify contextual information for the hybrid threat detection framework, a novel algorithm, CIFO, is used to generate interaction features via feature engineering. The contextual interaction feature engineering technique aims to enable the model to uncover complex behavioural relationships between application features and their runtime conditions, as shown in Fig. 4.

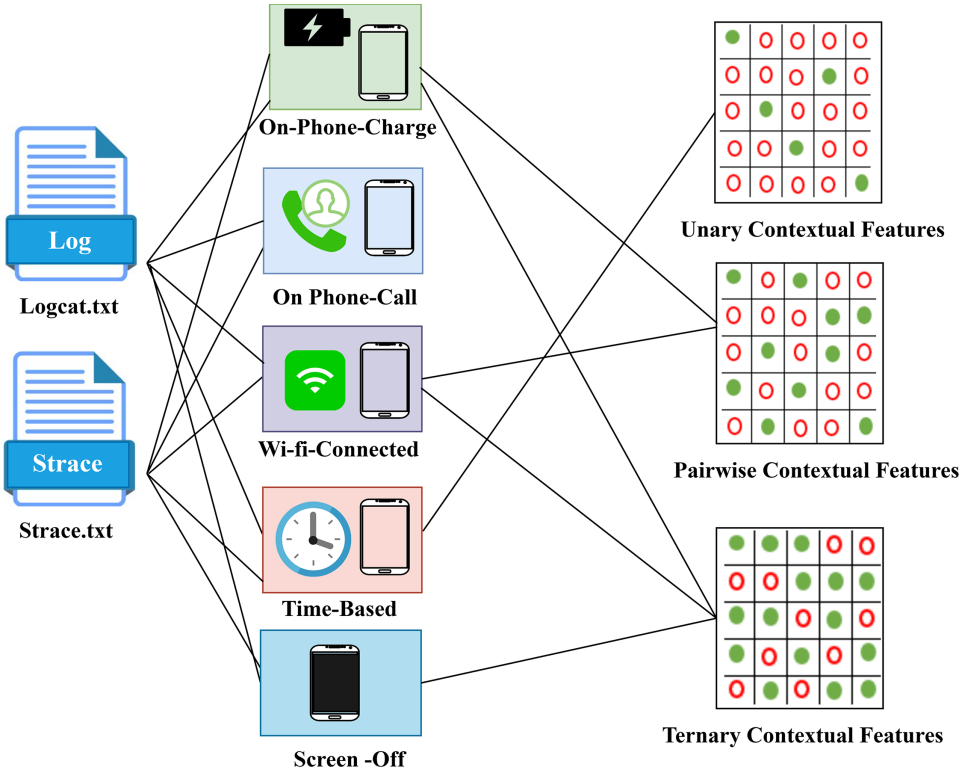

Figure 4: Interaction feature engineering.

In Algorithm 2, the output of Algorithm 1 is used as input, along with a dataset of samples across 5 contexts. Fig. 4 showcases the integration of features with the context in combination patterns such as (1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2), etc.

Algorithm 2: Contextual interaction feature orchestrator (CIFO)

Input: A set of dominant features selected = (output of Algorithm 1) with Contexts C = $\{C_1, C_2, \ldots, C_n\}$
Output: = List of final selected features.
Symbol used:
$X_{Interaction}$ is the total interaction features.
C_{count} is the total count of contexts
F_{count} is the total count of features.
Step 1: Initiate $X_{Interaction} = \varnothing$
Step 2: Compute the multiplicative interaction of every combination of (C, F)

$$I_{(C,F)} = \prod_{i=0}^{Ccount} C_i \times \prod_{j=1}^{Fcount} F_j$$

Step 3: Update $X_{Interaction}$

$$X_{Interaction} = X_{Interaction} \cup I_{(C,F)}$$

Step 4: Abs_shap ← mean (|shap_values|, axis = 0)
Quantifies feature impact through mean absolute SHAP values across the test partition.
Step 5: Train the model on all possible feature sets to obtain the final prominent feature set.

The Algorithm 2 systematically integrates selected behavioral features extracted through static and dynamic analysis with environmental contexts, such as on-phone charging, Wi-Fi connected, on-phone call, Screen-off, and time-based. The integration of features with context within predefined combination patterns such as (1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2), etc. To capture context-conditional effects by listing standard context × feature interactions (1C × 1F, 1C × 2F, 2C × 1F, 2C × 2F, 3C × 1F, 1C × 3F, etc.), where C is context, and F is feature, yielding 60,000 candidates. These are selected to detect real-world ransomware execution. When 4 or more context features are combined for multi-context triggers, it results in redundant detection. The feature interaction stage significantly expanded the feature space by creating pairwise interaction terms, resulting in a high-dimensional representation containing approximately 60,000 interaction features. To control dimensionality and eliminate redundant interactions, these are ranked with mean absolute Shapley Additive explanations (SHAP) importance. SHAP is employed to estimate the contribution of feature interactions, enabling the selection of top interactions. A cumulative SHAP importance analysis indicated that 600 features captured 99% of the overall feature importance. To further validate the selected feature subset, a systematic K-sweep analysis was conducted across a range of feature subset sizes. The experimental results indicated that performance stabilised beyond this point, suggesting that lower-ranked features offered negligible additional predictive information. As a result, the 600-feature subset was adopted as the context-critical feature subset, ensuring high predictive performance while maintaining interpretability and computational feasibility.

3.5 Classification and Performance Evaluation

Different ML classifiers are used to train the model, such as Extra Trees (ET), Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), Gradient Boosting (GB), decision trees (DT), k-nearest neighbours (KNN), Naive Bayes (NB), and Multi-layer Perceptron (MLP). The ML classifiers utilized in this study were selected for their proven effectiveness and widespread adoption in Android malware detection studies [12,25]. Additionally, both individual and ensemble-based classifiers were employed to comprehensively evaluate the discriminative strength and generalisation performance of the proposed feature set. Models are compared using the different feature sets, such as follows.

- The Original Feature Subset comprise all extracted static and dynamic features without any filtering and contextual enrichment.

- The Selected Feature Subset (SelectFS) employs the feature set selected from ANOVA + RFECV optimization.
- The augmented features enriched the baseline features with contextual triggers such as phone charging, phone calls, Wi-Fi connected, time-based, and screen off.
- The context-critical features (CCF) filtered by SHAP importance values.

The training and testing ratio is 80:20. This configuration used 80% of the allocated data for model training, with the remaining 20% retained for independent evaluation. The standard split ratio ensures enough samples for effective model training while maintaining a representative test set. The random seed is set to 42 for all models. The splitting method is reproducible with random_state, facilitating consistent comparisons across experiments that employ different models and feature selection strategies. Table 4 explains the hyperparameters for all the classifiers.

Table 4: Experimental perameters.

LSanguage	Python 3.12.1
Libraries	Scikit-learn 1.5.1 TensorFlow 2.16.1 Pandas, NumPy, Matplotlib
RF	n_estimators = 200, min_samples_split = 2
ET	n_estimators = 200, criterion = gini, max_depth = None, min_samples_split = 2, bootstrap = False, n_jobs = −1
SVM	RBF kernel with C = 1.0
KNN	n_neighbors = 5, weights = uniform, metric = minkowski (p = 2)
GB	n_estimators = 100, learning_rate = 0.1, max_depth = 3, subsample = 1.0
NB	var_smoothing = 1e−9
LR	penalty = 'l2' and solver = 'liblinear'
DT	criterion = gini, splitter = best, max_depth = None, min_samples_split = 2
MLP	Activation Function: ReLU, Optimizer: Adam (learning_rate = 0.001) Training Parameters: epochs = 20, batch_size = 32 Dropout Rate: 0.3 Input Layer: 141 neurons Hidden Layers [128, 64, 32] neurons Loss Function: Binary cross-entropy

All models were trained on the training data and assessed on a fixed test set using standard metrics such as accuracy, F1 score, precision, recall, and ROC-AUC, thereby avoiding data leakage and ensuring equitable performance comparisons across the four feature selection strategies. To evaluate significant performance differences among classifiers, the Friedman test followed by the Nemenyi post hoc test was conducted at $\alpha = 0.05$.

4 Results and Discussions

4.1 Experimental Details

The McIFAR is implemented on a system with a unified and controlled environment. The detailed description of the experimental environment setup, including system requirements, Android emulator specifications, and software details, is summarised in Table 5.

Table 5: Experimental setup.

Component	Specification	Android Emulator Configuration	
OS	Ubuntu 20.04	Emulator	AVD (x86_64)
CPU	Intel 8 Core i7 processor	Android Version	11 (API 30)
GPU	NVIDIA T400 GPU (4 GB)	RAM/Storage	4 GB/8 GB
RAM	32 GB	User Interaction	Monkey
Androguard Tool	v4.1.2	System call tracing	Strace
Language and Libraries			
	Python	3.12.1	
	Pandas	2.2.0	
	NumPy	1.26.4	
	Scikit-learn	1.5.1	
	Seaborn	0.13.2	
	Shap	0.46.0	
	Matplotlib	3.8.2	

The host system configuration determines the computational environment for feature extraction, training, and evaluation. In contrast, the emulator configuration enables the consistent execution of Android applications within controlled, contextual environments. Additionally, the specified software libraries and their corresponding versions enable reproducible feature engineering and ML experiments across diverse setups.

4.2 Feature Selection Benchmark over Filter-Wrapper Combinations

To identify the best combination of feature selection techniques using the CVFSE algorithm, the proposed approach tested nine various filter-wrapper combinations.

The proposed CVFSE algorithm evaluates multiple combinations of filter-based and wrapper-based feature selection techniques to identify the most discriminative feature subset for Android ransomware detection. The filter methods used include MI, Chi-square, and ANOVA F-score, while the wrapper methods include SFS, RFE, and RFECV. Each filter-wrapper combination was tested using classification metrics such as accuracy, precision, recall, and F1-score without incorporating contextual features. For filter-based methods, features were ranked by score, and the top 50 were selected to retain relevant information while controlling dimensionality. As shown in Fig. 5, multiple classifiers, including RF, LR, SVM, and GB, were evaluated to determine the optimal feature selection pair. Among all combinations, ANOVA+RFECV with the RF classifier achieved the best performance, obtaining 97.19% accuracy along with high precision (96.20%) and recall (96.19%). This comprehensive evaluation identified ANOVA+RFECV as the most effective feature selection combination for subsequent modeling stages.

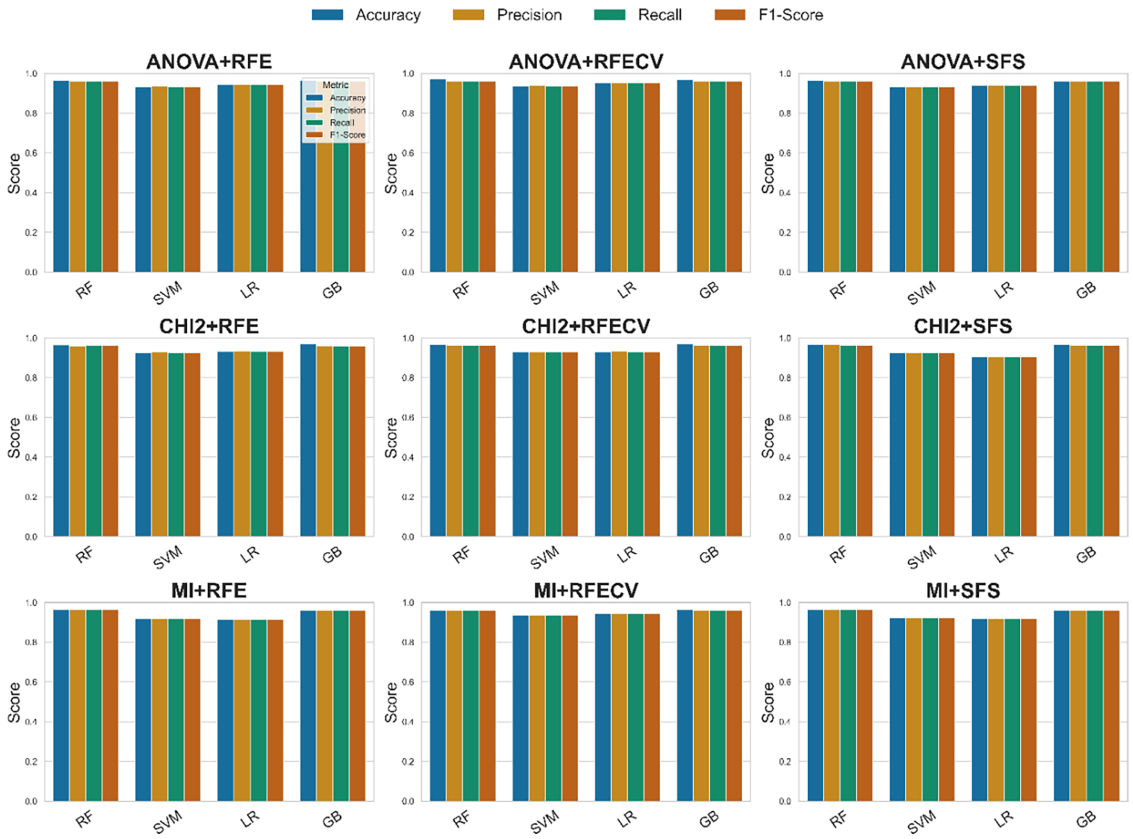

Figure 5: Comparative analysis of different feature selection combinations.

4.3 Best Subset Selection (Baseline Strategy)

According to the benchmark, the ANOVA+RFECV pair achieved accuracy, precision, recall, and F1-score values of 97.19%, 96.20%, 96.19%, and 96.19%, respectively. The resulting feature subset consisted of 38 features, as listed in Table 6, which were significant attributes most closely related to malicious behavior. Interestingly, several features, such as Internet, fstatfs64, CAMERA, and WAKE_LOCK, exhibited benign behavior in isolation when used by social media, navigation, video conferencing, file manager, and media apps. They were retained because they may have synergistic effects when combined with other features and context. All these features are analyzed across different contexts, such as on-phone charging, on-phone calls, Wi-Fi-connected, time-based, and screen-off.

4.4 Contextual Interaction Feature Engineering

To enhance the discriminative power of feature representation, contextual interaction feature engineering was applied by combining static, dynamic, and contextual features to capture deeper behavioral patterns in Android applications. Features such as permissions, intents, system calls, and API calls were analyzed across contexts, including phone charging, phone calls, Wi-Fi connectivity, nighttime conditions, and screen-off states to identify strategic ransomware behaviors. Using the ANOVA+RFECV feature selection method, 38 dominant features were selected and integrated with five contextual conditions through the proposed algorithm. This generated interaction features that revealed conditional malicious behaviors occurring only under specific runtime contexts. In total, 60,325 interaction features were engineered using patterns such as 1C-1F, 1C-2F, 2C-1F, 2C-2F, 3C-1F, and 3C-2F. Since the expanded feature space introduced redundancy and

noise, SHAP importance was used to rank and filter the interaction features, resulting in the selection of the 604 most influential features for the final model.

Table 6: List of ANOVA+RFECV selected features.

S.NO	Selected Features	S. No.	Selected Features
1	features_read	20	features_NrIntActivities
2	features_GET_TASKS	21	features_ACCESS_WIFI_STATE
3	features_fstatfs64	22	features_READ_SMS
4	features_BIND_DEVICE_ADMIN	23	features_socketpair
5	features_KILL_BACKGROUND_PROCESSES	24	features_mkdirat
6	features_WAKE_LOCK	25	features_madvise
7	features_READ_EXTERNAL_STORAGE	26	features_NrIntActivitiesActions
8	features_CHANGE_WIFI_STATE	27	features_getrandom
9	features_getdents64	28	features_READ_PHONE_STATE
10	features_READ_CALL_LOG	29	features_ACCESS_NETWORK_STATE
11	features_MOUNT_UNMOUNT_FILESYSTEMS	30	features_mprotect
12	features_WRITE_EXTERNAL_STORAGE	31	features_Activities
13	features_CAMERA	32	features_INTERNET
14	features_SEND_SMS	33	features_RECEIVE_SMS
15	features_WRITE_SETTINGS	34	features_NrIntReceivers
16	features_NrIntReceiversActions	35	features_NrIntServices
17	features_sendmsg	36	features_NrIntServicesActions
18	features_munmap	37	features_wait4
19	features_READ_CONTACTS	38	features_NrServices

Impact of context

Context-critical ransomware is a type of Android malware that conditionally deploys its malicious payload, e.g., file encryption, device lockdown, or resource takeover, only if certain contextual conditions are met.

Strategic ransomware often delays payload execution until specific contextual conditions, such as charging state, screen-off state, Wi-Fi connectivity, phone calls, or nighttime conditions, are met. By exploiting these context-critical triggers, ransomware can evade traditional detection frameworks that rely on generic execution environments. To detect such hidden behaviors, the proposed CIFO algorithm integrates static features, dynamic runtime events, and contextual triggers to generate interaction-based features. Fig. 6 presents the importance of these context-critical interactions using SHAP analysis. Interaction patterns include 1C-1F, 1C-2F, 2C-1F, 2C-2F, 3C-1F, and 3C-2F, where C denotes context, and F denotes feature. For example, the combination of screen-off state, charging condition, filesystem probing (fstatfs64), and background process termination (KILL_BACKGROUND_PROCESSES) strongly indicates ransomware behavior, as malicious applications often activate payloads only during idle conditions.

The results demonstrate that contextual factors significantly influence ransomware behavior. The proposed interaction-based feature representation, combining ANOVA + RFECV selected features, contextual conditions, and SHAP-ranked interaction features, outperformed baseline and context-agnostic models. The final classification model utilized 647 features, including behavioral, contextual, and interaction-based attributes.

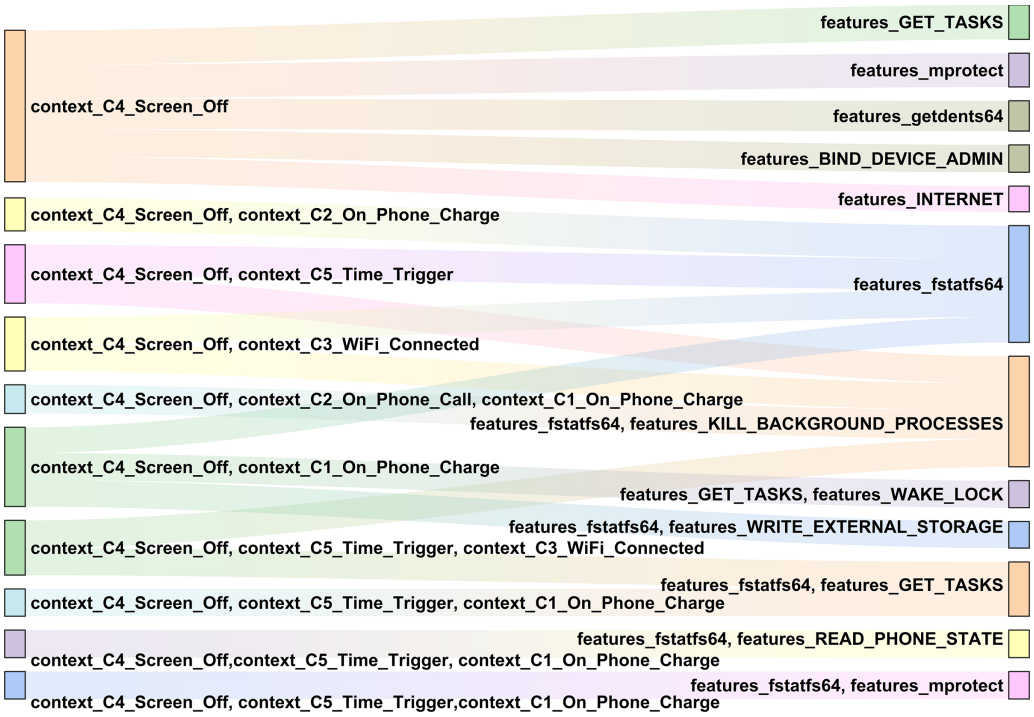

Figure 6: Comparative analysis of interaction features.

4.5 Classifier Evaluation

To obtain clearer insights into the proposed comprehensive feature set, which integrates baseline-selected, contextual, and interaction-based attributes, the performance of various ML classifiers was assessed using standard evaluation metrics, including accuracy, precision, recall, F1-score, and ROC curve. Fig. 7 compares four feature strategies: OFS, SelectFS, augmented features (SelectFS + context), and the CCF (interaction features).

Nine classifiers, including GB, ET, RF, DT, KNN, NB, SVM, LR, and MLP, were evaluated to validate performance across different feature engineering strategies. The results show a clear improvement as the feature engineering process evolves toward the proposed framework. Strategies that incorporated CVFSE-based dominant feature selection, contextual information, and interaction features generated by the CIFO algorithm consistently achieved higher performance. Ensemble tree-based models, particularly ET, RF, and GB, outperformed conventional classifiers due to their ability to capture complex feature interactions. Compared to the baseline feature set, the proposed feature representation improved accuracy by up to 4% and recall by 5%, highlighting the effectiveness of contextual and interaction-based feature engineering. Among all classifiers, ET achieved the best performance with 99.48% accuracy, 99.47% recall, 99.47% precision, and 99.47% F1-score. High recall is especially important in ransomware detection, as misclassifying ransomware as benign can have severe consequences. The strong performance of ET, GB, and RF demonstrates the proposed framework's ability to detect evolving and evasive ransomware behaviors, including potential zero-day threats.

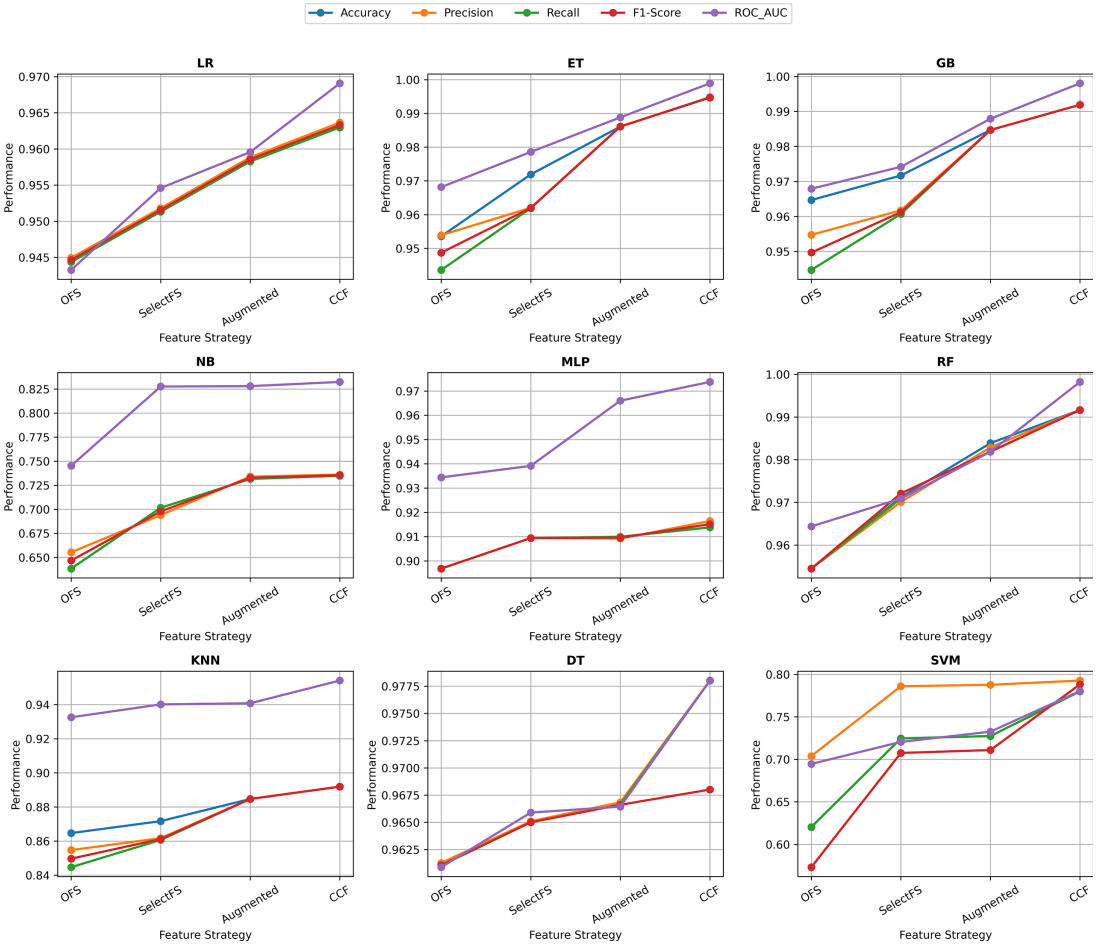

Figure 7: Comparative evaluation of the classifier's performance under various feature selection strategies.

4.6 Ablation Study

To determine the individual contribution of essential components within the McIFAR detection system, an ablation study was conducted by sequentially adding each component and analyzing the resulting improvement in accuracy using ET classification. The ablation study reveals that the joint integration of hybrid feature selection using the CVFSE algorithm (CVFSE), context-critical feature augmentation (context), and SHAP-based interaction feature construction (SHAP-based interaction pruning). Table 7 summarises the comprehensive results of ET classifiers across all performance metrics, including accuracy, recall, F1-Score, and precision.

The experimental results validate the effectiveness of the proposed feature engineering pipeline. Using raw features, the model achieved 95.35% accuracy and 94.87% F1-score. After applying the proposed CVFSE feature selection algorithm, performance improved to 97.19% accuracy and 96.19% F1-score, showing that hybrid feature selection effectively reduces feature-space noise. By incorporating contextual features, the model improved to 98.60% accuracy and 98.61% F1-score, highlighting the importance of contextual information for detecting ransomware behaviour. The complete framework, integrating CVFSE, contextual features, and SHAP-based interaction feature pruning, achieved the best performance with 99.48% accuracy and 99.47% F1-score. Overall, the proposed approach improved accuracy by 4.13% and F1-score by 4.60%

compared to the raw-feature model, demonstrating the effectiveness of interaction-based feature engineering for Android ransomware detection.

Table 7: Comparative analysis of models across different feature selection strategies.

Feature-Based Model	CVFSE	Context	SHAP Interaction Pruning	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Raw features	–	–	–	95.35	95.39	94.35	94.87
Baseline	✓	–	–	97.19	96.20	96.19	96.19
Augmented	✓	✓	–	98.60	98.61	98.60	98.61
Proposed	✓	✓	✓	99.48	99.47	99.47	99.47

4.7 Comparative Analysis of Statistical Results

4.7.1 Friedman Test

The Friedman test was conducted to evaluate F1-Scores derived from repeated stratified cross-validation (5 × 5 folds). In repeated stratified cross-validation, the 5-fold cross-validation was performed five times to achieve more robust and reliable performance estimates. In this study, 9 classifiers were employed across 25 folds. The Friedman test yielded a statistic of 180.198 with a p-value of 9.35×10^{-35}, indicating that the differences among classifiers are statistically significant. Table 8 illustrates the mean F1-Score and associated average ranking of each classifier computed across multiple folds.

Table 8: Friedman test results.

Classifier	Mean F1-Score	Average Rank
ET	99.47	1.44
GB	99.19	2.1
RF	99.16	2.94
MLP	97.78	4.18
LR	96.90	4.64
DT	96.80	6.46
KNN	96.41	6.52
SVM	95.30	7.72
NB	73.82	9

The ranking is from lowest to highest order; the ET classifier has an average rank of 1.44, which is considered the best. The obtained ranks were subsequently averaged and applied in the Friedman test to evaluate statistical significance. Based on average rankings, ensemble models such as ET, GB, and RF outperformed other models.

4.7.2 Nemenyi Post-Hoc Test

Since the Friedman test indicates statistically significant differences among the evaluated classifiers, the Nemenyi post hoc test is used to identify the specific pairs that differ significantly. The Nemenyi post hoc test further examines these differences using the critical difference (CD) diagram shown in Fig. 8.

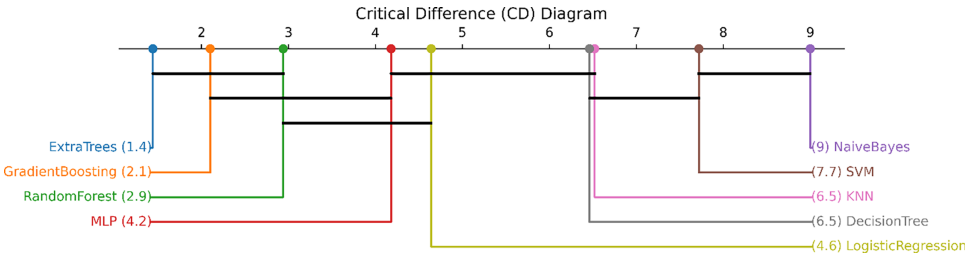

Figure 8: Critical differences among classifiers.

The CD diagram indicates that classifiers such as Naive Bayes, SVM, and KNN show significant differences compared to the top-performing classifiers. However, the differences among top ensemble models are not consistently statistically significant, suggesting comparable performance within this group.

4.8 Comparative Analysis

As demonstrated in the statistical analysis, ET was selected as the optimal classifier for the proposed framework. The McIFAR achieved superior performance across all evaluation metrics when using the ET model. To validate the McIFAR against existing state-of-the-art malware detection methods, a comparative analysis is presented in Table 9.

Table 9: Comparative results with existing detection frameworks.

Reference	Det	S	D	H	Feature Selection Method			Accuracy (%)	Recall (%)	F1-Score (%)	Precision (%)
					F	W	E				
[12]	✓	✓	×	×	✓	×	✓	96.27	97.27	96.26	96.22
[13]	✓	×	×	✓	×	✓	×	98.8	98	98	98
[19]	✓	✓	×	×	✓	×	×	98.08	98.21	98.12	98.46
[24]	✓	×	✓	×	✓	✓	×	98.34	98.5	98.27	98.1
[25]	✓	×	×	✓	×	×	×	92.93	97	93.90	91
[31]	✓	×	✓	×	✓	×	×	97.8	97	97	97
Proposed	✓	×	×	✓	✓	✓	×	99.48	99.47	99.47	99.47

Note: Abbrev: Det-Detection, D-Dynamic, E-Embedded-based selection, F-Filter-based selection, H-Hybrid, S-static, W-Wrapper-based selection.

This provides a comprehensive comparison of existing approaches across analysis type (static, dynamic, and hybrid), feature selection techniques, and classification performance metrics, including accuracy, precision, recall, and F1-score. McIFAR outperformed [12,13,19,24,25,31] across all evaluation metrics. In summary, the results indicate that both feature selection techniques and analysis types significantly influence the effectiveness of ransomware detection systems.

5 Conclusion and Future Scope

This study presents McIFAR, an Android ransomware detection framework that integrates in-context emulation, hybrid feature selection, and interaction-based feature engineering. The framework captures hidden malicious behaviors often missed by conventional static and dynamic analysis methods by simulating realistic execution conditions within malware sandboxing. The proposed methodology combines the CVFSE algorithm for dominant feature selection with the CIFO algorithm for modeling feature-context interactions. Experimental results demonstrate that the final feature set, consisting of selected features, contextual information, and interaction features, consistently outperformed baseline and augmented feature strategies.

Among all classifiers, the ET model achieved the best performance with 99.48% accuracy and 99.47% F1-score. Statistical analysis using the Friedman and Nemenyi post hoc tests further confirmed the superior performance of the ET classifier. Despite these promising results, future work can improve the framework by incorporating more diverse contextual scenarios, realistic user interactions, and real-device deployment. Additional contextual conditions, such as geolocation-based behaviors, and scalable approaches, such as federated learning, could further enhance robustness, generalization, and privacy-preserving ransomware detection in real-world environments.

Acknowledgement: This work has been supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2026R701), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Funding Statement: This work was funded by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2026R701), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Author Contributions: Sonam Jain: Conceptualization, Formal analysis, Methodology, Software, Validation, Investigation, Data curation, Writing—original draft; Tanya Gera: Conceptualization, Formal analysis, Methodology, Investigation, Supervision, Writing—original draft; Rupali Gill: Formal analysis, Supervision, Validation, Software, Resources; Afnan Almegren: Resources, Project administration, Editing, Reviewering and editing; Ateeq Ur Rehman: Software, Editing, Reviewering and editing, Investigation; Salil Bharany: Conceptulization, Data curation, Editing, Reviewering and editing. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The datasets generated and analyzed during the current study are available from the corresponding authors upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mobile operating system market share worldwide [Internet]. StatCounter Global Stats. [cited 2026 Mar 15]. Available from: https://gs.statcounter.com/os-market-share/mobile/worldwide.
2. Sharma S, Kumar R, Rama Krishna C. A survey on analysis and detection of Android ransomware. Concurr Comput. 2021;33(16):e6272. doi:10.1002/cpe.6272.
3. Kivva A. IT threat evolution in Q3 2025. Mobile statistics [Internet]. Kaspersky. 2025 [cited 2026 Apr 3]. Available from: https://securelist.com/malware-report-q3-2025-mobile-statistics/118013/.
4. Beaman C, Barkworth A, Akande TD, Hakak S, Khan MK. Ransomware: recent advances, analysis, challenges and future research directions. Comput Secur. 2021;111(6):102490. doi:10.1016/j.cose.2021.102490.
5. Global app stores ranked by available apps 2026 [Internet]. Statista. [cited 2026 Apr 3]. Available from: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.
6. Kostka C. Android users increasingly targeted by ransomware [Internet]. Ransomware.org; 2022 [cited 2026 Mar 20]. Available from: https://ransomware.org/blog/android-users-increasingly-targeted-by-ransomware/.
7. Andronio N, Zanero S, Maggi F. HelDroid: dissecting and detecting mobile ransomware. In: Research in attacks, intrusions, and defenses. Cham, Switzerland: Springer; 2015. p. 382–404. doi:10.1007/978-3-319-26362-5_18.
8. Chen J, Wang C, Zhao Z, Chen K, Du R, Ahn GJ. Uncovering the face of Android ransomware: characterization and real-time detection. IEEE Trans Inf Forensics Secur. 2018;13(5):1286–300. doi:10.1109/TIFS.2017.2787905.
9. Smmarwar SK, Gupta GP, Kumar S. Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: a comprehensive review. Telemat Inform Rep. 2024;14:100130. doi:10.1016/j.teler.2024.100130.
10. Oz H, Aris A, Levi A, Uluagac AS. A survey on ransomware: evolution, taxonomy, and defense solutions. ACM Comput Surv. 2022;54(11s):1–37. doi:10.1145/3514229.

11. Martín A, Hernandez-Castro J, Camacho D. An in-depth study of the jisut family of Android ransomware. IEEE Access. 2018;6:57205–18. doi:10.1109/ACCESS.2018.2873583.
12. Jain S, Goyal H, Arora A, Kumar D. EnFeSTDroid: ensembled feature selection techniques based Android malware detection. Comput Electr Eng. 2026;129(1):110763. doi:10.1016/j.compeleceng.2025.110763.
13. Sharma S, Prachi, Chhikara R, Khanna K. A novel feature selection technique: detection and classification of Android malware. Egypt Inform J. 2025;29(1):100618. doi:10.1016/j.eij.2025.100618.
14. Hossain MA, Hasan T, Ahmed F, Cheragee SH, Kanchan MH, Haque MA. Towards superior Android ransomware detection: an ensemble machine learning perspective. Cyber Secur Appl. 2025;3(11):100076. doi:10.1016/j.csa.2024.100076.
15. Singh N, Tripathy S. It's too late if exfiltrate: early stage Android ransomware detection. Comput Secur. 2024;141(8):103819. doi:10.1016/j.cose.2024.103819.
16. Mohanraj A, Sivasankari K. Android traffic malware analysis and detection using ensemble classifier. Ain Shams Eng J. 2024;15(12):103134. doi:10.1016/j.asej.2024.103134.
17. Sharma Y, Arora A. IPAnalyzer: a novel Android malware detection system using ranked intents and permissions. Multimed Tools Appl. 2024;83(33):78957–9008. doi:10.1007/s11042-024-18511-6.
18. Sharma S, Krishna CR, Kumar R. RansomDroid: forensic analysis and detection of Android ransomware using unsupervised machine learning technique. Forensic Sci Int Digit Investig. 2021;37:301168. doi:10.1016/j.fsidi.2021.301168.
19. Boukhamla AZE, Verma A. HyDroid: android malware detection using network flow combined with permissions and intent filter. Int J Mob Commun. 2023;22(1):70–91. doi:10.1504/ijmc.2023.131799.
20. Baghirov E. Comprehensive framework for malware detection: leveraging ensemble methods, feature selection and hyperparameter optimization. In: Proceedings of the 2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT); 2023 Oct 18–20; Baku, Azerbaijan. p. 1–5. doi:10.1109/AICT59525.2023.10313179.
21. Mahdavifar S, Alhadidi D, Ghorbani AA. Effective and efficient hybrid Android malware classification using pseudo-label stacked auto-encoder. J Netw Syst Manag. 2021;30(1):22. doi:10.1007/s10922-021-09634-4.
22. Smmarwar SK, Gupta GP, Kumar S. A hybrid feature selection approach-based Android malware detection framework using machine learning techniques. In: Cyber security, privacy and networking. Singapore: Springer; 2022. p. 347–56. doi:10.1007/978-981-16-8664-1_30.
23. Santosh Jhansi K, Chakravarty S, Ravi Kiran Varma P. A two-tier fuzzy meta-heuristic hybrid optimization for dynamic Android malware detection. SN Comput Sci. 2022;4(2):117. doi:10.1007/s42979-022-01523-0.
24. Iqbal MJ, Aurangzeb S, Aleem M, Srivastava G, Lin JC. RThreatDroid: a ransomware detection approach to secure IoT based healthcare systems. IEEE Trans Netw Sci Eng. 2023;10(5):2574–83. doi:10.1109/TNSE.2022.3188597.
25. Almomani I, Qaddoura R, Habib M, Alsoghyer S, Al Khayer A, Aljarah I, et al. Android ransomware detection based on a hybrid evolutionary approach in the context of highly imbalanced data. IEEE Access. 2021;9:57674–91. doi:10.1109/ACCESS.2021.3071450.
26. Gera T, Singh J, Mehbodniya A, Webber JL, Shabaz M, Thakur D. Dominant feature selection and machine learning-based hybrid approach to analyze Android ransomware. Secur Commun Netw. 2021;2021(4):7035233. doi:10.1155/2021/7035233.
27. Abuthawabeh M, Mahmoud K. Enhanced Android malware detection and family classification, using conversation-level network traffic features. Int Arab J Inf Technol. 2020;17(4A):607–14. doi:10.34028/iajit/17/4a/4.
28. Alzaylaee MK, Yerima SY, Sezer S. DL-Droid: deep learning based android malware detection using real devices. Comput Secur. 2020;89(5):101663. doi:10.1016/j.cose.2019.101663.
29. Chavan N, Di Troia F, Stamp M. A comparative analysis of android malware. In: Proceedings of the 5th International Conference on Information Systems Security and Privacy; 2019 Feb 23–25; Prague, Czech Republic.
30. Bibi I, Akhunzada A, Malik J, Ahmed G, Raza M. An effective Android ransomware detection through multi-factor feature filtration and recurrent neural network. In: Proceedings of the 2019 UK/China Emerging Technologies (UCET); 2019 Aug 21–22; Glasgow, UK. p. 1–4. doi:10.1109/ucet.2019.8881884.

31. Huda S, Islam R, Abawajy J, Yearwood J, Hassan MM, Fortino G. A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection. Future Gener Comput Syst. 2018;83(2):193–207. doi:10.1016/j.future.2017.12.037.
32. Zhu HJ, You ZH, Zhu ZX, Shi WL, Chen X, Cheng L. DroidDet: effective and robust detection of android malware using static analysis along with rotation forest model. Neurocomputing. 2018;272(5):638–46. doi:10.1016/j.neucom.2017.07.030.
33. Guerra-Manzanares A, Bahsi H, Nõmm S. KronoDroid: time-based hybrid-featured dataset for effective android malware detection and characterization. Comput Secur. 2021;110:102399. doi:10.1016/j.cose.2021.102399.
34. Augello A, De Paola A, Lo Re G. M2FD: mobile malware federated detection under concept drift. Comput Secur. 2025;152(7):104361. doi:10.1016/j.cose.2025.104361.
35. Desnos A, Gueguen G. Android: from reversing to decompilation. In: Proceedings of the Black Hat Abu Dhabi 2011; 2011 Dec 12–15; Abu Dhabi, United Arab Emirates. p. 77–101.
36. Mahindru A, Sangal AL. FSDroid:-a feature selection technique to detect malware from Android using machine learning techniques: FSDroid. Multimed Tools Appl. 2021;80(9):13271–323. doi:10.1007/s11042-020-10367-w.
37. Hoque N, Bhattacharyya DK, Kalita JK. MIFS-ND: a mutual information-based feature selection method. Expert Syst Appl. 2014;41(14):6371–85. doi:10.1016/j.eswa.2014.04.019.
38. Alazab M. Automated malware detection in mobile app stores based on robust feature generation. Electronics. 2020;9(3):435. doi:10.3390/electronics9030435.
39. Chen YJ, Kuo WH, Tsai SY, Chen JL, Chen YH, Xu WZ. Artificial intelligence hybrid learning architecture for malware families classification. In: Proceedings of the 2019 21st International Conference on Advanced Communication Technology (ICACT); 2019 Feb 17–20; PyeongChang, Republic of Korea. p. 503–10.
40. Shabtai A, Kanonov U, Elovici Y, Glezer C, Weiss Y. "Andromaly": a behavioral malware detection framework for android devices. J Intell Inf Syst. 2012;38(1):161–90. doi:10.1007/s10844-010-0148-x.