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

A PERMISSION-BASED MACHINE LEARNING MODEL FOR ANDROID RANSOMWARE DETECTION

Muhammad Akram Khan[*1], Muhammad Nadeem[2], Jan Muhammad[3]

[*1,3]*Faculty of ICT, BUITEMS, Quetta, Pakistan*
[2]*Department of CIS, HCT, Abu Dhabi, UAE*

[*1]mohammad.akram@buitms.edu.pk, [2]mnadeem@hct.ac.ae, [3] jan.muhammad@buitms.edu.pk

DOI: https://doi.org/10.5281/zenodo.21818735

Keywords
Android Ransomware, Machine Learning, Permissions, Malware Detection, Static Analysis, Logistic Regression, WEKA

Article History
Received: 27 January 2026
Accepted: 12 March 2026
Published: 26 March 2026

Corresponding Author: *
Muhammad Akram Khan

Abstract
Ransomware is among the most disruptive types of malware that present a threat to the Android ecosystem, blocking access to a victim's files or device with the aim of extorting ransom in return. The permission list provided in an Application's manifest–available without code execution or dynamic instrumentation, but requiring the application to explicitly say what sensitive capabilities it will use–is a lightweight, static and computationally inexpensive signal for distinguishing malicious intent from legitimate functionality. In this research, a balanced dataset consisting of 995 benign and 995 ransomware application samples is decompiled and analyzed to extract 365 distinct attributes based on permissions, and a machine learning model that detects the ransomware is developed by using these attributes. The data-mining suite (Weka) was used to create and test four classifiers: Logistic Regression, Random Tree, Decision Table and Bayes Network, using the 10-fold cross validation method. The overall accuracy for Logistic Regression was the highest (97.69%), followed by Bayes Network (96.88%), Random Tree (96.53%) and Decision Table (95.73%). A second comprehensive permission-level analysis also pinpoints RECEIVE_BOOT_COMPLETED, SYSTEM_ALERT_WINDOW and MOUNT_UNMOUNT_FILESYSTEMS as the most powerful permissions that distinguish between benign and ransomware activities. The findings show that permission-based features provide high accuracy ransomware classification and can be a viable first-line defense in resource-limited pre-installation detection pipelines, like app-store vetting.

1. Introduction

1.1 Background and Motivation

Ransomware is a form of malicious software that prevents a user from accessing their own computer or files, usually by encrypting the files or locking the screen, until a ransom is paid [1]. Ransomware says its name, and it's often not hidden in such a way that it extracts data or resources from the victim, but rather it is designed to be discovered and their victims are expected to pay to have it reversed. Since the first documented ransomware variants appeared, attackers have been constantly evolving and diversifying their ransomware attacks, gradually improving both the methods used to encrypt data and the social engineering techniques used to spread the malware and the ransom demands [2]. As ransomware expands, so too have mobile handsets and Internet-of-Things

endpoints been entangled in the same extortion economy, notes Yaqoob et al., that was once limited to targeting desktop computers [3]. Ransomware on mobile is not theoretical or diminishing; reports that monitor ransomware attacks specifically on mobile devices are tracking them constantly for both individuals and organizations [4].

Ransomware can wreak very dangerous and often irreparable damage if it infects a machine and in many cases documented attacks, it is only when the ransom is paid that the damage is not considered "irreparable". One frequently cited case was a Tennessee county sheriff's office that, after being infected by ransomware, chose to pay a ransom in bitcoin, as the only option for recovery [5]. The ease of production and distribution of ransomware has only made things worse: In 2016, McAfee researcher J. Walter unraveled the emergence of ransomware-as-a-service toolkits, which enable attackers with low technical abilities to initiate ransomware attacks, and by mid-2017, McAfee Labs estimated that the total number of ransomware samples had exceeded 10 million. Industry retrospectives on ransomware as a cyberattack category also outline a trend of growth in ransomware attack size, sophistication and commercialization over the last ten years [7], and security-vendor analyses of the most damaging ransomware campaigns over the last few years confirm the impact of ransomware in terms of financial and operational damage [8].

1.2 Android as a Primary Target

Ransomware has tracked Android's market share – and it's moving to mobile platforms. Android OS has been on a gradual growth path in the global mobile operating system market since early 2012, when it accounted for approximately 27% of the market, through to late 2020 when it has clearly become the dominant operating system, accounting for more than 73% of the market [9]. This domination has made Android to be the most appealing platform for mobile ransomware, as a single successful ransomware strain will be able to reach an immense number of mobile devices. Reports on the number of mobile ransomware attacks indicate that the volume of mobile ransomware infections has exploded over the past few years, with the number of infections reported in each year since the attacks were introduced on mobile platforms showing significant increases over time as attackers modified their attack techniques and distribution methods, such as in [10]. Industry advice for consumers also suggests that ransomware is a current threat for consumers of mobile devices, and not a past event [11].

The permission list specified in an application's manifest represents a potential and lightweight detection signal because ransomware requires to demand a fairly standard set of permissions in order to be persistent, lock the screen, or access storage. This paper directly explores that possibility.

Prevention is better than detection, as ransomware can often cause permanent damage to the encrypted files or lock the device, if it is detected after it has encrypted files or locked devices, it may be too late to restore anything without paying the ransom. Static analysis techniques – those that analyze an application's code, resources and manifest without running the application – are appealing for this purpose because they are fast, scalable and are not sandboxed. The permission list is one of the artifacts that can be obtained in a static analysis: the Android OS requires that apps explicitly ask for access to sensitive functionality like files, device administration, SMS, internet connection, etc. and ransomware apps typically ask for a unique set of permissions to extort.

1.3 Contributions

This paper presents the following contributions: (1) a feature set, based on a large, independent, balanced collection of 995 ransomware and 995 benign Android apps, on which ransomware and benign apps can be compared; (2) rigorous (10-fold) cross-validation testing of four machine learning classifiers on the feature set using the above-mentioned collection of ransomware and benign applications; (3) a detailed analysis of the permissions on the feature set, identifying which permissions are most discriminative in distinguishing ransomware from benign apps, on a permission-by-permission basis; and (4) a

discussion of the practical implications of these results for lightweight, "pre-installation" ransomware screening.

1.4 Paper Organization

The rest of this paper is organized as follows. In Section 2, the authors offer background information about cyber extortion, ransomware taxonomy, the evolution of Android ransomware, and the anatomy of a typical Android ransomware attack. In Section 3, we discuss previous ransomware detection research in static and dynamic, hybrid and permission-based ransomware detection. The construction of the dataset, feature extraction and methodology of machine learning is described in section 4. Experimental results are given in Section 5. The findings of the study, their implications for practice and study limitations are discussed in section 6. The paper is concluded in Section 7 and directions for future work are given.

2. Background

2.1 Cyber Extortion and Ransomware

Ransomware is a form of cyber extortion: a type of online crime in which an individual or organization is threatened with harm, usually the denial or exposure of its data, unless a ransom is paid [1]. Cyber extortion typically comes in two types: ransomware attacks are those where the attacker threatens to disrupt a target's Internet service if the attacker is not paid, and distributed denial-of-service (DDoS) attacks are those where the attacker threatens to overload a target's Internet service if the attacker is not paid. Ransomware criminals typically set a short timeframe for the victim, usually raising the ransom if the victim doesn't pay within an initial period and even threatening to delete the victim's data if the deadline is missed altogether [1].

2.2 Locker versus Crypto Ransomware

In the ransomware family there are two major strains identified. The victim is no longer able to use the device, and Locker ransomware usually includes a message that covers the entire screen and can't be closed, usually in the guise of a notification from law enforcement about imaginary crimes. Crypto ransomware, on the other hand, does not prevent the device from being used in other ways, but it encrypts certain files or directories with strong cryptographic algorithms, and only releases the encrypted files if a decryption key is received from the attacker when payment is made. Very early examples of this crypto variant include the CryptoLocker campaign that targeted more than 500,000 people using malware to encrypt files, but disguised as law-enforcement notices demanding ransom for “cybercrimes” [12]. An independent report on the evolution of malware, from the same timeframe as the one above, confirms that crypto-style file-encrypting ransomware quickly became one of the most prevalent malware families seen in the wild on both mobile and desktop devices [13].

2.3 Evolution of Android Ransomware

The first reported Android ransomware appeared in mid-2013, it had a locker style and was spread via a drive-by download mechanism in which malicious advertising, or 'malvertising', was used to trick the user into visiting a malicious website where an automatic download of the ransomware was triggered without any deliberate action from the user [14]. The first ever Android crypto-ransomware, also known as Simplocker, first emerged in 2014, encrypting popular types of external storage files and demanding a ransom in prepaid voucher services; although the ransomware predated many of the more advanced and numerous that came later, it is estimated that it infected around 150,000 Android users [3]. Mobile ransomware guidance from consumers and security experts has been consistent since the beginning of the year, with the basic detection and payment scheme of encrypting or locking a device and filling the victim with an untraceable payment method primarily being bitcoin [15]. Industry hindsight reports of the evolution of mobile ransomware outline a variety of family-specific innovations discussed in Section 2.4 [16]. The number of newly discovered ransomware families targeting Android had also risen significantly by the end of 2017, as seen above, with the ransomware-as-a-service model growing more popular with less sophisticated attackers,

attributable to the platform's rising popularity and the ease with which to access ransomware building toolkits.

2.4 Notable Android Ransomware Families

One example of the many ransomware families that have been documented on Android are the following: In 2014, ScarePakage was first detected in the United States and presented a fake notice from the FBI alleging criminal activity in an attempt to collect a ransom [16]. In September 2014, Russian researchers discovered Android Locker, which gained device administrator privileges by pretending to be a system update and then locked the device using a remote command-and-control signal [16]. In October 2014, Worm Koler was discovered and spread via text messages that included a shortened URL, which when activated would show a fake law enforcement message asking for a payment via a prepaid voucher service [16]. The same ransomware strain used in 2010 extorted ransom victims by showing their IP address and carrier details on top of an alleged criminal message, that appeared to be a pornographic application [16]. SimpLocker, which was discovered in 2015, was spread using unverified application-download Web sites disguised as a flash-player update, and encrypted victims' files while presenting a message from a national security agency [16]. Also discovered in 2015, LockerPin also acquired administrative access and modified the device's screen-lock PIN to one that can't even be guessed by the attacker, making the device inaccessible except through a factory reset even if the ransom is paid. This development was continued by later families. Shortly after the WannaCry outbreak on desktop machines in 2017, WannaLocker also emerged disguised as an extension of a popular mobile game, and surprisingly required only a small amount of money rather than a large ransom [17]. Discovered in 2017, DoubleLocker was a device lock that also encrypted files, and was remarkable for being able to exfiltrate banking information through a misuse of Android accessibility features [17]. In 2014, a 2014 variant, dubbed Koler, fooled victims into installing a malicious police notice that demanded payment [17]. Instead of encrypting the victim's files, however, LeakerLocker threatened to expose their personal text messages, photos and web-browsing history, making it a newer member of the family that emphasized extortion over denial of access [17]. The ransomware families that followed later on showed the many techniques and methods of the ransomware locker style which the Svpeng family had already refined. Combined with other known campaigns, these campaigns represent a significant proportion of the worst ransomware attacks in recent years [8].

2.5 Anatomy of an Android Ransomware Attack

Even though there are differences between the families, most attacks with Android ransomware follow the typical stages of deployment, installation, command and control, destruction, and extortion [19]. Phishing emails, fake application updates, third-party app stores or security or media player ads are common methods of deployment, frequently cloaked as a valid security, media player or utility. During installation, the application tries to acquire elevated privileges, most commonly device-administrator privileges that it can use to block the uninstallation process and enforce lock-screen policies. The application will often communicate with a remote command and control server after installation to report that it has been installed, and to get or confirm an encryption key (with crypto-ransomware variants). In the destruction phase, the ransomware performs its main functionality, such as Simplocker using AES encryption on files on external storage, while hiding locally available backup points, compelling the victim to back up data with the attacker [20]. Lastly, in the extortion stage, the ransomware will show a ransom demand - often in the form of a legitimate-looking law enforcement notice - and guide the victim to an untraceable payment method, typically a bitcoin wallet or prepaid voucher service, which renders it more difficult for investigators to identify the transaction and its perpetrator [15].

The five phases is one reason why many of the permissions are found repeatedly in ransomware samples as described in detail in Section 5. RECEIVE_BOOT_COMPLETED helps the

malicious application start itself over after a system reboot, device-administration permissions support the installation phase by making the malicious application harder for the user to remove, and SYSTEM_ALERT_WINDOW helps the extortion phase by making it difficult to dismiss the payment demand.

2.6 Android's Permission Model and its implications for payment anonymity.

Android's permission system mandates that apps specify what system capabilities they will use when they are installed or the first time they run. The purpose of this design is to be able to see what an application can do and to control what it can do; the design is also detectable before execution, a point that this paper takes advantage of for detection purposes. One of the other related problems that defenders will face is the inability to track ransom payments after they are made. Previous research on the evaluation of AID tools has shown that observability of the underlying signal on which any detection or attribution system relies is the key component, which makes the prevention and pre-execution detection approach the most feasible way to reduce damages in the ransomware context [21] - a view that is echoed by researchers specifically studying proactive crypto-ransomware detection and prevention methods [22].

3. Related Work

There is a large number of studies that have investigated static, dynamic and hybrid analysis approaches and different machine learning classifiers to detect the Android ransomware. In this section, literature will be reviewed that is directly relevant to the permission based approach that will be used in this paper, sorted by the analysis technique.

3.1 Static Analysis Approaches

Static analysis is very fast and scalable as an application is not run, but may be susceptible to code obfuscation. HelDroid is an influential static detection system proposed by Mercaldo et al., which consists of three detectors: threatening-text detector, encryption detector, and screen-locking detector to detect three parts of the extortion mechanism described in Section 2.5 [23]. Ameer's doctoral research on detection of ransomware across Android platforms specifically focused on the ability of feature-based and static detectors to withstand adversarial evasion attacks, in which attackers will deliberately alter the requested features to evade detection and will be able to evade detection if the permissions are considered. This concern is directly tackled in Section 7. Discussion of future work.

3.2 Dynamic Analysis Approaches

Dynamic analysis runs the application under instrumented conditions to see what happens during execution, and can detect malicious behavior which static analysis may not. Chen et al. proposed a real-time detection technique based on user-interface widgets and finger-movement patterns and reported that it was not effective for samples which are repackaged, as they performed additional compression operations on the encrypted files along with the encryption [24]. Song, Kim, and Lee suggested a complementary dynamic prevention approach that continuously monitors resource usages (such as processor I/O, memory I/O, and storage I/O) of running processes, marking processes with anomalous I/O patterns as suspicious and, if needed, terminating them, and saving data in marked processes for future detection [25].

3.3 Hybrid Analysis Approaches

Hybrid approaches are a mixture of static and dynamic methods to counterbalance each other. A static analysis framework called DNA-Droid was presented by Gharib and Ghorbani, which rapidly analyzes an application and only passes to dynamic monitoring if the static analysis test identifies it as suspicious, while still identifying malicious samples early in the process, and saving the cost of dynamic monitoring for truly ambiguous samples [26]. Alzahrani et al. introduced a new ransomware detection system, RanDroid that uses structural similarity scores to extract similarity between the layout of the application and a ransomware application corpus and resource similarity scores to extract similarity between the

resources of the applications, along with dynamic capture of the extortion messages on the screen [27]. Ferrante et al. have proposed a two-stage hybrid classification approach which first uses static analysis using opcode frequency and then uses dynamic analysis based on cpu, memory, network and system call statistics when the static analysis is inconclusive, with ransomware detection accuracy of up to 99.8% and false-positive rates of less than 4% [28]. Nieuwenhuizen also suggested a behaviour-based approach to ransomware detection, noting that the attackers are continuously changing ransomware versions and samples, relying on the typical run-time behaviour to achieve resilience [29].

3.4 Permission- and Machine-Learning-Based Approaches

The most similar to the one taken in this paper is a method of detection of locker-ransom Android malware spread via Chinese social networks, developed by Su et al. that uses the classifiers Support Vector Machine, Decision Tree, and Logistic Regression to a set of 301 ransomware samples, relying on static analysis [30]. Sharma et al. proposed a Naive Bayes classifier-based enhanced RansomwareElite application that detects ransomware by analyzing an application's code for threatening text, embedded images or suspicious files along with the permissions that the application requests, tested on nine sample applications on 48 Android devices [31]. The results of the analysis of benign and ransomware applications presented by Alsoghyer and Almomani [32] provide clear motivation for both the feature selection and the permission-combination analysis used in Section 5.2 of this paper, as they showed that the combination of permissions, and not any single permission, is the strongest indicator of whether an application is benign or ransomware. The work proposed by Yang et al., which included static features extraction and classification for ransomware detection and analysis, provided a significant portion of the ransomware hash corpus used by later researchers, including the construction of the ransomware dataset described in Section 4 of this paper [33].

3.5 Summary of Related Studies

Table 1. Summary of related Android ransomware detection studies.

Study	Feature Type / Approach	Classifier(s)	Dataset Size
Mercaldo et al. (HelDroid)	Threatening text, encryption, locking detectors (static)	Rule-based detectors	Not classifier-based
Chen et al. (RansomProber)	UI-interaction analysis (dynamic)	Heuristic real-time detector	Field-deployed samples
Gharib & Ghorbani (DNA-Droid)	Static triage + dynamic profiling (hybrid)	Profile-matching framework	Not specified
Alzahrani et al. (RanDroid)	Image/string similarity + dynamic capture (hybrid)	Similarity-scoring framework	Not specified
Ferrante et al.	Opcode frequency (static) + system-call statistics (dynamic)	Classifier ensemble (hybrid)	Not specified
Su et al.	Static analysis	SVM, Decision Tree, Logistic Regression	301 ransomware samples
Sharma et al. (RansomwareElite)	Code + permission inspection	Naive Bayes	9 apps / 48 devices
Alsoghyer & Almomani	Permissions only (static)	Multiple ML classifiers	~500 benign / ~500 ransomware
This paper	Permissions only (static)	Logistic Regression, Random Tree, Decision Table, Bayes Network	995 benign / 995 ransomware

4. Methodology

4.1 Overview

4.1 Overview

The methodology used in this study consists of six steps: (1) collecting benign and ransomware Android application samples from independent sources; (2) decompiling each application to obtain its AndroidManifest.xml file; (3) extracting the permission list declared in each AndroidManifest.xml file; (4) building a balanced, labeled dataset of both classes; (5) training four machine learning classifiers using the resulting set of features; and (6) testing using 10-fold cross-validation. The following provides detail of each stage.

4.2 Application Collection

Using the apk-downloader extension for the browser, benign samples were gathered from the Google Play Store that covered a wide variety of application categories such as games, shopping, social communication, productivity and utilities. There were 1,500 benign APK files downloaded and 995 of them were decompiled successfully, resulting in a usable AndroidManifest.xml file.

To increase diversity, ransomware samples were collected from three independent sources. A significant portion of the ransomware hashes came from the HelDroid project that was built around their research and provided labeled hash lists across a variety of ransomware families [33]. Samples were also collected from Koodous, an online collection site to which security researchers upload Android ransomware and malware samples for community analysis [34]. Another set of samples was collected through public repositories, and then checked against VirusTotal for malicious classification before being added to the collection. Overall, 3,710 candidate ransomware APKs were gathered, and 995 of these (for balanced evaluation reported in this paper) could be successfully decompiled and verified as ransomware.

4.3 Decompilation

A mix of the tools were used: APK Easy Tool v1.57 and Kali Linux apktool (some ransomware samples were not decompiled cleanly using either alone), and Jadx v1.1.0 because some obfuscated or malformed ransomware samples failed to decompile cleanly with any of the aforementioned tools. By having a variety of decompilation tools to use together, the number of samples that could not be decompiled was minimized, especially for ransomware samples that are likely to intentionally obfuscate their analysis.

4.4 Permission Extraction

The AndroidManifest.xml files for each application that could be decompiled were then used with a Python script to generate a comprehensive list of all the declared uses-permission elements. Extracted permission names were normalized by stripping standard and vendor-specific prefixes and consolidated into two class-labeled datasets - one for benign applications and one for ransomware applications.

4.5 Structuring the Datasets and Features Sets

The two permission data sets were analysed to determine the number of distinct permissions requested within the benign application set, which yielded 296 distinct permissions and the ransomware set, which yielded 185 distinct permissions. Of these, 116 permissions were shared by both classes, 180 permissions were requested only by benign applications and 69 permissions were requested only by ransomware applications, resulting in a total feature space of 365 permission-based attributes plus a binary class label (ransomware / not ransomware). This final dataset contains binary vectors of size 365 for each application instance, where a value of 1 indicates that the applications contains a declaration for the corresponding attribute in its manifest, and a value of 0 means it does not. The ground-truth class label is included. The final balancing dataset that is used to train/evaluate the classifiers contains 995 benign and 995 ransomware samples (1,990 total samples).

4.6 Machine Learning Classifiers

Four classifiers were chosen for comparison because they were shown to be effective in previous studies on Android malware and ransomware detection research, and to represent a

diverse set of linear, tree, rule- and probabilistic-based models.

Logistic Regression (LR): A linear classifier which estimates class probabilities using the logistic sigmoid function. Both logistic regression and ANNs have been reviewed by Dreiseitl and Ohno-Machado, as well as being used as baselines in a variety of applied classification problems involving binary outcomes, and logistic regression was found to be a reliable and well-understood baseline in a variety of those applied classification problems [35].

Random Tree (RT): a decision tree constructed from a randomly selected subset of features at each node. A comparison of decision-tree methods by Zhao and Zhang showed that decision trees of this type can be as competitive as other decision-tree approaches, but their performance is greatly influenced by the structure of the underlying feature space [36]. Random Tree is conceptually close to the larger class of randomized tree-ensemble methods like Breiman's Random Forests, which combine many trees trained on randomly sampled subsets of features to decrease the variance and increase the generalization [37].

Decision Table (DT): A rule-based classifier that represents a lookup table of feature-value combinations that are associated with class labels; it can be easily interpreted and is a simple alternative to tree-based approach.

Bayes Network (BN): A probabilistic graphical model that describes the conditional dependencies between the permission attributes, enabling the model to capture co-occurrence relationships between permissions, instead of considering each permission separately.

Boosting based ensemble methods, which are another way to combine a set of weak classifiers into one strong one, such as the original AdaBoost algorithm presented by Freund and Schapire [38] are another extension of the ensemble strategy that was not tested in the present study, but is mentioned as a direction for extension in Section 7.

4.7 Experimental Setup and Evaluation methodology

All the experiments were carried out in an open-source package of machine learning algorithms for data-mining tasks called WEKA [39]. This balanced data set (995 benign and 995 ransomware instances, 1,990 instances across 365 permission attributes) was tested with a widely employed 10-fold cross-validation approach, by dividing the data into 10 equal-sized folds, where each time nine folds were used for training and the remaining fold was used for testing in 10 trials [22]. It was then repeated to ensure that each example was used only once when testing, and reported metrics are the average of all ten folds. Ten-fold cross validation was chosen over train-test split since it is a more powerful estimate of the generalization error and minimizes the chances that the results are caused by a specific split of the data.

Each classifier has three metrics: classification accuracy, true positive rate (TPR – percentage of instances classified as ransomware that are actually ransomware), and false positive rate (FPR – percentage of instances classified as benign that are actually ransomware). Confusion matrices are also provided to enable checking the distribution of correct and incorrect classifications for each classifier.

5. Results

5.1 Dataset Statistics

Table 2. Overview of the benign and ransomware permission datasets.

Application Class	Downloaded	Decompiled	Distinct Permissions	Common	Class-Exclusive
Benign	1,500	995	296	116	180
Ransomware	3,710	995	185	116	69

The two datasets provide 365 unique permission attributes in total – 116 common ones, 180 benign-exclusive and 69 ransomware-exclusive – that make up the full permission-based feature space for training the classifiers.

5.2 Discriminative Permission Analysis

Table 3. Top ten permissions requested by benign and ransomware applications.

Rank	Permission (Benign)	Frequency	Permission (Ransomware)	Frequency
1	ACCESS_NETWORK_STATE	97.6%	INTERNET	90.41%
2	ACCESS_WIFI_STATE	94.2%	RECEIVE_BOOT_COMPLETED	90.41%
3	BILLING	80.2%	WRITE_EXTERNAL_STORAGE	82.73%
4	BIND_GET_INSTALL_REFERRER_SERVICE	63.3%	SYSTEM_ALERT_WINDOW	82.17%
5	INTERNET	62.1%	ACCESS_NETWORK_STATE	77.46%
6	READ_EXTERNAL_STORAGE	59.6%	MOUNT_UNMOUNT_FILESYSTEMS	69.30%
7	RECEIVE	48.5%	SEND_SMS	59.39%
8	VIBRATE	47.9%	READ_PHONE_STATE	43.65%
9	WAKE_LOCK	42.2%	WAKE_LOCK	42.69%
10	WRITE_EXTERNAL_STORAGE	41.7%	VIBRATE	40.29%

Nearly all ransomware users need this ransomware class, which is confirmed with RECEIVE_BOOT_COMPLETED with 90.41% frequency, to be able to restart itself immediately after a device restart, even if the user tries to restart the device to get rid of the lock screen. SYSTEM_ALERT_WINDOW (82.17%) is used by an application to draw a window above all others, which is the UI-hijacking behaviour that Chen et al designed RansomProber to detect [24]. The propagation behavior and creation of fake law enforcement messages as seen with some of the documented ransomware families (see Section 2.4) are both represented by the MOUNT_UNMOUNT_FILESYSTEMS (69.30%) and SEND_SMS (59.39%) privileges.

The most commonly requested permissions - ACCESS_NETWORK_STATE (97.6%), ACCESS_WIFI_STATE (94.2%) and INTERNET (62.1%) - are typical of network access, not malicious. In particular, the top-performing permissions in both classes are common to both ransomware and benign use cases, but their combination and co-occurrence are what excludes them in either case, such as the presence of both SYSTEM_ALERT_WINDOW and RECEIVE_BOOT_COMPLETED, which appear at high frequency in both classes, but are rare in benign use cases—consistent with the study by Alsoghyer and Almomani, which found that combinations of permissions rather than individual permissions carry the most predictive signal for ransomware detection [32].

5.3 Classifier Performance

Table 4. 10-fold cross-validation results across four classifiers

Classifier	Correctly Classified	Accuracy	TPR (Weighted Avg.)	FPR (Weighted Avg.)
Logistic Regression	1944 / 1990	97.69%	0.977	0.023
Bayes Network	1928 / 1990	96.88%	0.969	0.031
Random Tree	1921 / 1990	96.53%	0.965	0.035
Decision Table	1905 / 1990	95.73%	0.957	0.043

Table 5. Confusion matrix breakdown across four classifiers (Ransomware = positive class).

Classifier	TP (Ransomware)	FN (Ransomware)	FP (Benign)	TN (Benign)
Logistic Regression	974	21	25	970
Bayes Network	947	48	14	981
Random Tree	968	27	42	953
Decision Table	956	39	46	949

Logistic Regression had the highest overall accuracy of 97.69%, balanced value of 0.977, and value of 0.023 for its false positive rate. This indicates that, in this particular feature space, requested permissions and ransomware status are well separable, favoring a linear classifier like Logistic Regression over tree-based and table-based classifiers, as shown by Dreiseitl and Ohno-Machado in their work [35] on other binary classification problems with applied domains.

Bayes Network was a close second with 96.88% accuracy, with a low false positive rate on the ransomware class (14 misclassified benign, but a somewhat lower true positive rate on the ransomware class 947 of 995). This is good, in the sense that benign applications were not likely to be classified as ransomware, but it was not quite as good as it was on the ransomware class itself, in that it was more likely to mistake ransomware for benign.

Random Tree and Decision Table had only 96.53% and 95.73% accuracy respectively, both slightly below the probabilistic and the linear methods. Zhao and Zhang showed that the performance difference between different decision-tree methods could be significant depending on the structure of the feature space [36] and this is consistent with the differences in performance that we found here between Random Tree and the linear/probabilistic classifiers for this high-dimensional binary permission feature set.

5.4 Comparison with Related Studies

Table 6 compares the results of this study with the permission and classifiers based studies described in Section 3.4. Although direct comparison of accuracies must be taken with a grain of salt due to the differences in dataset size and composition, the results reported here are generally comparable to, if not higher than, those reported by Su et al. [30] for a smaller dataset of 301 ransomware samples using SVM, Decision Tree and Logistic Regression, and consistent with the high-accuracy permission-based results reported by Alsoghyer and Almomani [32] for a similarly sized and balanced dataset.

Table 6. Comparison of classifier accuracy with related permission- and ML-based studies.

Study	Best Classifier	Reported Accuracy	Dataset Size
Su et al.	Logistic Regression / SVM	Not directly comparable (dataset-specific)	301 ransomware samples
Alsoghyer & Almomani	Best-performing ML model on permissions	High accuracy (permission-based)	~500 per class
This paper	Logistic Regression	97.69%	995 benign / 995 ransomware

6. Discussion

6.1 Classifier Results Interpretation

The strong overall performance of Logistic Regression compared to the tree-based and table-based methods indicates that the decision boundary separating ransomware from benign permission profiles is near linear in this feature space at least for the set of permission attributes used in this dataset. Additionally, this has implications for the models that would be used on deployment, as the Logistic Regression model type evaluated here would be much cheaper to train, and much faster to make predictions with, than the more complex models (tree ensembles and

fully specified probabilistic graphical models) and would be able to be used as a low latency, low cost filter model in an app-store vetting pipeline without significant degradation in accuracy compared to the more complex models.

6.2 Permission Patterns interpretation.

The results of the analysis on permission levels in Section 5.2 suggest that ransomware detection accuracy is not associated with any single high frequency permission, but instead to the combination of permissions in four categories of possible functionality: persistence (RECEIVE_BOOT_COMPLETED), UI hijacking (SYSTEM_ALERT_WINDOW), storage manipulation (MOUNT_UNMOUNT_FILESYSTEMS, WRITE_EXTERNAL_STORAGE), and communication abuse (SEND_SMS). Applications asking for multiple permissions in several of these categories, at once, are a good candidate for receiving a higher priority in a detection or triage system, either by the full machine learning classifier evaluated here, or by a simple rule-based pre-screening step.

6.3 Practical Implications

The permission-based approach used in this paper can be directly applied to a pre-installation screening context in which computational cost and latency are critical constraints, such as screening pipelines in the app store or enterprise mobile-device-management (MDM) policies. This approach is significantly less resource-consuming than the dynamic and hybrid approaches discussed in Sections 3.2 and 3.3 since permissions can be extracted without running the app, disassembling its bytecode, or monitoring its runtime behavior and it can be as accurate or more than some of the more complex dynamic and hybrid approaches. This aligns with the pre-execution-screening rationale proposed by researchers in the field of crypto-ransomware prevention in particular, which state that the prevention of such attacks is preferable to the remediation efforts once the attack has been launched, and that such remediation is often irreversible as a result of the attack's encryption effects [22].

6.4 LIMITATIONS AND THREATS TO VALIDITY

The following limitations should be taken into account when interpreting these results. First, while classifiers trained solely on static features like permissions can be susceptible to attackers who try to take advantage of the fact that they may be able to obtain the same functionality without permission, or by requesting unnecessary permissions, and then simply not using them, adversarial evasion of this type is warned of in Ameer's study [19]. Second, while the dataset is significantly larger than a number of similar study samples, it is limited to ransomware families also identified by 2020, and the patterns of permissions requests could change as both Android's permission model and attackers' tactics evolve, requiring periodic retraining on new samples to ensure accurate detection of new ransomware types. Thirdly, this study is restricted to classifiers that are classical machine learning classifiers; more complex classifiers such as ensemble classifiers and deep learning classifiers were not included and are discussed in the future work section below. Lastly, due to decompilation failures, a significant portion of ransomware APKs collected (approximately 73%) were not added to the final dataset, leading to some selection bias for those samples that are more easily decompiled statically.

7. Future work and conclusion.

This paper shows that using only the permissions granted by Android applications to train a machine learning model it is possible to reach good accuracy in the classification of ransomware and benign applications, without any code execution, dynamic instrumentation or network monitoring. Logistic Regression performed best overall – with 10-fold cross validated accuracy of 97.69% across four classifiers using a balanced dataset of 1,990 samples and all the classifiers exceeded 95% accuracy. The permissions that were more discriminative were ones like SYSTEM_ALERT_WINDOW,

RECEIVE_BOOT_COMPLETED and MOUNT_UNMOUNT_FILESYSTEMS, which align with the way locker- and crypto-ransomware take control of a device and its storage.

The current assessment could be expanded in a number of ways. One way to enhance these permission-based capabilities described here, and make them more robust against the evasion risks identified by Ameer [19]–which are adversarial in nature–while retaining the lightweight nature of static analysis, would be to add a lightweight dynamic-monitoring component, in the spirit of Song, Kim and Lee's process-monitoring approach [25]. Third, hybrid methods like Ferrante et al. Two-pass static-then-dynamic [28] or Ghorbani and Ghorbani's Triage-and-escalate [26] could provide a template for escalating analysis only for applications that are flagged as suspicious by the permission-based model proposed here, which may improve both the accuracy and efficiency of the model. Third, boosting-based ensemble methods, built upon the seminal work by Freund and Schapire [38] and random forest type methods (e.g., Breiman's Random Forests [37]) need to be compared directly with the four classifiers tested here. Last but not least, the dataset should be expanded and updated to include more recent ransomware families, as the ransomware families continue to adapt to the changing permissions model of Android and the ransomware tactics employed.

REFERENCES

N. A. Hassan, Ransomware Revealed. Springer, 2019.

S. Aurangzeb, M. Aleem, M. A. Iqbal, and M. A. Islam, "Ransomware: a survey and trends," J. Inf. Assur. Secur., vol. 6, no. 2, pp. 48-58, 2017.

Ahmad, B., Jillani, S. A., Rafique, M., & Soomro, A. A. (2026, January). Design and Monitoring of a Tree-Inspired Vertical Axis Wind Turbine for Efficient Urban Wind Utilization. In 2026 1st International Conference on Innovations in Information and Communication Technologies (IICT) (pp. 1-6). IEEE.

HIPAA Journal, "Ransomware on Mobile Devices," accessed June 2020. [Online]. Available: https://www.hipaajournal.com/ransomware-mobile-devices/.

E. Arnold, "Tennessee Sheriff Pays Ransom to Cybercriminals in Bitcoin," Memphis Business Journal, 2014.

Ahmad, B., Majeed, M. K., Soomro, A. A., & Jillani, S. A. (2026, January). Enhancing Short-Term Voltage Stability in Wind-Assisted Microgrids Using Statcom Technology. In 2026 1st International Conference on Innovations in Information and Communication Technologies (IICT) (pp. 1-6). IEEE.

TechBeacon, "Ransomware on the Rise: The Evolution of a Cyberattack," 2019.

ESET, "The 6 Biggest Ransomware Attacks of the Last 5 Years," ESET White Paper, 2019.

Statista, "Mobile Operating Systems' Market Share Worldwide from January 2012 to December 2019," accessed May 2020.

V. Wilson, "G Suite Ransomware Protection," Spinbackup Blog, 2018.

Wandera, "Is Ransomware a Problem on Mobile?" accessed 2020.

Digital Guardian, "Detecting CryptoWall 3.0 Using Real Time Event Correlation," 2020.

C. V., "Mobile Malware Evolution 2018," Securelist, 2019.

Blue Coat Systems, "From Ransomware to Malvertising," Blue Coat Security Blog, 2014.

Khan, U. H., Khan, R., Alsaedi, T., Chelloug, S. A., Alturise, F., & Alkhalaf, S. (2026). Federated TinyML and digital twin framework for secure and resilient IoMT-based ICU monitoring. *Scientific Reports*.

S. Sjouwerman, "The Evolution of Mobile Ransomware," KnowBe4 Blog, 2020.

C. Corrigan, "Ransomware Information Every Android Client Should Know About," AVG Signal Blog, 2018.

KnowBe4, "Ransomware Knowledgebase: Svpeng Mobile Ransomware," accessed June 2019.

M. Ameer, Android Ransomware Detection using Machine Learning Techniques to Mitigate Adversarial Evasion Attacks. PhD thesis, Capital University, Columbus, OH, USA, 2019.

Khan, U. H., Zia, A., Ali, H., Rahim, A., Tanveer, U., & Sher, K. F. (2025). Duplicate pull requests: Automated detection using s-bert and machine learning. *Spectrum of Engineering Sciences*, 991-1010.

A. Viswanathan, K. Tan, and C. Neuman, "Deconstructing the assessment of anomaly-based intrusion detectors," in International Workshop on Recent Advances in Intrusion Detection, pp. 286-306, Springer, 2013.

Khan, U. H., Ali, H., Zia, A., Khan, H., Tanveer, U., & Bibi, S. (2025). PRIORITIZING PULL REQUESTS THROUGH DUAL PREDICTION OF ACCEPTANCE AND INTEGRATOR RESPONSE. *Spectrum of Engineering Sciences*, 1701-1715.

F. Mercaldo, V. Nardone, A. Santone, and C. A. Visaggio, "Ransomware steals your phone. Formal methods rescue it," in International Conference on Formal Techniques for Distributed Objects, Components, and Systems, pp. 212-221, Springer, 2016.

J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du, and G. J. Ahn, "Uncovering the face of android ransomware: Characterization and real time detection," IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 1286-1300, 2017.

Khan, U. H., Qamar, A., Khan, R., Alawad, M. A., Alturise, F., & Hayat, M. (2025). TrustEdge: A Quantum-Safe, Self-Healing Framework for Federated TinyML in Critical ICU Monitoring. *Internet of Things*, 101855.

A. Gharib and A. Ghorbani, "DNA-Droid: A real-time android ransomware detection framework," in International Conference on Network and System Security, pp. 184-198, Springer, 2017.

A. Alzahrani, A. Alshehri, H. Alshahrani, R. Alharthi, H. Fu, A. Liu, and Y. Zhu, "RanDroid: Structural similarity approach for detecting ransomware applications in android platform," in 2018 IEEE International Conference on Electro/Information Technology (EIT), pp. 892-897, IEEE, 2018.

A. Ferrante, M. Malek, F. Martinelli, F. Mercaldo, and J. Milosevic, "Extinguishing ransomware - a hybrid approach to android ransomware detection," in International Symposium on Foundations and Practice of Security, pp. 242-258, Springer, 2017.

D. Nieuwenhuizen, "A behavioural-based approach to ransomware detection".

D. Su, J. Liu, X. Wang, and W. Wang, "Detecting android locker-ransomware on chinese social networks," IEEE Access, vol. 7, pp. 20381-20393, 2018.

G. Sharma, A. Johri, A. Goel, and A. Gupta, "Enhancing ransomwareelite app for detection of ransomware in android applications," in 2018 Eleventh International Conference on Contemporary Computing (IC3), pp. 1-4, IEEE, 2018.

S. Alsoghyer and I. Almomani, "On the effectiveness of application permissions for android ransomware detection," in 2020 6th Conference on Data Science and Machine Learning Applications (CDMA), pp. 94-99, IEEE, 2020.

T. Yang, Y. Yang, K. Qian, D. C.-T. Lo, Y. Qian, and L. Tao, "Automated detection and analysis for android ransomware," in 2015 IEEE 17th International Conference on High Performance Computing and Communications, pp. 1338-1343, IEEE, 2015.

Koodous, "Koodous Documentation," accessed May 2020. [Online]. Available: https://docs.koodous.com/.

Khan, U. H., Qamar, A., Khan, R., Alturise, F., Alshaabani, A. R., & Alkhalaf, S. (2025). Secure edge-based IoMT framework for ICU monitoring with TinyML and post-quantum cryptography. *Scientific Reports*, *15*(1), 36195.

.

Y. Zhao and Y. Zhang, "Comparison of decision tree methods for finding active objects," Advances in Space Research, vol. 41, no. 12, pp. 1955-1959, 2008.

L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in Proceedings of the International Conference on Machine Learning (ICML), vol. 96, pp. 148-156, 1996.

Ahmad, B., Ali, S., Muneer, M., Fatima, A., & Abbas, N. (2025). Wind energy integration into the SAARC region: A comprehensive review of optimal wind sites for a sustainable super smart grid. Wind Energy, 3(2).