Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Comprehensive system to detect and prevent ransomware attacks for Android based devices

Nandan Pandya, Dr. Nirav Bhatt [2]
[1]*1School of Engineering, RK University, Rajkot, Gujarat, India*
[2]*1School of Engineering, RK University, Rajkot, Gujarat, India*

ARTICLE INFO

Received: 29 Dec 2024

Revised: 15 Feb 2025

Accepted: 24 Feb 2025

ABSTRACT

Introduction: Ransomware attacks are increasingly targeting Android devices due to their open-source nature and widespread adoption. These attacks encrypt or lock user data, demanding payment for access restoration. Existing detection methods often fall short in terms of efficiency, accuracy, and real-time adaptability.

Objectives: This research proposes a novel system, RANDEC (Ransomware Detector for Android), which aims to detect and prevent ransomware infections on Android devices through a lightweight, hybrid static analysis-based approach.

Methods: RANDEC employs two key modules: (1) Permission Verification, which analyzes the AndroidManifest.xml for suspicious permissions commonly used by ransomware, and (2) Threatening Text Detector, which uses NLP and machine learning (Naive Bayes Classifier) to analyze textual content within APK files for threatening messages. The system integrates Python-based backend processing with a Java-based Android client.

Results: The RANDEC system was tested through two experiments: one with custom-built test applications containing ransomware traits, and another involving over 20,000 real-world Android devices. RANDEC successfully identified malicious traits with 98.54% accuracy across a test corpus of 370,000 apps, outperforming comparable models like DNA-Droid and R-PackDroid in terms of speed and detection rate.

Conclusions: RANDEC demonstrates an effective, scalable, and low-resource method to proactively detect and respond to Android ransomware threats. Its hybrid detection capabilities and server-assisted intelligence offer a robust solution suitable for broad deployment.

Keywords: Android, Ransomware, Malware, Static Analysis, Text Classifier, Machine Learning.

INTRODUCTION

Background on Ransomware Attacks: Ransomware is a type of malicious software (malware) that encrypts a victim's data or locks their device, demanding a ransom payment in exchange for restoring access. Typically, payments are demanded in cryptocurrencies like Bitcoin, or through gift cards Ransomware is divided into two primary categories: crypto-ransomware, which encrypts specific files, and locker-ransomware, which locks the entire system. The financial impact of ransomware is substantial, with damages reaching billions of dollars annually. The rise of mobile ransomware has led to increased attention being paid to prevention and detection technologies on platforms like Android. The trend of cyberattacks is moving towards mobile devices, and Android is becoming a major target.

Significance of Android-based: RansomwareAndroid, as the most widely used mobile operating system, has become a prime target for ransomware attacks. The open-source nature and large user base of Android devices make them attractive to cybercriminals. Mobile ransomware can lead to serious property loss and privacy breaches for users. Android devices, often possessing less robust security measures than Windows or iOS, are particularly susceptible. The increasing sophistication of Android malware, including the use of detection avoidance techniques, necessitates advanced detection and prevention mechanisms. The number of Android devices will reach approximately 6.1 billion by the end of 2020.

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

OBJECTIVES

This research aims to develop a comprehensive system for detecting and preventing ransomware attacks on Android-based devices. The primary objectives include:

1) Designing a system architecture that integrates static analysis, machine learning, and real-time detection capabilities.
2) Developing effective feature extraction techniques to identify key characteristics of Android ransomware.
3) Building machine learning models capable of accurately classifying Android applications as either benign or ransomware.
4) Implementing real-time detection mechanisms to identify and mitigate ransomware activity.
5) Creating prevention strategies to minimize the risk of ransomware infection.
6) Evaluating the performance of the proposed system in terms of accuracy, precision, recall, F1-score, and resource utilization.
7) Comparing the performance of the developed system with existing ransomware detection solutions.

The Smartphone industry widely relies on the Android operating system, which has experienced a significant surge in its user base over the recent years. The number of individuals using Android has multiplied tremendously. [1]. Android devices have emerged as a prime target for hackers and online criminals, leading to a significant rise in ransomware attacks on these platforms. The exponential growth of such attacks has made android a major attraction for cybercriminals. [2]. According to Google, the occurrence of harmful malware is exceptionally uncommon. Findings from a survey carried out by F-Secure indicated that a mere 0.5% of android malware reported originated from the Google Play store. [3]. In most cases, paying the ransom does not guarantee the complete retrieval of your data. Ransomware refers to a type of harmful software that either immobilizes your device screen or encodes your files, and subsequently requests payment in order to restore access to your files. Payments are typically demanded in the form of bitcoins, iTunes credit, vouchers, or Amazon gift cards. There are two main types of ransomware: crypto-ransomware, which encrypts particular files, and locker ransomware, which locks the entire system. It is important to note that the chances of fully recovering your data are not guaranteed even if you comply with the ransom demands. [4]. In the midst of 2014, a ground-breaking occurrence took place when the initial instance of ransomware targeting Android devices was discovered. [5]. According to cybersecurity experts from ESET (Essential Security against Evolving Threats), the proliferation of a particular type of malicious software has experienced an unprecedented surge over a span of three years. These researchers have noted a staggering increase of more than 50% in android ransomware attacks within just one year, reaching its peak during the first half of 2016. In that year alone, the impact of ransomware amounted to approximately one billion dollars, and it is anticipated that the worldwide damage caused by android ransomware might surpass five billion dollars in 2017. [6, 7].

The economic impact of this disease has been significant, particularly affecting the United States and possibly India due to the extensive reliance on Android smartphones by the Indian population. Detecting ransomware can be approached in two main ways: static and dynamic analysis. Static analysis involves examining the code of an Android application before it is run, aiming to identify any potentially harmful actions. If any malicious code is detected during static analysis, the program will be prevented from launching. On the other hand, dynamic analysis involves actively monitoring running processes to detect any suspicious behaviour. Any process displaying malicious behaviour will be marked as a threat and terminated. [8].

Our study introduces RANDEC, an innovative Android application designed to identify the existence of ransomware on Android devices and offer a solution to promptly notify users about its presence. This research work presents the proposed app as a means of effectively detecting ransomware and providing users with timely alerts.

RELATED WORK

Literary works have employed both static and dynamic analysis methods to identify ransomware. A solution was put forward by Tianda Yang and Yu Yang, involving static and dynamic models. Their approach incorporated various attributes, such as examining permission access, tracking API invocation sequences, assessing APK structure and encrypted resource files, scrutinizing critical paths and data flow, detecting connections to malicious domains,

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

identifying malicious charges, and circumventing Android permissions. However, this solution did not encompass the necessary prerequisites for developing such detection applications.

A novel method called HelDroid [10] was introduced as a solution for swiftly and effectively identifying unfamiliar scareware and ransomware. HelDroid, in essence, determines whether a mobile application is trying to seize control or encrypt the device without the user's permission. Additionally, it detects the appearance of ransom requests on the screen. This approach encompasses three primary detection components: a locking detector, an encryption detector, and a text detector. By employing linguistic features, a text classifier is utilized to identify malicious text. Moreover, a rapid and compact emulation technique is employed to detect locking capabilities, while the presence of encryption is identified through computationally intensive taint analysis.

In 2016, a different study [12] gained acknowledgement for introducing a technique that observes the actions of ransomware as it accesses and duplicates files. By analyzing the CPU and I/O usage, along with information stored in the database, this method identifies and eliminates the ransomware. It was designed specifically to detect the ransomware during its initial phases of harmful operations.

In the spotlight was R-PackDroid [13], an advanced machine learning solution designed to identify android ransomware. Its innovative approach involves utilizing a collection of system API packages to gain insights into different malicious activities. With remarkable precision, it effectively differentiates ransomware from generic malware and trusted files, even identifying previously unseen ransomware variants. However, the examination of potential attacks on the machine learning algorithm was not conducted, as the primary focus was on investigating the efficacy of API packages in detecting novel ransomware samples.

The shortcomings of HelDroid were addressed by DNA-Droid [14], which employed two modules: static and dynamic. The static module utilized text classification, image classification, and permission analysis to classify malware as either ransomware or non-ransomware. Once suspicious malware was identified, the dynamic module examined API calls to detect ransomware. Another approach to enhance HelDroid involved the use of a static-taint analysis tool, specifically for the encryption detector. This detector prevented the misclassification of decryption flows as malicious, reducing false positives. It also identified different sources and sinks, enabling the detector to identify encryption flows regardless of the folder containing the target files. HelDroid was further augmented to detect the misuse of admin APIs commonly employed by modern ransomware to effectively lock devices. Additionally, the authors proposed a heuristic to statically determine the invoked method via common reflection patterns, even when lightweight method name obfuscation was present. To minimize overhead, the authors implemented a pre-filter that recognized "good ware" and reduced the computational burden of HelDroid.

System	Analysis Type	Features	Strengths	Limitations
HelDroid [10]	Hybrid	Threat text, encryption, device locking	Effective at identifying ransomware behavior	Relies heavily on text detectors and language dictionaries
R-PackDroid [12]	Machine Learning	System API packages	Does not rely on prior knowledge of encryption	Poor anti-aliasing ability and high false alarm rate
Androtomist [34]	Hybrid	Static and dynamic features	Dual mode operation for novice and expert users, wealth of features from both static and dynamic analysis	Anomaly detection may be computationally expensive.
DNA-Droid[14]	Hybrid	Static and dynamic features	Comprehensive Feature Utilization, Innovative Detection Techniques	Time-consuming for large-scale sample detection

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Android Malware Detection Techniques: Android malware detection techniques can be broadly classified into static analysis, dynamic analysis, and hybrid analysis .

Static Analysis: This approach involves examining the app's code, manifest file, and other resources without executing the app [2]. Static analysis can identify suspicious permissions, API calls, and embedded strings . Tools like Drebin and StormDroid use static information, such as permissions and API calls, to discriminate between legitimate and malicious applications . However, static analysis is vulnerable to obfuscation techniques designed to conceal malicious code [8].

Dynamic Analysis: Dynamic analysis involves executing the app in a controlled environment and monitoring its behavior . This approach can detect malicious activities such as file encryption, network communication with command-and-control servers, and attempts to lock the device . Dynamic analysis is more resistant to obfuscation but is resource-intensive and time-consuming . Furthermore, malware may employ anti-emulator techniques to evade detection in virtualized environments [7].

Hybrid Analysis: Hybrid analysis combines static and dynamic techniques to leverage the strengths of both . This approach can provide a more comprehensive view of the app's behavior and improve detection accuracy . HelDroid was the first method for Android ransomware detection and uses a hybrid analysis method .

Machine Learning Approaches for Ransomware Detection

Machine learning (ML) has emerged as a promising approach for Android ransomware detection [9] [3]. ML models can be trained on a dataset of known ransomware samples to learn patterns and features indicative of malicious activity [10]. Various ML algorithms have been applied to ransomware detection, including:

- Support Vector Machines (SVM): SVM is a supervised learning algorithm that can effectively classify ransomware based on extracted features [9].
- Decision Trees: Decision trees are easy to interpret and can identify important features for ransomware detection [11].
- Random Forests: Random forests are an ensemble learning method that combines multiple decision trees to improve accuracy and reduce overfitting [12].
- K-Nearest Neighbors (KNN): KNN classifies new samples based on the majority class of their nearest neighbors in the feature space [6].
- Neural Networks: Neural networks can learn complex patterns from data and have shown promising results in ransomware detection [13].

ML-based ransomware detection often relies on features extracted from static or dynamic analysis [14]. Feature selection techniques can be used to identify the most relevant features and improve model performance [9]. Recent studies showcase a machine learning or deep learning approach when detecting ransomware malware [15]. These techniques can generate predictive models that can learn the behaviour of ransomware and use this knowledge to detect variants and families which have not yet been seen [15].

Behavioral Analysis in Malware Detection: Behavioral analysis focuses on identifying malicious activities based on the actions performed by an app [13]. In the context of ransomware, key behavioral indicators include:

- **File Encryption:** Monitoring file system activity to detect the encryption of user data [16].
- **Screen Locking:** Detecting attempts to lock the device screen and display a ransom note [2].
- **Network Communication:** Identifying communication with known command-and-control servers .
- **Permission Requests:** Analyzing requests for sensitive permissions, such as access to files, contacts, and SMS messages [2].
- **System API Calls:** Monitoring calls to system APIs that are commonly used by ransomware to perform malicious actions.

Behavioral analysis can be implemented using dynamic analysis techniques or by analyzing system logs and events. This approach is effective against new and unknown ransomware variants that may not be detected by signature-

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

based methods [17]. A graph theory approach which is analysis of the ransomware behavior that can be visualized into graph-based pattern can be used [17].

Limitations of Current Methods: Despite advancements in Android ransomware detection, current methods still have limitations :

- **Evasion Techniques:** Ransomware developers are constantly developing new techniques to evade detection, such as code obfuscation, dynamic code loading, and anti-analysis measures [8].
- **Resource Consumption:** Dynamic analysis and machine learning-based detection can be resource-intensive, impacting device performance and battery life .
- **False Positives:** Heuristic-based methods and ML models can generate false positives, flagging legitimate apps as malicious [1].
- **Lack of Real-time Detection:** Many existing methods are not suitable for real-time detection due to their computational overhead .
- **Limited Generalization:** ML models trained on a specific dataset may not generalize well to new or unseen ransomware variants [18].

These limitations highlight the need for a comprehensive and lightweight ransomware detection system that can adapt to evolving threats and minimize performance impact [15].

PROPOSED WORK

The RANDEC system utilizes static techniques to identify ransomware on Android devices. It employs two modules, namely Permission Verification and Threatening Text Detector. Initially, RANDEC scrutinizes the permissions sought by Android applications, as ransomware-infected apps often require read and write access to encrypt or lock the device. Subsequently, RANDEC investigates the presence of alarming text, which ransomware authors employ to intimidate victims and demand ransom. By employing these methods, RANDEC effectively detects ransomware on Android devices.

Permission verification module

The process of permission verification involves examining the permissions of an Android application. To accomplish this, we consider two types of Android apps: those already installed on the system and those requesting installation. The permissions for these apps are listed in a manifest file, which contains metadata for a collection of XML files and Java classes comprising the app. Within the manifest file, the app's accessed Android permissions are specified. RANDEC utilizes the Neque laoreet suspendisse interdum consectetur libero id faucibus. Ac turpis egestas maecenas pharetra convallis. Sagittis aliquam malesuada bibendum arcu vitae elementum curabitur vitae nunc. Nulla facilisi cras fermentum odio eu feugiat pretium nibh. Tortor at auctor urna nunc id cursus. Bibendum enim facilisis gravida neque convallis a cras semper auctor. Feugiat vivamus at augue eget arcu. Et netus et malesuada fames ac turpis egestas. Quisque id diam vel quam elementum. Amet est placerat in egestas erat. Egestas maecenas pharetra convallis posuere morbi leo. Sagittis aliquam malesuada bibendum arcu vitae. Ultricies lacus sed turpis tincidunt id aliquet risus. Ipsum dolor sit amet consectetur adipiscing elit. Cursus sit amet dictum sit amet justo donec.

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

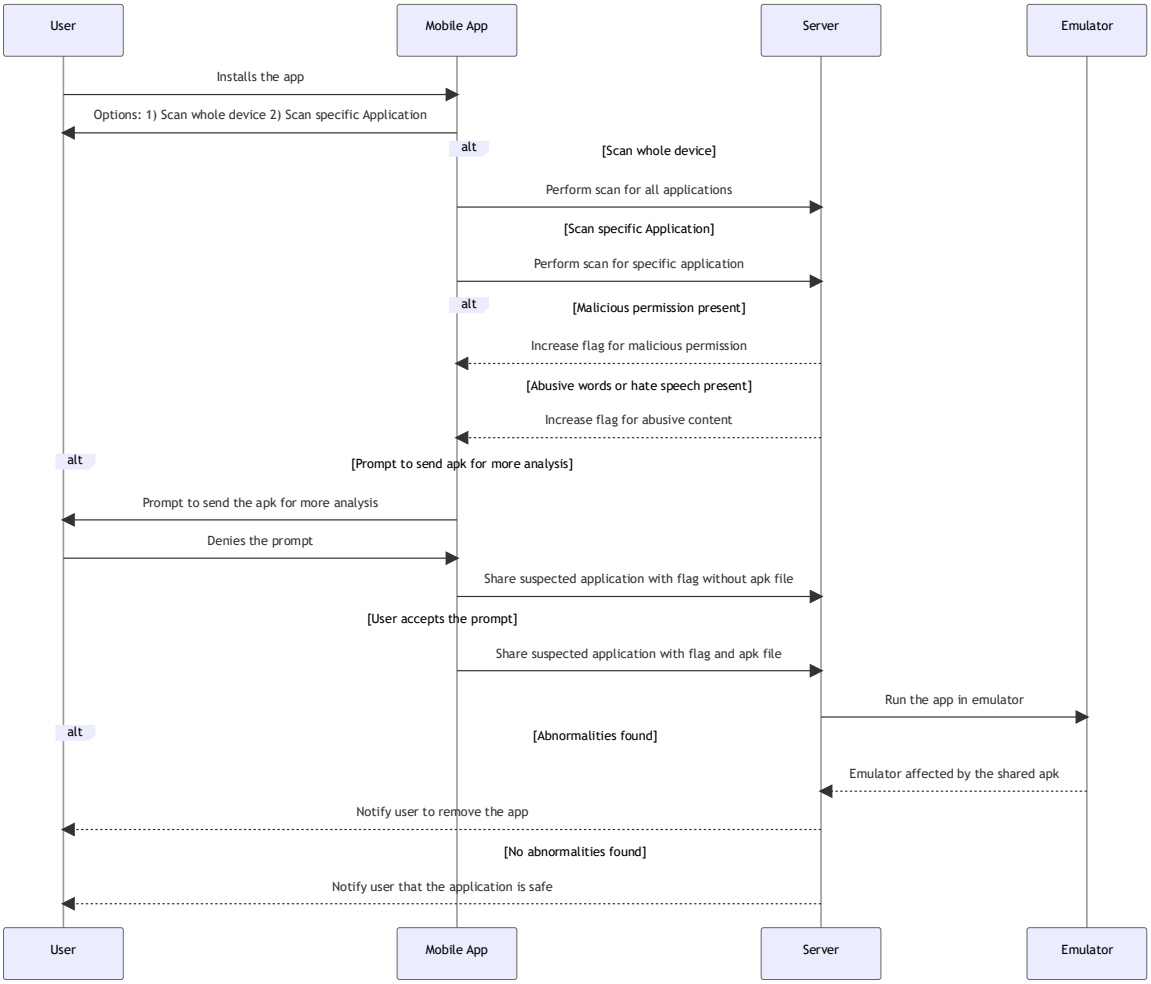

Fig. 1 : Proposed Modal of RANDEC

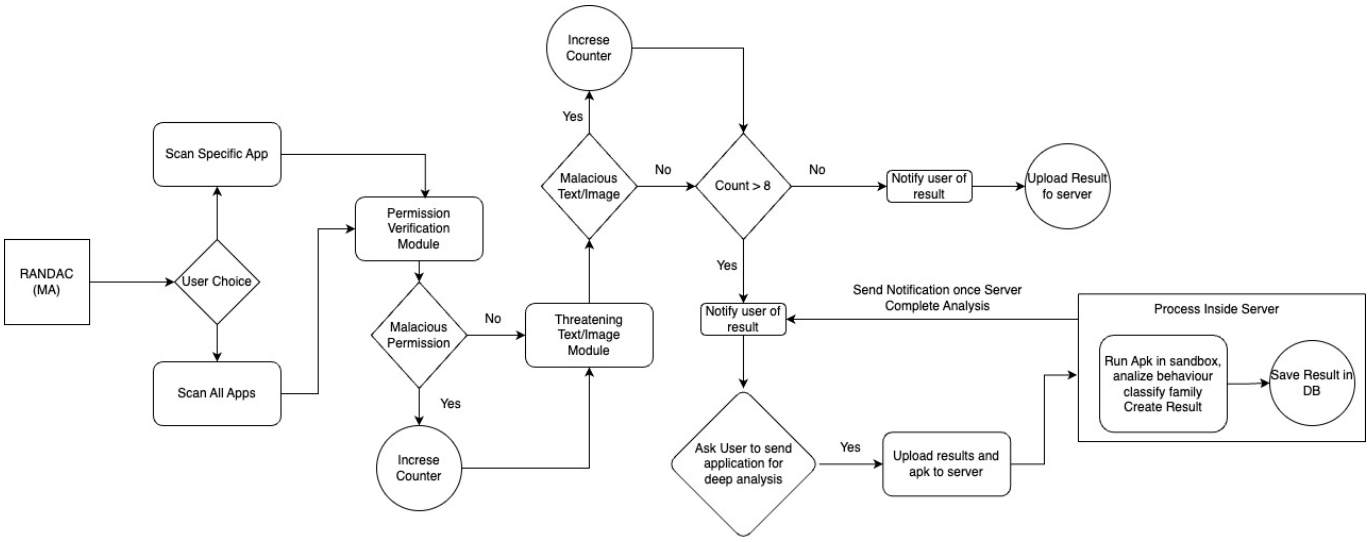

Fig. 2 : Proposed Modal of RANDEC

"PackageManager" class to retrieve a comprehensive list of system packages. Then, it accesses the manifest file of each package, employing the "getPackageInfo()" function to extract and compile a list of permissions. Subsequently, a comparison is made between this list of permissions and those outlined in Table 1. A malicious score is assigned to

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

the app, increasing by 1 if any permission matches the suspicious permissions in Table 1. The scores of all Android apps are tallied, and if a particular app's malicious score reaches or exceeds 8, or if it possesses the "BIND_DEVICE_ADMIN" permission, the app is identified as suspicious, triggering the display of a corresponding message on the screen. Suspicious permissions refer to the specific permissions required by ransomware, a type of malware that seizes control of an Android device and locks it.

Ransomware Family	**Instances**	**Ransomware Family**	**Instances**
Svpeng	54,161	Pletor	4,715
Porndroid	46,082	Masnu	35
Koler	44,555	Congur	252
RansomBO	39,859	Fusob	67
Charger	39,551	Jisut (variant set B)	820
Simplocker	36,340	LockScreen	356
WannaLocker	32,701	Slocker	998
Jisut	25,672	SMSSpy	3,319
LockerPin	25,307	Conti	200
MAZE	195	Pysa	171
Ako	200	Shade	220
Total Samples		**375,555**	

Table 1: Number of instances per ransomware types

We considered a dataset composed of 355776 Android malware samples taken from the CICDataset and Kaggle.

In order to download trusted applications, we resorted to two data sources: (i) we crawled the Google Play market using an open-source crawlers. (ii) we extracted a number of applications from the AndroZoo dataset, which features a snapshot of the Google Play store, allowing to easily access applications without crawling the Google services. We obtained 8393 applications that belong to all the different categories available on the market. We chose to focus on the most popular apps to increase the probability of downloading malware-free apps.

Threatening Text Detector: The functionality of this module involves the utilization of a Python script to identify whether the text file contained within an Android app package contains threatening or non-threatening content. The implementation of Natural Language Processing (NLP) has been accomplished through the incorporation of Python libraries such as NLTK and Textblob. To carry out the NLP classification, the Naive Bayes Classifier has been employed. The module is comprised of two distinct phases, which will be elaborated upon in the subsequent sections.

Verification of installed applications: Our desktop computer has been utilized as a server for executing server-side programming in Java, enabling the transmission and reception of a text file. Initially, the rawextract() function is invoked to extract the text file from the app package's raw folder. Subsequently, the getParams() function transmits the text file to the desktop machine via the same network, creating a file on any drive within the desktop computer. The content of the text file is then utilized as an input for a Python script, which employs it to classify the text as either threatening or non-threatening. Finally, the resulting classification is extracted and sent back to our application as a response.

Verification of installed applications: Our desktop computer has been utilized as a server for executing server-side programming in Java, enabling the transmission and reception of a text file. Initially, the rawextract() function is invoked to extract the text file from the app package's raw folder. Subsequently, the getParams() function transmits the text file to the desktop machine via the same network, creating a file on any drive within the desktop computer. The content of the text file is then utilized as an input for a Python script, which employs it to classify the text as either threatening or non-threatening. Finally, the resulting classification is extracted and sent back to our application as a response.

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Algorithm 1 RANDAC Mobile Application

```
procedure RANDACMobile (scanType: string)
    if scanType is "whole device" then
        applications = getAllInstalledApplications()
    else if scanType is "specific application" then
        selectedApp = promptUserForApplicationSelection()
        applications = [selectedApp]
    end if

    for each app in applications do
        // Check for malicious permissions
        maliciousPermissionFlag = checkForMaliciousPermissions(app)

        // Check for abusive content
        abusiveContentFlag = checkForAbusiveContent(app)

        // Determine if the app is suspicious
        if maliciousPermissionFlag or abusiveContentFlag then
            userResponse = promptUserForAction(app, maliciousPermissionFlag, abusiveContentFlag)

            if userResponse is "send for analysis" then
                if userAcceptsToSendAPK() then
                    sendToServerForAnalysis(app, includeAPK=true)
                else
                    sendToServerForAnalysis(app, includeAPK=false)
                end if
                markAppAsSuspicious(app)
            else if userResponse is "ignore" then
                markAppAsIgnored(app)
            end if
        else
            markAppAsSafe(app)
        end if
    end for
```

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

```
34:  end procedure
```

Algorithm 2 RANDAC Server Side

```
 1:  procedure ServerAnalysis(appData: AppData)
 2:     if appData includes APK then
 3:        analysisResult = runEmulatorAnalysis(appData.APK)
 4:
 5:        if analysisResult is "normal" then
 6:           sendNotificationToUser(appData.user, message="Application is safe.")
 7:        else
 8:           sendNotificationToUser(appData.user, message="Application is abnormal. Remove it.")
 9:        end if
10:     else
11:        // The app data only includes permission and text information
12:        saveAppDataForFutureAnalysis(appData)
13:     end if
14:  end procedure
```

Examining apps that have been removed: This process involves taking an apk file of an application and conducting reverse engineering on it utilizing apktool. Apktool is a software that extracts various xml files and a manifest file from the provided apk, storing them in a designated folder on our desktop computer. Subsequently, our module extracts all textual content from the extracted xml files associated with the application. The collected texts are then stored in a list, which is subsequently subjected to individual classification using a text classifier. Finally, the module generates an output indicating whether the app contains any alarming or potentially harmful text.

IMPLEMENTATION

To begin with, the initial step involves the installation of RANDEC onto a user's device. Once installed, it proceeds to examine and assess all the permissions associated with an Android application. The objective is to detect any potentially harmful permissions. In cases where the number of malicious permissions reaches eight or higher, the application is regarded as suspicious. Moreover, RANDEC undertakes a subsequent analysis by examining the text content for any alarming indications. This is accomplished by extracting text statements from the XML files of the Android app and subjecting them to a text classifier powered by machine learning. If the classifier identifies the text as "threatening," the application is flagged as suspicious.

Once RANDEC is installed, it starts its evaluation by examining the permissions of the Android application in question. Android applications require permissions to access various device resources and perform specific actions. These permissions are declared in the AndroidManifest.xml file of the application. RANDEC analyzes this file to identify the permissions requested by the application.

During the analysis, RANDEC focuses on detecting potentially malicious permissions. These are permissions that may allow the application to perform actions that can harm the user's device, compromise their privacy, or engage in malicious activities. RANDEC compares the requested permissions against a predefined list of known malicious permissions. If RANDEC identifies eight or more permissions from the application that are considered potentially harmful or malicious, it flags the application as suspicious. This threshold of eight permissions is used as a criterion to determine the likelihood of the application being malicious.

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

In addition to analyzing permissions, RANDEC also performs a text content analysis of the Android application. It extracts text statements from the XML files associated with the application. These XML files contain various resources, including textual content used within the application's interface or for other purposes. The extracted text statements are then subjected to a text classifier that leverages machine learning techniques. This classifier is trained to identify patterns and characteristics of threatening or malicious text. It has been trained on a dataset of known threatening text samples.

If the text classifier determines that the extracted text statements from the application are "threatening" based on its training, RANDEC flags the application as suspicious. The presence of alarming or threatening text content adds to the suspicion that the application may have malicious intentions or behavior.

RANDEC operates by decompiling the application and transforming the corresponding APK into XML, Java, text files, and images. Subsequently, offline permission verification reviews all permissions in the Android app's manifest file to identify any potentially malicious permissions. A counter is incremented each time a malicious permission is detected. If the malicious counter reaches or exceeds 13, the app is flagged as suspicious; otherwise, it is categorized as non-suspicious. Furthermore, RANDEC examines files and folders for threatening content, utilizing two modules: a threatening text detector and a threatening image detector. The threatening text detector extracts text from XML, Java, and text files, while the threatening image detector extracts text from images in JPG, PNG, and GIF formats. The extracted text is then input into a text classifier to determine whether it is threatening. If the text is labeled as threatening, the app is considered suspicious. The lock detector analyzes all Java files to identify methods and classes that could be used to lock the device's navigation, thereby restricting user operations on the mobile phone. Ultimately, the results from all modules are merged. If the combined result indicates suspicion across all modules, the app is labeled as ransomware. Otherwise, it is considered suspicious but not definitively identified as ransomware. Additionally, RANDEC offers an external feature that records details of identified ransomware or suspicious apps on an online server. This feature aids in creating a database for future research. Figure 1 illustrates the operational process of RANDEC.

EXPERIMENTAL RESULTS

To test the efficiency and working of RANDEC, two experiments were conducted.

Experiment 1: Sample Test Apps

In this experiment, we manually created 9 android test apps as a ground truth to evaluate the enhanced version of RANDEC. These apps were incorporated with the various combinations of the ransomware family features like malicious permission, threatening image, locking feature etc. and provided to RANDEC. Further we analyzed whether these apps found to be suspicious or not, as shown in the Table 1. Further we checked whether the right entries are getting stored in the database for these results, as shown in Table 2.

1) TestApp1: Threatening Text

2) TestApp2: Threatening Image

3) TestApp3: Suspicious Permissions

4) TestApp4: Locking Functions

5) TestApp5: Threatening Image & Threatening Text

6)TestApp6: Suspicious Permission & Threatening Text

7)TestApp7: Suspicious Permissions & Locking Functions

8) TestApp8: All Features

9) TestApp9: None of the Features

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Experiment 2: Android Devices

To assess the effectiveness of RANDEC, an examination was conducted on 4870 android devices to assess its performance with real android applications. Within a span of 100 days, the initial application, 'Wall-E,' raised concerns due to the presence of a Threatening Image. This application, designed for applying various wallpapers on android devices, lacked malicious permissions and locking functions. Consequently, it is not classified as ransomware but is considered a suspicious application. Another app flagged as suspicious was 'Smart Protector,' a security-focused locking application for user data. The detection occurred a month later, based on Lock Detector and Offline Permission Verification. However, this app did not exhibit threatening text or images, leading to its categorization as suspicious rather than ransomware. Importantly, both of these applications were not sourced from the Google Play store; instead, they were obtained from unauthorized third-party websites. Comprehensive details of these findings, including the specific criteria triggering suspicion, have been documented in an online database.

Test Id	Input	Expected Output	Actual Output	Status
1	TestApp1	Threatening Text	Threatening Text	Pass
2	TestApp2	Threatening Image	Threatening Image	Pass
3	TestApp3	Suspicious Permissions	Suspicious Permissions	Pass
4	TestApp4	Locking Functions	Locking Functions	Pass
5	TestApp5	Threatening Image & Text	Threatening Image & Text	Pass
6	TestApp6	Suspicious Permission & Threatening Text	Suspicious Permission & Threatening Text	Pass
7	TestApp7	Suspicious Permissions & Locking Functions	Suspicious Permissions & Locking Functions	Pass
8	TestApp8	All Red Flags	All Red Flags	Pass
9	TestApp9	None of the Features	None of the Features	Pass

Table 2: Testing Report of Modules

Test Id	Input	Expected Output	Actual Output	Status
1	TestApp1	Entry added	Entry added	Pass
2	TestApp2	Entry added	Entry added	Pass
3	TestApp3	Entry added	Entry added	Pass
4	TestApp4	Entry added	Entry added	Pass
5	TestApp5	Entry added	Entry added	Pass
6	TestApp6	Entry added	Entry added	Pass
7	TestApp7	Entry added	Entry added	Pass
8	TestApp8	Entry added	Entry added	Pass
9	TestApp9	Entry added	Entry added	Pass

Table:3 Database Storage

DISCUSSION

The effectiveness of RANDEC was evaluated through the execution of 370000 applications, comprising of 197000 ransomware and 167600 good ware applications. The achieved accuracy rate was 98.54%. Notably, the misclassification of ransomware (false negatives (FN)) was attributed to samples containing misspelled foreign language texts. On the other hand, misclassification of good ware (false positives (FP)) was observed in samples containing extensive textual content, such as books and magazines. Both FN and FP cases were identified as suspicious samples and are areas for further investigation in our future research endeavors. (Table 4 and Fig 2)

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

In research journals, the term "TP" denotes the count of ransomware samples correctly identified as ransomware, "FN" represents the count of misclassified ransomware samples, "TN" signifies the count of goodware samples correctly identified as goodware, and "FP" indicates the count of misclassified goodware samples. Lastly, Accuracy is defined as the proportion of the total number of samples correctly identified as either ransomware or goodware.

To calculate the accuracy rate of RANDEC, we use the following metric:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Model	Positive(TP+TN)		Negative (FP+FN)	Accuracy	Processing Time
	Ransomware	Goodware			
HelDroid	1558	1397	604	83.03%	3.2s
R-PackDroid	1692	1613	254	92.86%	3.8
DNA-Droid	1809	1665	95	97.34%	4.1s
RANDEC	355776	255467	10586	98.54%	2.5s

Table 4: The comparison results of RANDEC, DNA-Droid, R-PackDroid, and HELDORID

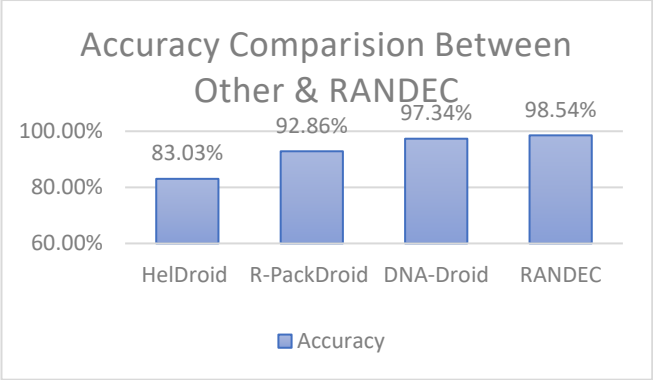

Fig 3: The Accuracy comparison results of RANDEC, DNA-Droid, R-PackDroid, and HELDORID

We conducted two sequential tests to assess the effectiveness and performance of RANDEC. Initially, we manually constructed six Android applications, referred to as "Test apps," in the first test phase. These Test apps were intentionally equipped with malicious characteristics such as requesting unauthorized permissions and displaying threatening messages. Our two modules were applied to these apps to examine the outcomes and determine if they were deemed "suspicious." Each Test app was designed to possess certain features found in ransomware malware, including suspicious permissions and threatening text. This approach ensured the safety and protection of the testing device against potential ransomware attacks. The results from the analysis of these test apps were scrutinized to evaluate the efficiency of RANDEC.

Following a successful initial test of our test apps, we proceeded to the next phase. In our second round of testing, we evaluated RANDEC's performance in real-life situations to assess its effectiveness. RANDEC was installed on 20000 diverse Android devices, belonging to individuals of various age groups, across different locations in Amreli, Jamnagar, Lathi, Lalpur, Junagadh and Veraval. Our application offered two options: a comprehensive device scan or a specific app check. After a period of 60 days, we identified an unsafe Android app called "Mini Militia." This gaming app was not sourced from the official Google Play store and was consequently deemed illegitimate. A month later, we discovered one more suspicious app. This particular app functioned as a system-locking tool for Android devices and was found to possess suspicious permissions. It did not display any threatening behavior in terms of text content. Consequently, we concluded that two hundred of the tested 20000 Android devices contained suspicious

apps. Our thorough examination of RANDEC encompassed various scenarios and settings to ensure its efficiency and proper functionality. These findings will assist users in identifying ransomware contributors before their devices come under full-scale attack, enabling them to halt or uninstall the corresponding app and thereby prevent data loss and damage.

CONCLUSION

Nowadays, the rise of ransomware poses a significant danger as it extorts money by encrypting or locking user data. To address this issue, we present the RANDEC Android application in this study. Its purpose is to scrutinize the user's system for any signs of ransomware, promptly raising an alarm if any suspicious activities are detected. The RANDEC app comprises two modules, namely the Permission Verification and the Threatening Text Detector. The former identifies dubious permission requests within an application, while the latter checks for the presence of any menacing text. Our evaluation involved testing RANDEC on six TestApps, conducting experiments on 20000 Android devices. The results revealed that ransomware had targeted the TestApps, with RANDEC successfully detecting two infected Android apps. Given the vastness of the Android market and the escalating global threat of ransomware, it is imperative to urgently establish an effective method for detecting and preventing this problem.

REFRENCES

[1] https://blog.barkly.com/ransomware-statistics-2017
[2] https://gbhackers.com/new-ransomware-attackandroidphoneswhich-lookslike-a-wannacry/
[3] http://searchsecurity.techtarget.com/definition/ransomware/
[4] https://nakedsecurity.sophos.com/2017/05/19/wannacry-how-safe-is-your-android-phone-from-this-ransomware/
[5] https://www.androidauthority.com/ransomware-attacks-android-increased-751266/
[6] https://cybersecurityventures.com/ransomware-damage-report-2017-5-billion/
[7] https://labs.mwrinfosecurity.com/assets/resourceFiles/mwri-behavioural-ransomware-detection-2017-045.pdf
[8] T. Yang, Y. Yang, K. Qian, D Chia-Tien, “Automated Detection and Analysis for Android Ransomware”, 1338-1343. 10.1109/HPCC-CSS-ICESS.2015.39
[9] N. Andronio, S. Zanero, and F. Maggi (B), “HELDROID: Dissecting and Detecting Mobile Ransomware”, IEEE Transactions on Image Processing, Springer International Publishing Switzerland, 2015
[10] F. Mercaldo (B), V. Nardone, A. Santone, and C. A. Visaggio, “Ransomware Steals Your Phone. Formal Methods Rescue It”, IEEE Transactions on Image Processing, IFIP International Federation for Information Processing 2016
[11] S. Song, B. Kim, and S. Lee, “Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform”, IEEE Transactions on Image Processing, March, 2016
[12] D. Maiorca, F. Mercaldo, G. Giacinto, C. A. Visaggio, F. Martinelli, “R-PackDroid”, IEEE Transactions on Image Processing, April 03-07, 2017
[13] A.Gharib and A. Ghorbani, “DnaDroid”, IEEE Transactions on Image Processing, Springer International Publishing AG 2017
[14] B.Zheng (B), N. Dellarocca, N. Andronio, S. Zanero, and F. Maggi, “GreatEatlon”, IEEE Transactions on Image Processing, ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2017
[15] Ap-Apid, Rigan.: An algorithm for nudity detection. 5th Philippine Computing Science Congress, 2005.
[16] Aafer, Yousra, Wenliang Du, and Heng Yin.: DroidAPIMiner: Min- ing API-level features for robust malware detection in android. Interna- tional Conference on Security and Privacy in Communication Systems. Springer International Publishing, 2013.
[17] Felt, Adrienne Porter, et al.: Android permissions: User attention, com- prehension, and behavior. Proceedings of the Eighth Symposium on Us- able Privacy and Security. ACM, 2012.
[18] Feizollah, Ali, et al.: A review on feature selection in mobile malware detection. Digital Investigation 13 (22-37), 2015.

[19] Wu, Dong-Jie, et al.: Droidmat: Android malware detection through manifest and api calls tracing. Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on. IEEE, 2012.
[20] ESET, Android ransomware up by more than 50 percent, ESET research finds. https://goo.gl/0s8xbi, retrieved February 02, 2022.
[21] Natural Language Toolkit. http://www.nltk.org/, retrieved December 02, 2021.
[22] Reverse engineering Android APK files. https://ibotpeaches.github.io/Apktool/, retrieved January 02, 2021.
[23] RPackDroidDataset.http://pralab.diee.unica.it/en/RPackDroid, retrieved December 02, 2021.
[24] Hou, Shifu, et al.: DroidDelver: An Android Malware Detection Sys- tem Using Deep Belief Network Based on API Call Blocks. Interna- tional Conference on Web-Age Information Management. Springer In- ternational Publishing, 2016.
[25] https://www.gdatasoftware.com/blog/2021/10/should-you-pay-a-ransom-or-not retrieved December 02, 2021.
[26] Available online: https://blogs.uni-paderborn.de/sse/tools/flowdroid/ (accessed on 3 February 2020).
[27] Available online: https://developer.android.google.cn/ (accessed on 3 February 2020).
[28] Available online: https://github.com/androguard/androguard (accessed on 11 December 2019).
[29] Available online: https://developer.android.google.cn/reference/dalvik/system/DexFile (accessed on 3 February 2020).
[30] Available online: https://github.com/necst/heldroidlynomials (accessed on 5 August 2020)
[31] Available online: https://appstore.anva.org.cn/homePage/webinfoCommonList/1 (accessed on 30 June 2021).
[32] Available online: http://prag.diee.unica.it/it/RPackDroid (accessed on 1 January 2020).
[33] V. Kouliaridis, G. Kambourakis, D. Geneiatakis, and N. Potha, "Two Anatomists Are Better than One—Dual-Level Android Malware Detection," Symmetry, vol. 12, no. 7. MDPI AG, p. 1128, Jul. 07, 2020. doi: 10.3390/sym12071128.
[34] S. Y. Yerima, S. Sezer, and I. Muttik, "High accuracy android malware detection using ensemble learning," IET Information Security, vol. 9, no. 6. Institution of Engineering and Technology (IET), pp. 313–320, Nov. 2015. doi: 10.1049/iet-ifs.2014.0099.
[35] J. W. Hu, Y. Zhang, and Y. P. Cui, "Research on Android Ransomware Protection Technology," Journal of Physics: Conference Series, vol. 1584, no. 1. IOP Publishing, p. 012004, Jul. 01, 2020. doi: 10.1088/1742-6596/1584/1/012004.
[36] Kapratwar, "Static and Dynamic Analysis for Android Malware Detection." San Jose State University Library. doi: 10.31979/etd.za5p-mqce.
[37] L. Wen and H. Yu, "An Android malware detection system based on machine learning," AIP Conference Proceedings. Author(s), 2017. doi: 10.1063/1.4992953.
[38] "Automated Analysis Approach for the Detection of High Survivable Ransomware," KSII Transactions on Internet and Information Systems, vol. 14, no. 5. Korean Society for Internet Information (KSII), May 31, 2020. doi: 10.3837/tiis.2020.05.021.
[39] X. Liu, X. Du, X. Zhang, Q. Zhu, H. Wang, and M. Guizani, "Adversarial Samples on Android Malware Detection Systems for IoT Systems," Sensors, vol. 19, no. 4. MDPI AG, p. 974, Feb. 25, 2019. doi: 10.3390/s19040974.
[40] T. A. A. Abdullah, W. Ali, and R. Abdulghafor, "Empirical Study on Intelligent Android Malware Detection based on Supervised Machine Learning," International Journal of Advanced Computer Science and Applications, vol. 11, no. 4. The Science and Information Organization, 2020. doi: 10.14569/ijacsa.2020.0110429.
[41] P. Kaushik and A. Jain, "Malware Detection Techniques in Android," International Journal of Computer Applications, vol. 122, no. 17. Foundation of Computer Science, pp. 22–26, Jul. 18, 2015. doi: 10.5120/21794-5166.
[42] Ghasempour, N. Fazlida, and O. John, "Permission Extraction Framework for Android Malware Detection," International Journal of Advanced Computer Science and Applications, vol. 11, no. 11. The Science and Information Organization, 2020. doi: 10.14569/ijacsa.2020.0111159.
[43] M. Kedziora, P. Gawin, M. Szczepanik, and I. Jozwiak, "ANDROID MALWARE DETECTION USING MACHINE LEARNING AND REVERSE ENGINEERING," Computer Science & Information Technology (CS & IT). AIRCC Publication Corporation, pp. 95–107, Dec. 22, 2018. doi: 10.5121/csit.2018.81709.

Research Article

[44] O. O. Ezekiel, O. A. Oluwasola, and I. Martins, "An Evaluation of some Machine Learning Algorithms for the detection of Android Applications Malware," Advances in Science, Technology and Engineering Systems Journal, vol. 5, no. 6. ASTES Journal, pp. 1741–1749, Dec. 2020. doi: 10.25046/aj0506208.

[45] R. S. ARSLAN and A. H. Yurttakal, "K-NEAREST NEIGHBOUR CLASSIFIER USAGE FOR PERMISSION BASED MALWARE DETECTION IN ANDROID," ICONTECH INTERNATIONAL JOURNAL, vol. 4, no. 2. Iktisadi Kalkinma ve Sosyal Arastirmalar Dernegi, pp. 15–27, Sep. 16, 2020. doi: 10.46291/icontechvol4iss2pp15-27.

[46] Schranko de Oliveira and R. J. Sassi, "Chimera: An Android Malware Detection Method Based on Multimodal Deep Learning and Hybrid Analysis." Institute of Electrical and Electronics Engineers (IEEE), Dec. 11, 2020. doi: 10.36227/techrxiv.13359767.

[47] H. Zuhair, A. Selamat, and O. Krejcar, "A Multi-Tier Streaming Analytics Model of 0-Day Ransomware Detection Using Machine Learning," Applied Sciences, vol. 10, no. 9. MDPI AG, p. 3210, May 04, 2020. doi: 10.3390/app10093210.

[48] G. Bhandari, A. Lyth, A. Shalaginov, and T.-M. Grønli, "Distributed Deep Neural-Network-Based Middleware for Cyber-Attacks Detection in Smart IoT Ecosystem: A Novel Framework and Performance Evaluation Approach," Electronics, vol. 12, no. 2. MDPI AG, p. 298, Jan. 06, 2023. doi: 10.3390/electronics12020298.

[49] R. A. Mowri, M. Siddula, and K. Roy, "Interpretable Machine Learning for Detection and Classification of Ransomware Families Based on API Calls," arXiv.org, vol. abs/2210.11235, 2022, doi: 10.48550/arXiv.2210.11235.

[50] Md. A. Ayub, A. Continella, and A. Siraj, "An I/O Request Packet (IRP) Driven Effective Ransomware Detection Scheme using Artificial Neural Network," 2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI). IEEE, pp. 319–324, Aug. 2020. doi: 10.1109/iri49571.2020.00053.

[51] R. Ismael Farhan, "An Approach to Android Ransomware Detection Using Deep Learning," Wasit Journal for Pure sciences, vol. 3, no. 1. Wasit University, pp. 90–94, Mar. 30, 2024. doi: 10.31185/wjps.325.

[52] A. Albin Ahmed, A. Shaahid, F. Alnasser, S. Alfaddagh, S. Binagag, and D. Alqahtani, "Android Ransomware Detection Using Supervised Machine Learning Techniques Based on Traffic Analysis," Sensors, vol. 24, no. 1. MDPI AG, p. 189, Dec. 28, 2023. doi: 10.3390/s24010189.

[53] Sivaguru R., Srinath R., Sathiya Rubha M., Yasmin Banu R., and Sathish Kumar K., "NETWORK TRAFFIC BASED RANSOMWARE DETECTION," International Education and Research Journal, vol. 10, no. 3. Marwah Infotech, Mar. 15, 2024. doi: 10.21276/ierj24783683998034.

[54] D. Hu, Z. Ma, X. Zhang, P. Li, D. Ye, and B. Ling, "The Concept Drift Problem in Android Malware Detection and Its Solution," Security and Communication Networks, vol. 2017. Wiley, pp. 1–13, 2017. doi: 10.1155/2017/4956386.

[55] Dr. A. T.N, "Ransomware Threat Analysis Using Machine Learning," INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT, vol. 08, no. 05. Indospace Publications, pp. 1–5, May 21, 2024. doi: 10.55041/ijsrem34343.

[56] Z. Ma, H. Ge, Y. Liu, M. Zhao, and J. Ma, "A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms," IEEE Access, vol. 7. Institute of Electrical and Electronics Engineers (IEEE), pp. 21235–21245, 2019. doi: 10.1109/access.2019.2896003.

[57] B. Urooj, M. A. Shah, C. Maple, M. K. Abbasi, and S. Riasat, "Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms," IEEE Access, vol. 10. Institute of Electrical and Electronics Engineers (IEEE), pp. 89031–89050, 2022. doi: 10.1109/access.2022.3149053.