Received: 14 March 2025 | Revised: 20 June 2025 | Accepted: 30 June 2025 | Published online: 11 July 2025

RESEARCH ARTICLE

FinTech and Sustainable Innovation
2025, Vol. 00(00) 1–13
DOI: 10.47852/bonviewFSI52025685

BON VIEW PUBLISHING

# Next-Gen Threat Hunting: A Comparative Study of ML Models in Android Ransomware Detection

**Linet Momposhi[1], Alana Maurushat[1], Rodrigo N. Calheiros[1] and Shawal Khan[1,*]**

[1] *Western Centre for Cybersecurity Aid and Community Engagement (WCACE), Western Sydney University, Australia*

**Abstract:** Ransomware attacks on Android devices have been increasing in recent years, posing a significant threat to users' data and privacy. In the finance sector, ransomware increasingly targets banking applications that impact their financial operations. This research presents a comprehensive evaluation of four popular machine learning algorithms – K-nearest neighbors (KNN), neural networks (NN), random forest (RF), and support vector machines (SVM) – in classifying Android ransomware. In this work, we utilize an open-source ransomware dataset available on Kaggle that comprises 10 types of ransomware and benign instances of Android applications, extracting relevant features for analysis. The performance of each classifier is assessed using various evaluation metrics, including accuracy, precision, recall, and $F1$-score. The experimental work shows that the RF classifier achieves the highest accuracy of 96.22%, followed by SVM with an accuracy of 83.51%, NN at 81.91%, and finally KNN at 70.49%. Furthermore, the research explores the strengths and limitations of each algorithm, providing insights into their suitability for real-world ransomware detection scenarios. The findings contribute to the development of robust and efficient security mechanisms for safeguarding Android devices against the evolving threat of ransomware.

**Keywords:** ransomware detection, machine learning, random forest, K-nearest neighbors, neural networks, android ransomware, cybersecurity

## 1. Introduction

The rapid proliferation of mobile devices, particularly those powered by the Android operating system, has profoundly reshaped the digital landscape. As of July 2023, Android commands a staggering 70.8% market share among mobile operating systems [1]. Developed by Google, Android is an open-source, Linux-based mobile platform that supports a wide array of technologies, including Wi-Fi, short message service (SMS), Bluetooth, accelerometers, cameras, global positioning systems, voice over LTE (VoLTE), and more. The latest iteration, Android 13, was released in 2022. Ransomware has emerged as one of the most insidious and disruptive threats targeting mobile devices, particularly those running the Android operating system and applications. Unlike traditional malware, which aims to steal data or compromise systems, ransomware takes a more direct and aggressive approach. Its primary objective is to hold users' data hostage by encrypting files, documents, photos, and other sensitive and valuable information stored on their devices.

A ransomware attack typically begins when an unsuspecting user unknowingly downloads and installs a malicious application masquerading as legitimate software. These malicious apps can be found on third-party app stores and websites or even disguised as updates to popular applications. Once installed, the ransomware quietly runs in the background, mapping the device's storage and identifying valuable data to encrypt.

After encrypting the targeted files, the ransomware displays a ransom note, demanding payment, often in the form of cryptocurrencies like Bitcoin, to obtain the decryption key [2]. The encrypted data remains inaccessible without the decryption key, rendering the device and data useless for most practical purposes. The ransom note typically includes a countdown timer, adding urgency and pressure on the victim to pay up before the deadline expires.

Android devices and applications are particularly vulnerable to ransomware attacks due to the open nature of the Android ecosystem [1]. While the official Google Play Store has security measures in place, users often sideload applications from untrusted sources, inadvertently opening the door to malicious software. Additionally, older Android versions with unpatched vulnerabilities can provide entry points for ransomware to exploit. Their target also includes banking applications that pose a significant threat to the financial technology (FinTech) sector. For instance, in 2021, the Reserve Bank of New Zealand became a victim of a ransomware attack that exploited the vulnerability in Accellion File Transfer (FTA), a third-party file-sharing service. The attack was linked to the Clop ransomware group that led to the disclosure of sensitive financial data including Australian banking partners. Adversaries particularly exploit the vulnerabilities within mobile banking applications such as outdated authentication modules, insecure Application program interfaces (APIs), and social engineering. Since the banking sector relies on Android mobile applications for service delivery, ensuring security is critical to prevent such high-cost cyber threats.

***Corresponding author:** Shawal Khan, Western Centre for Cybersecurity Aid and Community Engagement (WCACE), Western Sydney University, Australia. Email: 22085153@student.westernsydney.edu.au

The consequences of a successful ransomware attack can be harmful to both individuals and businesses. Personal data, including irreplaceable photos and videos, financial records, and confidential information, may be permanently lost if the ransom is not paid or the attackers fail to provide a working decryption key. Furthermore, the financial losses can be substantial, as ransomware groups often demand exorbitant ransom payments, sometimes reaching thousands or even millions of dollars [2]. According to [3], a Middle Eastern fintech company bears a direct financial loss of exceeding $2.5 million that comes with long-term reputational damage. Other incidents reported in [4] record a loss of up to $3 million per incident due to vulnerabilities exploited by cybercriminals.

To address the above mentioned problem and combat Android ransomware requires a multifaceted approach, including user education, robust security measures, and advanced detection techniques. Users should be cautious when downloading applications from untrusted sources and keep their devices updated with the latest security patches. Additionally, regularly making backups of important data can reduce the impact of data loss.

Researchers and security experts are also exploring various machine learning (ML) and artificial intelligence (AI) techniques to detect and prevent ransomware attacks proactively [5]. By analyzing patterns in network traffic, system behavior, and file activity, these advanced methods can potentially identify ransomware before it can encrypt data. Ultimately, the fight against Android ransomware is an ongoing battle, as cybercriminals continuously evolve their tactics and develop new ransomware strains. Staying vigilant, implementing robust security measures, and leveraging edge-cutting detection techniques are crucial steps in protecting individuals and organizations from the devastating consequences of these attacks.

Research shows that signature-based detection is effective at stopping known, commonly used ransomware, but it cannot identify new types of ransomware. The common use of unique versions of malware for each attack campaign by ransomware groups also makes signature-based detection ineffective. Traditional signature-based detection uses hash signature samples, and because zero-day attacks are not recorded in antivirus software databases, detecting ransomware by using an anomaly-based detection method is more effective. Therefore, there is an urgent need to explore and analyze the potential of ML techniques in detecting and mitigating Android ransomware threats. This study aims to analyze the effectiveness of ML models in detecting Android ransomware. The research questions are discussed in the next subsection.

This research work aims to evaluate the efficacy of ML techniques in classifying Android ransomware. Specifically, it focuses on KNN, support vector machine (SVM), neural network (NN), and random forest (RF) classification algorithms. These algorithms leverage features extracted from network traffic data to distinguish between benign and malicious activities associated with ransomware. The principal contributions of this study are as follows:

1) First, we use a real-world dataset with 10 known Android ransomware samples, which ensures that the findings of the study accurately reflect the actual behavior of Android ransomware, hence making it more applicable to real-world scenarios.
2) Second, the study completes a comprehensive analysis of the performance of the different ML classifiers – KNN, SVM, NN, and RF – in detecting Android ransomware, providing insights into their strengths and limitations.

The detailed experimental evaluation shows the effectiveness of the ML algorithms. Also, the limitations and possible improvements are discussed in detail.

While the ML techniques have shown promising results in android malware detection and classification, they remain limited by several shortcomings that include relying on static features, which are vulnerable to evasion through code obfuscation and behavioral variation in ransomware types.

The rest of the paper is organized as follows: In Section 1, we present the introduction and contribution. Section 2 includes the background, motivation, and taxonomy of ransomware. In Section 3, we explain the recent advancements and report on the recent work in ransomware detection and classification. Section 4 explains the methodology, and in Section 5, we report the evaluation metrics and results. Finally, Section 6 concludes this research and future research direction.

## 2. Related Work

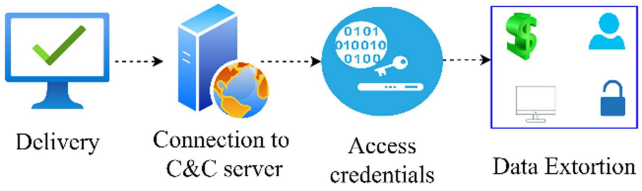
Ransomware is a type of malware that prevents its victims from accessing their systems by locking the device's lock screen or by encrypting the user's files. This type of malware normally prevents access to the system until the requested ransom is paid. In other cases, even after the victims have paid the ransom as requested by the attacker, they may not receive their stolen data or access to their system. According to a report by TrendMicro, almost one in five companies that are affected by ransomware do not get their data back or get a decryption key. Another report by Kaspersky in 2021 found that 17% of the people who paid the ransom did not receive their stolen data. Ransomware attacks have been successful mainly due to their pseudonymous ransom payment method. The attackers usually conceal their identity by asking for ransoms in the form of cryptocurrency like Bitcoin to restore access to data and files.

Ransomware blocks access to the device or encrypts the files/-data in the device. The attacker then asks for ransom money to decrypt the files/data or unlock the device. The attacker, not the computer owner, holds the decryption key, rendering the files inaccessible unless the threat actor decrypts them. In a typical ransomware attack, the attacker demands a ransom, often in the form of cryptocurrency, to decrypt the files. This ransom can range from hundreds to millions of dollars, depending on the target. Some ransomware attacks also involve a time limit, after which the files are deleted, adding pressure on the victims to pay quickly. In other instances, the attacker steals copies of the data and threatens to release them if the ransom is not paid. These attacks usually target companies and government organizations that maintain secret information. Despite the devastating effects of ransomware, there is no guarantee that paying the ransom will result in the files being decrypted. Shulmistra found that even with the decryption key, some companies struggle to recover their data due to the strong encryption algorithm employed by the attackers [2].

Ransomware attacks start as a breach in a computer file or network, which is caused by a successful penetration through any of the attack vectors. For instance, a user might click on a malicious link they received via email or text message, which in turn downloads ransomware onto their device, giving the attackers access to either their network or device [6]. The phishing technique plays a vital role in sending and luring users to click and download the files sent through these emails. Once the attackers have access to the user's device or network, it takes them a short time to deploy their ransomware, which then automatically encrypts all the user's files or locks the user out of their device.

In 2021, the critical infrastructure observed an increase in ransomware attacks globally that include countries such as the USA, Australia, and the UK. The Federal Bureau of Investigation, the Cybersecurity and Infrastructure Security Agency, and the National

**Figure 1**
**Ransomware infection stages**




Security Agency observed incidents involving ransomware against 14 of the 16 US critical infrastructure sectors. The Australian Cyber Security Centre observed continued ransomware targeting of Australian essential entities of infrastructure, including those in the healthcare and medical, financial services and markets, higher education and research, and energy sectors. The UK's National Cyber Security Centre (NCSC-UK) recognizes ransomware as the biggest cyber threat facing the UK.

Ransomware infections can be generalized in their attack phases as in Figure 1, which include the following phases:

1) Delivery: This is the initial stage in which the attackers/cybercriminals find the best way to deliver the ransomware to their desired victim's device (mobile devices, PCs, IoT, MIoT, etc.). This can be done through phishing emails, malicious applications, drive-by downloads, exploiting system or target device vulnerabilities, SMS or multimedia messaging service (MMS), and social engineering [7].
2) Command and Control: Once the ransomware is transferred from the attacker's machine to the target computer, to persist the connection, the ransomware makes a connection to the attacker's machine through a Command & Control (C&C) server that can be used for further exploitation such as lateral movement in the network.
3) Extortion: Once the ransomware has performed its malicious action, it progresses to the final stage of infection, initiated by displaying a ransom note. This ransom note often includes the ransom payment method, the ransom amount to be paid, and an account for the ransom to be paid to. The payment method attackers prefer is Bitcoin due to the pseudonymous nature of cryptocurrency.

## 2.1. Taxonomy of ransomware

Ransomware comes in various forms, each with its own characteristics and methods of operation. Ransomware however can be classified in four main ways, including the targets, infections, communication, and the type of malicious action that the ransomware performs. These four categories can further be broken down into specifics.

### *2.1.1. Targets*

Targets of ransomware can be of two types. A target of a ransomware attack could either be a victim or a target platform as analyzed by Oz et al. [2].

### *2.1.2. Victims*

Victims of ransomware attacks can be categorized into two groups. They include individuals or organizations. Individual end-users were the initial targets of ransomware attacks. In 1989, when the first ransomware was discovered, researchers reported that it was sent via floppy disks to individuals who attended the AIDS conference [8]. The presence of many end-users makes ransomware a lucrative business for attackers as they can reach many people on the internet, some of whom are not conversant with the ransomware business model.

Organizations have also been targets of ransomware attacks, given the profits the attackers could receive if they successfully receive a ransom payment [7]. Organizations include hospitals, schools, governments, businesses, and nongovernmental organizations.

### *2.1.3. Platforms*

While personal computers (PCs) have historically been primary targets for ransomware attacks due to their widespread use in both personal and business contexts, the landscape of ransomware targeting has evolved to encompass a broader range of devices and systems. PCs continue to be significant targets for ransomware operators, given the vast amount of sensitive data stored on these devices and their interconnectivity within networks. However, other platforms such as servers, mobile devices, IoT devices, and even cloud infrastructure have increasingly become targets for ransomware attacks. While PCs remain prominent targets, it's essential to recognize that ransomware threats are not limited to one type of device or system.

The Internet of Things (IoT) is the interconnected network of sensors, networks, actuators, and software that store and exchange data. The IoT makes indirect communication between individuals and smart devices possible, leaving it a target for ransomware attacks. By 2013, IoT had evolved, and it was being used in smart homes and buildings [7]. Although ransomware attacks in IoT environments are not very common now, they can be targeted by ransomware attacks. The timely, critical, and irreversible nature of ransomware attacks can have amplified effects when IoT is involved, especially in the case of critical infrastructure. IoT devices widen the attack surface for ransomware deployment, potentially leading to cascading consequences. Ransomware operators have targeted critical infrastructures or high-profile targets that likely rely on Operational Technology (OT) and Industrial Control Systems (ICS). The OT includes hardware and software and is used in ICS settings for monitoring and control purposes [9]. Attacks involving OT systems can be dangerous and have cascading effects down the supply chain, pressuring victim organizations to comply with ransom demands. Furthermore, ransomware for IoT devices would likely be conceptually different from ransomware targeting traditional operating systems. Many IoT devices use embedded operating systems, and the types of information a threat actor would look to encrypt or use for extortion purposes will likely be more limited.

With the proliferation of smartphones and tablets, mobile devices have become an integral part of everyday life, handling sensitive personal and professional information. As such, they have increasingly become targets for ransomware attacks. Like traditional computing systems, mobile devices are vulnerable to ransomware attacks due to their connectivity, storage capabilities, and reliance on software applications. The interconnected nature of mobile devices through networks and sensors makes them susceptible to exploitation by threat actors seeking to deploy ransomware [10].

## 2.2. Infection vectors

Razaulla et al. [11] categorize ransomware by their infection vectors. There are five main ways in which ransomware can be delivered to the victim. They include malicious or phishing emails,

SMS or MMS, malicious applications, drive-by downloads, and software vulnerabilities.

A malicious application is an application (app) that disguises itself as a legitimate application. These apps represent a common attack vector in the spread of ransomware in general. Users unknowingly download and install these apps from unofficial sources, providing attackers with an entry point for ransomware deployment. Ransomware may exploit vulnerabilities in SMS and messaging platforms to deliver malicious payloads. This involves the use of specially crafted messages or links that, when opened, initiate the ransomware installation process on the device.

The availability of vulnerabilities within the different existing operating systems, networks, and software poses a risk of cyber criminals exploiting these vulnerabilities to gain access to a device and deploy ransomware. Cybercriminals have leveraged both known and zero-day vulnerabilities to gain unauthorized access to systems/devices, enabling the installation and execution of malicious code including ransomware [11]. A zero-day vulnerability is a vulnerability in a device or system that security experts have had zero days to patch. Drive-by downloads and malicious scripts on compromised websites explain vulnerabilities in the device's browser or operating system, facilitating unauthorized access and ransomware installation on the device. This type of infection also leverages email attachments or malicious links sent to victims via email or SMS. Once the user clicks on the hostile link or file, the ransomware gets access to the system.

## 2.3. Motivation behind cyberattacks

There are different motivations behind cyberattacks as discussed in this section.

### *2.3.1. Financial gains*

Cyberattacks are driven by various motivations, and understanding them is vital in understanding the threat landscape. The primary motivation of a malicious actor is financial gain, and they use different techniques to steal and extort money from unsuspecting victims. Most cybercriminals gain access to victims' devices through malware and gain unauthorized access to their digital bank accounts, steal security credentials to financial platforms, and transfer funds or use phishing techniques to swindle money. Although different motivations drive threat actors, their main goal is usually to make a profit.

### *2.3.2. Sabotage and disruption*

Some cybercriminals are aiming to disrupt critical operations, services, and infrastructure. These criminals often disrupt the normal functioning of devices or networks for ideological or political reasons, which can lead to significant inconvenience, loss of reputation, and significant financial loss. Some use their hacking skills to sabotage large organizations to promote a campaign or warn an organization of its system's vulnerabilities [12].

### *2.3.3. Cyber espionage*

Organizations, states, and corporate competitors can engage in this form of cyberattack to gain a competitive edge over others. Conducted for commercial, strategic, or political gains, espionage involves collecting sensitive data, intellectual property, trade secrets, or government secrets. States can engage cybercriminals in cyber espionage to gather intelligence, promote their national interests, or engage in geopolitical maneuvers [12].

### *2.3.4. Personal vendetta*

Individuals who have access to sensitive data or critical systems can use them to carry out cyberattacks out of personal grievances, to the detriment of an individual, organization, or entity. This form of attack often comes from inside employees, partners, vendors, or contractors and is one of the biggest cybersecurity threats to organizations.

### *2.3.5. Recognition and notoriety*

Some malicious actors engage in cybercrime because of the promise of fame and recognition. These cybercriminals are motivated by the sense of achievement that comes with hacking into major systems, networks, or devices. They get recognized among fellow threat actors, and they act as individuals or in groups. Threat actors are generally competitive and are motivated by the challenges that their actions create.

## 3. Literature Review

Mobile ransomware detection is critical in safeguarding the security of mobile devices. The increased use of mobile devices has led to a surge in malware attacks, especially on Android phones. Android remains one of the most used mobile OS, with a market share of 72.2% as of May 2021, followed by iOS with a market share of 26.77%. Android has become the most-targeted mobile platform for malware, including ransomware.

Although mobile devices have many built-in security measures, their flaws and design weaknesses make them prone to malware attacks. Awareness of mobile vulnerabilities and modes of attack is vital for effective malware detection and security flaws analysis. The most prominent detection methods are ML-based, signature-based, and behavioral-based, which are explained below.

Senanayake et al. [13] believe that ML-based detection methods have proven effective in detecting mobile ransomware attacks because they can derive a classifier from limited training examples. Several studies have focused on ML-based detection and classification methods on Android devices. Several other notable mobile malware detection methods are an alternative to ML. As discussed by Sihag et al. [14], signature-based detection methods are used to analyze the behaviors of applications to generate the signature, which is then linked to the signature database to detect malicious activities. The researchers propose a signature-based malware detection solution that uses dynamic analysis to evaluate application behavior. The system filtered system logs generated from 260 applications on Google Play and matched them with the signatures. The system identified application behaviors that indicated data leakage, jailbreak attempts, and access to critical permissions. The signature-based approach was effective in providing valuable insights into mobile applications that cause mobile security and data breaches [15]. Although behavior-based malware detection is a different method of detecting mobile malware, Vanjire and Lakshmi [16] propose a behavior-based malware detection model that uses ML. They used the decision tree (DT), Naive Bayes (NB), and KNN algorithms to identify malicious behavior in mobile devices, with NB giving the highest accuracy of 97.37%. Behavior-based detection aims to identify previously unknown threats to mobile security, but it relies heavily on ML to flag unusual network activity as potentially malicious or benign. Aslan and Samet [17] discuss that data mining techniques such as n-gram are used to derive features from application behaviors. However, creating an effective detection system is challenging due to the many extracted features and difficulties in defining behavior. The researchers also discuss heuristic-based

malware detection, which also uses ML techniques, among other methods. Essentially, it entails using predetermined algorithms and rules to detect malicious behavior. However, it cannot detect complex malware and is prone to generating false positives [17]. As this research focuses on ML for ransomware detection, only studies using ML techniques are explored further in the following section.

### 3.1. Static analysis

Static analysis with ML is the most widely used technique for detecting Android ransomware. Jannat et al. [18] conducted a static analysis by using data from both benign and malicious Android applications. The MalGenom dataset containing 360 malicious Android applications and Kaggle datasets with over 4000 applications were used in this study. Some of the ML algorithms tested include RF, SVM, and NB. The RF classifier gave the best score, followed closely by DT. This study contended that static analysis is not as accurate as dynamic analysis. Elayan and Mustafa [19] also conducted a study to investigate the effectiveness of traditional ML classifiers using static analysis. The study utilized the CICAndMal2017 dataset, which consists of 347 benign samples and 365 malware samples of Android applications. This dataset provided real and realistic samples for static analysis, focusing on permissions and API calls as features indicative of malware behavior. SVM, KNN, DT, RF, and NB classifiers were evaluated for their performance in detecting Android malware. The RF classifier achieved the highest accuracy of 97.8% among the tested classifiers. The ML classifiers based on static features showed promising results, with RF performing well.

### 3.2. Dynamic analysis

Bhatia and Kaushal [20] propose a dynamic analysis approach for detecting malware in Android applications, utilizing system call traces collected during runtime interactions. Using datasets comprising 50 benign and 50 malicious Android applications, sourced from the Google Play Store and the Android Malware Genome Project, respectively, the study evaluates the efficacy of two ML algorithms: J48 DT and RF. Results indicate that both algorithms achieve high accuracy levels, with the J48 DT algorithm reaching 85% accuracy and the RF algorithm achieving 88% accuracy. Evaluation metrics including true positive and true negative rates, confusion matrices, and various other metrics provide insights into the algorithms' performance in classifying applications. The dataset utilization and thorough analysis of results underscore the effectiveness of the proposed dynamic analysis approach for Android malware detection, demonstrating its potential for real-world application in enhancing mobile security.

### 3.3. Hybrid analysis

Essentially, hybrid analysis provides a more robust malware detection capability by integrating the strengths of both static and dynamic approaches. Ding et al. [21] introduce a hybrid approach for malware classification and detection to improve both static and dynamic analysis. In the detection layer, static features such as permissions and intent are utilized, and through feature selection and algorithm comparison, the optimal static detection algorithm (RF) and feature selection method (chi-square test) are identified, achieving a final detection rate of 95.04%. Despite a marginal loss in detection rate, the experiment revealed the presence of numerous irrelevant and redundant features in the original dataset. Subsequently, in the dynamic analysis layer, network traffic images generated during dynamic execution are classified using Res7LSTM, demonstrating superior abilities in malware detection and subclassification, notably in the Android malware category and family classification. The integration of static detection with dynamic network traffic analysis effectively addresses the challenge of accurately identifying low-trust benign samples while enhancing malware detection capabilities. Another work proposed Android malware detection by integrating static and dynamic techniques. Leveraging datasets from the malware genome project, the Drebin project, and the CICMalDroid dataset, the study extracts comprehensive features for analysis. Static analysis involves attributes like manifest permissions, API call signatures, and intent filters, while dynamic analysis delves into behavior analysis using the CopperDroid framework. The research evaluates various ML and deep learning algorithms, identifying gradient boosting as the most accurate and efficient model for malware detection, achieving approximately 99% detection accuracy.

### 3.4. Machine learning for ransomware detection and classification

ML plays a critical role in detecting and mitigating Android ransomware. ML enables computers to learn from the data and make a decision or prediction based on that data. In contrast to following fixed rules, the ML algorithms analyze the patterns in data to improve their performance over time. According to Feng et al. [22], ML techniques have become mainstream in the detection of malicious applications in the Android ecosystem. The detection and analysis techniques are further categorized into three: static, dynamic, and hybrid analysis. Static analysis entails the analysis of a specific application without executing it, and in Android, it implies the analysis of the application package (APK) file. Contrary to static analysis, dynamic analysis involves the execution of an Android application in a sandboxed environment to monitor its behavior and detect malicious activities. Although this technique needs more computational power, it can help detect unknown malware [18]. As such, the hybrid type of analysis supports both static and dynamic analysis to fill the gap and address their shortcomings.

Mobile security solutions use ML and algorithms to analyze operating system behavior and detect ransomware. Detecting malware using ML entails two critical stages: analyzing Android APKs to identify the appropriate set of features and then using machine and deep learning to identify malicious APKs. The ML techniques mainly detect and identify whether the application falls into ransomware or a benign file [23]. It utilizes feature engineering to track ransomware patterns. The features used to train models to detect ransomware include file system activities, system call sequences, network traffic patterns, and API calls. These researchers demonstrate that ML relies heavily on data to identify patterns and make accurate predictions.

Different studies have tested the accuracy of ML in detecting ransomware. Narudin et al. [24] conducted a test using ML classifiers to detect mobile malware using samples from the MalGenome project, comprising 1260 Android malware samples from 49 different families. They selected the top 20 free applications on Google Play for the normal dataset. The study established that the RF for the Genome malware dataset demonstrated a 99.99% malware detection rate. The study also found that ML classifiers can detect even the latest malware. Liu et al. [25] provide a comprehensive review of ML-based approaches for detecting mobile malware and explain more aspects of ML methods. These include sample acquisition, data reprocessing, feature selection, feature type, ML method, and dataset division, among other aspects. The researchers emphasize

the importance of the sample acquisition aspect of ML because accurate predictions require good data samples. They also recognize MalGenome as one of the most widely used mobile malware datasets in ML. However, it quickly becomes outdated as malware evolves and is being substituted by more robust sample libraries such as AndroZoo. The work presented in [26] and [27] provides a comprehensive review of ML approaches for ransomware detection.

Usha et al. [28] specify that there has been a significant increase in the use of ML approaches to identify and mitigate ransomware attacks. This ransomware detection technique uses algorithms to enable computer systems to learn patterns and extract meaningful data from datasets. The study improves on previous studies that used various ML algorithms such as KNN, RF, and Bayesian Network, among others, by using the behavior of the ransomware files from the ISOT Ransomware Detection dataset to train and test the model. The study analyzes four ML algorithms, that is, K-nearest neighbor (KNN), RF, DT, and Gaussian NB. The results of the study indicate that RF is more accurate than the other algorithms and effectively differentiates between benign and infected files using the provided training and validation data.

This study makes several significant contributions to the field of Android ransomware detection research. While previous studies such as Noorbehbahani et al. [24] utilized the CICAndMal2017 dataset, our research employs the more recent Android Ransomware Detection dataset, providing insights into classifier performance on contemporary ransomware samples. Unlike Albin et al. [1], who explored DT, SVM, KNN, feedforward neural network, tubular attention network, and an ensemble model, our study specifically focuses on a systematic comparison of RF, KNN, SVM, and NN classifiers. Furthermore, our research employs a streamlined methodology by conducting a focused experiment using feature selection techniques to optimize the dataset before training the models, rather than comparing performance with and without feature selection as done in previous work. This targeted approach allows for a more direct comparison between the four selected classifiers under optimized conditions, enabling a clearer assessment of their relative strengths and weaknesses in detecting modern Android ransomware. By providing this comprehensive comparative analysis of classifier performance on a newer dataset with optimized features, our study enhances the understanding of ML-based ransomware detection capabilities and offers practical insights for selecting the most effective classification techniques for real-world mobile security applications.

## 4. Research Methodology

This section explores the essential methodology employed in this research work. We will define the research problem, set out the research questions, describe the data collection, and detail the data preprocessing steps before feeding the data into an ML classifier for further processing. The whole data analysis process is depicted in Figure 2. We also explain the selected ML algorithm and explain the metrics used for evaluation.

### 4.1. Data preprocessing

This study utilized the Android Ransomware Detection dataset created by Subhadeep Chakraborty and made available by a user named Cyber Cop on Kaggle[1]. The dataset comprises over 203,000 records encompassing Android ransomware and benign network traffic. These records were collected by monitoring the network activity of Android devices. The dataset comprises 85 attributes, each of which is described in detail in Appendix 1. The ransomware instances in the dataset are distinguished by the specific ransomware strain indicated in the Label attribute. There are ten identified ransomware variants present in the dataset.

These variants include SVpeng, PornDroid, Koler, RansomOB, Charger, Simplocker, WannaLocker, Jisut, Lockerpin, and Pletor. The distribution of these ransomware variants is presented in Table 1.

**Table 1**
**The distribution of ransomware variants in the dataset**

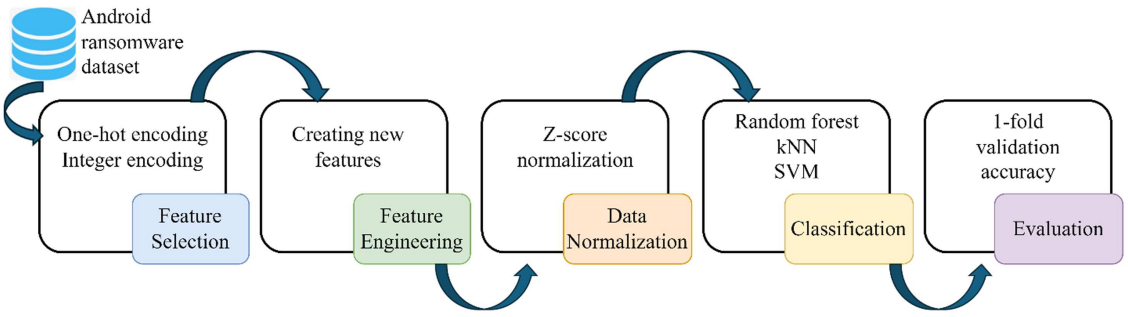
| Ransomware variant | Number of records |
|---|---|
| Svpeng | 54161 |
| PornDroid | 46082 |
| Koler | 44555 |
| RansomOB | 39859 |
| Charger | 39551 |
| Simplocker | 36340 |
| Wannalocker | 32701 |
| Jisut | 25672 |
| Lockerpin | 25307 |
| Pletor | 4715 |

#### *4.1.1. Feature selection*

Feature selection is a data preprocessing technique that aims to select a subset of features most relevant to the target variable. This first step involves selecting the best features in the dataset to

[1] https://www.kaggle.com/datasets/subhajournal/android-ransomware-detection?resource=download

**Figure 2**
**Dataset analysis process**



be used for the analysis process. This involved dropping some of the columns from the dataset that were not helpful in the analysis. First, the "Flow ID" column was dropped from the dataset as it was redundant. The attributes that constituted the "Flow ID" column were "Source internet protocol Internet Protocol (IP)," "Destination IP," "Source IP Port," "Destination IP Port," and "Protocol," which were already present as individual variables in the dataset, providing sufficient information to capture the flow characteristics. Consequently, the destination IP address was dropped from the dataset because the destination IP address is often used to identify the target of an attack, and in many cases, the same IP address can be used for benign and malicious traffic [90] source port, destination port, and protocol are more useful in identifying ransomware traffic [29, 30]. Lastly, the unnamed column was dropped as it was a numbering column used to assign a unique identifier to the data to maintain organization and reference. This column however lacks meaningful information for analysis and is primarily used for administrative purposes.

#### *4.1.2. Feature engineering*

Nargesian et al. [31] define feature engineering as the practice of construction of suitable features from a given set of features that leads to predictive performance. First, the "Timestamp" column was converted to a UTC zone and then split into three individual variables: "Month," "Day," and "HourMinute." The "Minute" and "Hour" columns were combined to make one column. This extraction of individual components allowed for a more granular representation of the temporal aspects of the data. Once the individual variables were derived, the original "Timestamp" column was no longer needed and was dropped from the dataset. The "Year" column was dropped as the data was acquired within a short period, and the year component would not provide any additional insight for the intended analysis. The "Minute" and "Hour" columns were combined to make one column to provide a coarser temporal resolution, which will give insight into identifying patterns at an hourly level rather than minute-by-minute variation.

Finally, for the non-numerical variables that are represented as integers, we performed one-hot encoding and integer encoding to convert them into numerical representations. Among the variables that needed to be converted were source IP, protocol, source port, destination port, and label. For source IP, source port, destination ports, and label integer encoding were performed. In the case of the protocol, one-hot encoding was performed due to the number of ports available in the dataset. The dataset contained only three protocols, which included User Datagram Protocol (UDP), Transmission Control Protocol (TCP), and Hop-by-Hop IPv6 extension header (Hop-by-Hop IPv6 extension header (HOPOPT) ports.

The timestamp splitting that is dividing the time-based data into more granular intervals helps capture the behavioral patterns. However, these features often lead to an increase in dimensionality of the dataset and make detection hard for resource constraint devices. Fortunately, the dimensionality reduction such principal component analysis (PCA) can help maintain high accuracy.

#### *4.1.3. Data normalization*

Normalization is the process of mapping a range of values that a numerical characteristic can assume to a standard range of values to either [−1,1] or [0,1]. The success of ML algorithms depends on the data quality; therefore, it is important to normalize the data. Data normalization helps in bringing features to a similar scale, prevents certain features with larger numerical ranges from dominating the learning process, and ensures a fair contribution of each feature to the model's performance. This study applied a general Z-score normalization technique using the Standard Scaler class from the scikit-learn library. The Z-score normalization, also known as standardization, transforms the data, where each feature has a mean of zero and a standard deviation of one [32]. By utilizing the Standard Scaler class, the dataset's features were scaled individually. The process involved subtracting the mean value of each feature and dividing it by the standard deviation. This ensures all features are centered around 0 with a unit standard deviation. Formula 5.1 denotes the equation of Z-score normalization, where $\mu$ is the mean and $\sigma$ is the standard deviation [33].

We choose Z-score normalization to ensure fair scaling across features. In contrast to the min-max normalization that scales the features to [0,1] range, the Z-score transforms the features to have a mean of 0 and standard deviation of 1 while preserving their original distribution. From our exploratory analysis, we discovered a non-uniform distribution in critical features like packet sizes, timestamps, and connection durations, which were important to preserve. This made the Z-score a better choice as these critical features are important in the classification of ransomware. The classifiers explored in this study are sensitive to data scaling and, if provided with compressed data, would result in misleading results. Another reason is the non-uniform distribution of features such as timestamp that may contain outliers; the min-max is sensitive to these types of outliers and can distort meaningful variance between benign and ransomware behavior.

### 4.2. Machine learning algorithms

#### *4.2.1. Random forest*

The RF model in this study was created using the RandomForestClassifier from scikit-learn. To find the best features for this model, an experiment was run using scikit-learn's Gridsearch () with the given number of trees between one and 150 and the maximum depth from one to 30. Through the experiment, the best number of trees was 140, and the maximum depth was 30. The Gridsearch method also performed 10-fold cross-validation and used the average score as a more accurate estimate of the model's ability to classify the data. These values were then used to train the model before testing its accuracy with the test data.

#### *4.2.2. K-nearest neighbors*

This study used Scikit-Learn's Gridsearch to find the best combination of k and the best distance metric. The best number of neighbors is one, and the best distance is Euclidean, according to the results from Gridsearch. The range of values given to the Gridsearch for k was one to 20, and the distance metrics were Euclidean, Manhattan, and Chebyshev. The model was then trained using the training data and tested with the test data.

#### *4.2.3. SVM*

GridSearch was employed to fine-tune the SVM model. The function was given kernels: linear, polynomial, sigmoid, and radial basis function (RBF), while C and Gamma were 0.01, 0.1, 1, and 10 for both parameters. The best combination of hyperparameters, namely, an RBF kernel, gamma set at 0.1, and a regularization parameter C equal to 1000, was identified through the GridSearch process. Gridsearch also performed a 10-fold cross-validation to find the optimal model to be trained. This optimized SVM model was then trained on the provided training data, effectively learning the underlying data relationships for robust classification. Its classification was then evaluated using the test data.

*4.2.4. Neural network*

Like other algorithms in this study, we ran an experiment using scikit-learn's Gridsearch to find the best parameters for the best model with activation functions: identity, logistic, tanh, and ReLu; solver: lbfgs, sgd, and adam; and the learning rate: constant, Invisalign, and adaptive as the parameters. A constant learning rate, Adam solver, and activation of tanh gave the best model. The model was then trained on the training data and tested on the test data using the evaluation metrics discussed in the following section.

## 5. Performance Evaluation

In this section, we will evaluate the performance of the four ML classifiers: SVM, KNN, NN, and RF. This section discusses their performance individually and their results. The following section will compare the performance of these classifiers together given the evaluation metrics. The study will also shed light on the limitations of this study. This chapter discusses the results of the study in detail. An overview of the ML classifiers will be given and the performance of each classifier together with the classification report. The details of the performance will be discussed, and the four models will also be compared in terms of their overall accuracy, precision, recall, and $F1$-score. We choose the above mentioned models because of their efficiency and effectiveness in use cases where ransomware reveals localized feature similarity to previously known samples. The NN selection is due to the ability to model high nonlinear relationships and intricate feature interactions that are very common in recent ransomware variants. Furthermore, RF is used to deal with heterogeneous feature sets and its robustness against overfitting. The SVM model provides high performance on high-dimensional datasets and finds an optimal boundary that is important when classifying ransomware behavior from benign.

### 5.1. Evaluation metrics

Here in this section, we give an overview of the five evaluation metrics used in this study, including accuracy, precision, recall, cross-validation, and $F1$-score.

*5.1.1. Cross-validation*

Cross-validation is a data resampling technique used to evaluate the performance of ML algorithms by splitting the dataset into training and testing. It helps to assess how the model performs on unseen data and reduce the possibility of overfitting [34]. This study used a 10-fold cross-validation method that involves partitioning the original data (train set) into a set of subsets or folds of 10. The model is then trained on all but one of the folds, which will be used as a test set. The process of training is then repeated until each of the folds has served as a test set once. The model's performance is then averaged over all iterations to provide a more accurate estimate of its ability to generalize unseen data.

*5.1.2. Accuracy*

Accuracy is a common metric used to evaluate the performance of a classification model. It measures the percentage of correctly predicted instances out of the total number of instances. A high accuracy is indicative of the model being able to make accurate predictions. Accuracy can be calculated using the formula given as follows:

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

where $TP$ represents the number of true positives, $TN$ is the number of true negatives, $FP$ is the number of false positives, and $FN$ is the number of false negatives [35].

*5.1.3. Precision*

Precision is another performance evaluation metric used for classification purposes, which deals with an imbalanced dataset. A high precision score indicates that the model has a low rate of incorrectly predicting positive instances. Precision can be calculated using the given formula, where TP represents the number of true positives, TN is the number of true negatives, and FP is the number of false positives [36].

$$Precision = \frac{TP}{TP + FP}$$

*5.1.4. Recall*

Recall measures the model's ability to identify positive instances correctly. It calculates the ratio of true positive predictions to the total number of actual positive instances in the dataset. Recall is particularly important in scenarios where identifying all positive instances is critical, such as in medical diagnoses or fraud detection. A high recall number mean the model can effectively capture the most positive instances.

$$Recall = \frac{TP}{TP + FN}$$

*5.1.5. F-1score*

The F-1 score is the hormonic mean of precision and recall; it balances a model's performance [36]. It is useful when there is a high difference between negative and positive classes in the dataset. The F1-score is important in this study as it is a ransomware detection problem where it is important to identify all false positive and false negative instances. F1-score can be calculated using the formula given as follows:

$$F_1 = \frac{Precision \times recall}{Precision + recall}$$

### 5.2. Experiments and results

The following section will compare the performance of these classifiers together given the evaluation metrics. The study will also shed light on the limitations of this study. This chapter discusses the experiment and results of the study in detail.

*5.2.1. Random forest*

The RF classifier outperformed the other models evaluated in this study, attaining an overall accuracy of 96.22%, precision of 95.95%, and recall of 94.52%. Notably, the RF model's configuration, determined through grid search during the training phase, comprised 140 DTs with a maximum depth of 30. This configuration enabled the model to effectively capture intricate patterns within the data while maintaining strong generalization capabilities, a crucial factor in developing robust ransomware detection systems. The classification report presented in Table 2 provides a detailed assessment of the RF model's performance across each class, evaluated using precision, recall, and F1-score metrics. The classification report shows that the RF model exhibited exceptional performance in identifying benign instances, achieving perfect precision, recall, and an F1-score of 1.0000. This result is particularly significant, as accurate discrimination between benign

and malicious instances is a fundamental requirement for effective ransomware detection systems.

**Table 2**
**Random forest classification report**

| Strain | Precision | Recall | F1-score |
|---|---|---|---|
| Benign | 1.0000 | 1.0000 | 1.0000 |
| Charger | 0.9450 | 0.9305 | 0.9377 |
| Jisut | 0.9464 | 0.9482 | 0.9473 |
| Koler | 0.9575 | 0.9665 | 0.9620 |
| Lockerpin | 0.9636 | 0.9287 | 0.9458 |
| Pletor | 0.9277 | 0.7799 | 0.8474 |
| PornDroid | 0.9558 | 0.9844 | 0.9699 |
| RansomOB | 0.9582 | 0.9507 | 0.9544 |
| SVpeng | 0.9504 | 0.9742 | 0.9622 |
| Simplocker | 0.9728 | 0.9584 | 0.9656 |
| WannaLocker | 0.9773 | 0.9755 | 0.9764 |

For the majority of ransomware classes, including Charger, Jisut, Koler, Lockerpin, PornDroid, RansomBO, SVpeng, Simplocker, and WannaLocker, the RF model demonstrated a strong balance between precision and recall, with F1-scores ranging from 0.9377 to 0.9764. This consistent performance across diverse ransomware classes highlights the model's robustness and adaptability to varying ransomware characteristics. However, the RF model exhibited relatively lower recall for the Pletor class, with a value of 0.7799, though maintaining a reasonable precision of 0.9277. This discrepancy suggests that the model may struggle to accurately identify certain instances of the Pletor ransomware, potentially due to unique characteristics or limited representation within the training data. Addressing this limitation through techniques such as data augmentation or class-specific optimization could further enhance the model's performance for this class.

The superior performance of the RF model can be attributed to its ability to leverage the strengths of multiple DTs, each capable of capturing diverse patterns within the data. Furthermore, the model's configuration, optimized through grid search, played a crucial role in striking a balance between complexity and generalization, enabling the model to effectively distinguish between benign and malicious instances while maintaining robustness across various ransomware classes. Despite the promising results, there remains scope for further improvement, particularly in addressing class imbalances or fine-tuning the model's hyperparameters for specific classes that exhibited lower performance. Additionally, incorporating ensemble techniques such as boosting or stacking could potentially enhance the model's overall accuracy and consistency across all classes. The lightweight RF model with this accuracy can immediately evaluate behavioral features in near real-time to flag a ransomware-like activity or behavior. Further, due to its low computational overhead, the RF model can be deployed directly on Android devices, avoiding constant communication with external third-party service providers, and can be scaled to up to millions of devices in near real-time.

*5.2.2. K-nearest neighbors*

The KNN model exhibited the lowest overall accuracy of 70.49% among the four classifiers evaluated in this study. During the training phase, grid search determined the optimal configuration for the KNN model, which comprised a single neighbor and the Euclidean distance metric. The model attained a precision of 71.16% and a recall of 70.49%. The classification report presented in Table 3 provides a detailed analysis of the KNN model's performance across each ransomware class, evaluated using precision, recall, and F1-score metrics.

As evidenced by the classification report, the KNN model demonstrated a notable strength in identifying benign instances, achieving a precision of 99.34%, a recall of 99.41%, and an impressive F1-score of 99.37%. This result underscores the model's ability to discriminate between benign and malicious instances accurately, a crucial factor in developing effective ransomware detection systems. The KNN model exhibited varying levels of performance across different ransomware classes. For instance, the Charger class demonstrated a relatively low precision of 0.5160 and a recall of 0.6909, resulting in an F1-score of 0.5907. Conversely, the KNN model exhibited moderate to balanced performance for several classes, including Jisut, Koler, Lockerpin, Pletor, PornDroid, and WannaLocker, with F1-scores ranging from 0.6440 to 0.7652. These results are comparable and consistent with the other classifiers in this study.

**Table 3**
**KNN classification report**

| Strain | Precision | Recall | F1-score |
|---|---|---|---|
| Benign | 0.9934 | 0.9941 | 0.9937 |
| Charger | 0.5160 | 0.6909 | 0.5907 |
| Jisut | 0.6607 | 0.7346 | 0.6957 |
| Koler | 0.7194 | 0.7725 | 0.7450 |
| Lockerpin | 0.6699 | 0.6321 | 0.6505 |
| Pletor | 0.6351 | 0.6352 | 0.6440 |
| PornDroid | 0.7839 | 0.7473 | 0.7652 |
| RansomOB | 0.6029 | 0.6313 | 0.6028 |
| SVpeng | 0.6857 | 0.6202 | 0.6513 |
| Simplocker | 0.6837 | 0.5416 | 0.6044 |
| WannaLocker | 0.7539 | 0.6797 | 0.7149 |

The performance of the KNN model can be attributed to its ability to leverage the similarity between instances, making it well-suited for scenarios where the underlying data distribution is relatively consistent. However, the model's reliance on distance metrics and the number of neighbors may limit its performance in scenarios with complex decision boundaries or high-dimensional feature spaces, as demonstrated by the superior performance of deep learning and ensemble methods in certain cases. Furthermore, it is essential to consider the trade-offs between model complexity and interpretability when selecting a suitable classifier for ransomware detection. While more complex models, such as deep learning architectures, may offer higher accuracy, they often lack transparency and interpretability, which can be crucial in security-critical applications. Overall, the KNN model demonstrates its potential as a viable option for ransomware detection, particularly when simplicity and interpretability are prioritized. However, further optimization and careful consideration of the specific requirements and constraints of the application domain are necessary to ensure the model's effectiveness and reliability. Additionally, incorporating insights and techniques from other studies in the field may aid in improving the model's performance and addressing specific limitations or challenges.

*5.2.3. Neural network*

The NN classifier achieved an overall accuracy of 81.91%, with a precision of 82.74% and a recall of 81.49%. To identify the optimal configuration for the multilayer perceptron model, grid search was employed to tune the hyperparameters. The best configuration was determined to be the Adam solver with a constant learning rate and the Tanh activation function. The classification report, summarized in Table 4, provides insights into the model's performance for each ransomware class and the benign class. As shown by the classification report, the NN model exhibited remarkable performance in identifying benign instances, achieving perfect precision, recall, and an F1-score of 1.0000 in detecting benign traffic. These results highlight how accurately the model classifies the benign and ransomware instances – a critical requirement for effective ransomware detection systems.

However, certain classes presented challenges for the model. The Charger class demonstrated moderate precision (0.7002) and recall (0.6977), with an F1-score of 0.6990, indicating room for improvement in the model's ability to identify instances of this particular ransomware strain consistently. Similarly, the RansomBO class exhibited lower precision (0.6969) and recall (0.5891), resulting in an F1-score of 0.6385, suggesting potential limitations in the model's capacity to generalize to this class.

**Table 4**
**Neural network classification report**

| Strain | Precision | Recall | F1-score |
|---|---|---|---|
| Benign | 1.0000 | 1.0000 | 1.0000 |
| Charger | 0.7002 | 0.6977 | 0.6990 |
| Jisut | 0.9056 | 0.8837 | 0.8945 |
| Koler | 0.8993 | 0.9222 | 0.9106 |
| Lockerpin | 0.8984 | 0.8581 | 0.8778 |
| Pletor | 0.8331 | 0.6405 | 0.7242 |
| PornDroid | 0.8939 | 0.9261 | 0.9097 |
| RansomOB | 0.6969 | 0.5891 | 0.6385 |
| SVpeng | 0.6728 | 0.7710 | 0.7185 |
| Simplocker | 0.7864 | 0.7668 | 0.7765 |
| WannaLocker | 0.8147 | 0.7988 | 0.8067 |

These results suggest that the NN model effectively captured the distinguishing characteristics of these ransomware variants, enabling accurate classification. However, certain classes presented challenges for the model. The Charger class demonstrated moderate precision (0.7002) and recall (0.6977), with an F1-score of 0.6990, indicating room for improvement in the model's ability to identify instances of this particular ransomware strain consistently. Similarly, the RansomBO class exhibited lower precision (0.6969) and recall (0.5891), resulting in an F1-score of 0.6385, suggesting potential limitations in the model's capacity to generalize to this class. Notably, the Pletor class exhibited a significant discrepancy between precision (0.8331) and recall (0.6405), leading to an F1-score of 0.7242. This imbalance could be attributed to factors such as class imbalance, overlapping feature representations, or the model's inability to capture the unique characteristics of this ransomware variant effectively.

*5.2.4. Support vector machine (SVM)*

It is evident from the results that the SVM classifier achieved an overall accuracy of 83.51%, with a precision of 83.43%, a recall of 82.36%, and an F1-score of 82.81%. To obtain the best model configuration, grid search was employed for hyperparameter tuning. The optimal SVM model was developed using an RBF kernel, a gamma value of 0.1, and a C value of 1000. These parameters yielded the highest performance for the SVM classifier as shown in Table 5.

**Table 5**
**SVM classification report**

| Strain | Precision | Recall | F1-score |
|---|---|---|---|
| Benign | 0.9995 | 0.9970 | 0.9983 |
| Charger | 0.7810 | 0.8297 | 0.8046 |
| Jisut | 0.8476 | 0.8699 | 0.8586 |
| Koler | 0.8505 | 0.9144 | 0.8813 |
| Lockerpin | 0.8466 | 0.8388 | 0.8427 |
| Pletor | 0.8270 | 0.7200 | 0.7698 |
| PornDroid | 0.9002 | 0.8757 | 0.8878 |
| RansomBO | 0.7370 | 0.7530 | 0.7449 |
| SVpeng | 0.7835 | 0.7949 | 0.7891 |
| Simplocker | 0.7705 | 0.7949 | 0.7355 |
| Wannalocker | 0.8335 | 0.7630 | 0.7967 |

The SVM model exhibited exceptional prowess in correctly identifying benign instances, attaining a precision of 0.9995, a recall of 0.9970, and an F1-score of 0.9983 in identifying positive benign cases of traffic. This performance highlights the model's efficiency in accurately discriminating between benign and malicious traffic, a critical prerequisite for effective ransomware detection systems. However, the SVM model demonstrated varying degrees of proficiency across different ransomware classes. For instance, while the Charger class exhibited moderate precision (0.7810) and recall (0.8297), resulting in an F1-score of 0.8046, the Jisut class displayed higher precision (0.8476) and recall (0.8699), culminating in an F1-score of 0.8586, signifying the model's aptitude in accurately classifying instances of this ransomware strain. Notably, the Koler class attained the highest recall (0.9144) among all ransomware classes, coupled with a precision of 0.8505 and an F1-score of 0.8813. This result underscores the SVM model's efficacy in identifying instances of the Koler ransomware variant.

While the Lockerpin class exhibited a balanced precision (0.8466) and recall (0.8388), with an F1-score of 0.8427, the Pletor class demonstrated a discrepancy between precision (0.8270) and recall (0.7200), resulting in a lower F1-score of 0.7698. This discrepancy suggests potential limitations in the model's ability to identify certain instances of the Pletor ransomware accurately.

In contrast, the RansomBO class exhibited the lowest precision (0.7370) among all classes, coupled with a recall of 0.7530 and an F1-score of 0.7449, highlighting potential challenges in accurately classifying instances of this ransomware strain.

## 6. Conclusion and Future Work

The rising threat of Android ransomware has made it crucial to develop effective detection and prevention mechanisms to protect user data and prevent financial losses. This research work evaluates the performance of various ML algorithms, including KNN, SVM, RF, and NN, in detecting Android ransomware. By conducting a comparative analysis of these classifiers, the research provides valuable insights into their strengths and weaknesses, enabling informed

decision-making for selecting appropriate models. The research involves feature engineering and dataset analysis, contributing to a better understanding of Android ransomware characteristics. The findings and methodologies employed can enhance Android security, advance malware detection research, guide industry adoption of effective ML models, and raise awareness about evolving ransomware threats. Ultimately, this work will help the researcher and practitioner improve mobile device security, advance cybersecurity research, and promote overall cybersecurity education.

In the future, we aim to explore ensemble and hybrid approaches that combine ML with other network traffic analysis techniques like deep packet inspection or flow analysis for improved ransomware detection accuracy. The ensemble method that includes stacking and boosting enhances the model's performance by combining the predictive strengths of multiple classifiers. The diverse learning biases can be integrated (RF for feature robustness and SVM for margin optimization) to address the problem of false negatives when dealing with zero-day ransomware. This helps the model to detect ransomware that deviates from well-known patterns. Developing online learning strategies to handle concept drift and adversarial network behavior would ensure the models remain effective against evolving ransomware communication patterns. Leveraging explainable AI techniques could provide insights into the network features and traffic patterns that characterize ransomware behavior, aiding interpretability. Optimizing the models for efficient deployment in network monitoring and security appliances is crucial for real-time detection without impacting network performance. Extending the research to analyze traffic from other platforms or incorporating host-based features like system calls or API monitoring could lead to more comprehensive detection systems. Collaborating with industry partners to integrate the developed ML models into existing network security solutions would address challenges related to deployment, scalability, and real-world performance evaluation in diverse network environments.

## Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

## Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

## Data Availability Statement

The data that support the findings of this study are openly available in Kaggle at https://www.kaggle.com/datasets/subhajournal/android-ransomware-detection?resource=download.

## Author Contribution Statement

**Linet Momposhi:** Conceptualization, Methodology, Software, Formal analysis, Writing – original draft, Writing – review & editing, Visualization. **Alana Maurushat:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Rodrigo N. Calheiros:** Validation, Investigation, Resources, Data curation, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Shawal Khan:** Validation, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration.

## References

[1] Albin Ahmed, A., Shaahid, A., Alnasser, F., Alfaddagh, S., Binagag, S., & Alqahtani, D. (2023). Android ransomware detection using supervised machine learning techniques based on traffic analysis. *Sensors*, *24*(1), 189. https://doi.org/10.3390/s24010189

[2] Oz, H., Aris, A., Levi, A., & Uluagac, A. S. (2022). A survey on ransomware: Evolution, taxonomy, and defense solutions. *ACM Computing Surveys*, *54*(11s), 1–37. https://doi.org/10.1145/3514229

[3] Razavi, H., Jamali, M. R., Emsaki, M., Ahmadi, A., & Hajiaghei-Keshteli, M. (2023). Quantifying the financial impact of cyber security attacks on banks: A big data analytics approach. In *2023 IEEE Canadian Conference on Electrical and Computer Engineering*, 533–538. https://doi.org/10.1109/CCECE58730.2023.10288963

[4] Razavi, H., & Habibnia, A. (2024). The rise of AI in Middle Eastern fintech with the case studies from the UAE and Turkey. In H. Taherdoost, N. Le, & M. Madanchian (Eds.), *Exploring global FinTech advancement and applications* (pp. 259–297). IGI Global Scientific Publishing. https://doi.org/10.4018/979-8-3693-1561-3.ch010

[5] Guerra-Manzanares, A. (2024). Machine learning for android malware detection: Mission accomplished? A comprehensive review of open challenges and future perspectives. *Computers & Security*, *138*, 103654. https://doi.org/10.1016/j.cose.2023.103654

[6] Atanassov, N., & Chowdhury, M. M. (2021). Mobile device threat: Malware. In *2021 IEEE International Conference on Electro Information Technology*, 7–13. https://doi.org/10.1109/EIT51626.2021.9491845

[7] Humayun, M., Jhanjhi, N. Z., Alsayat, A., & Ponnusamy, V. (2021). Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*, *22*(1), 105–117. https://doi.org/10.1016/j.eij.2020.05.003

[8] Ryan, M. (2021). *Ransomware revolution: The rise of a prodigious cyber threat*. Switzerland: Springer Cham.

[9] Cisco. (n.d). *How is OT different from IT? OT vs. IT*. Retrieved from: https://www.cisco.com/c/en/us/solutions/internet-of-things/what-is-ot-vs-it.html

[10] Mukhtar, B. I., Elsayed, M. S., Jurcut, A. D., & Azer, M. A. (2023). IoT vulnerabilities and attacks: SILEX malware case study. *Symmetry*, *15*(11), 1978. https://doi.org/10.3390/sym15111978

[11] Razaulla, S., Fachkha, C., Markarian, C., Gawanmeh, A., Mansoor, W., Fung, B. C., & Assi, C. (2023). The age of ransomware: A survey on the evolution, taxonomy, and research directions. *IEEE Access*, *11*, 40698–40723. https://doi.org/10.1109/ACCESS.2023.3268535

[12] Zuhri, F. A. (2019). *The illusion of the cyber intelligence era*. ZAHF.ME.

[13] Senanayake, J., Kalutarage, H., & Al-Kadri, M. O. (2021). Android mobile malware detection using machine learning: A systematic review. *Electronics*, *10*(13), 1606. https://doi.org/10.3390/electronics10131606

[14] Sihag, V., Swami, A., Vardhan, M., & Singh, P. (2020). Signature based malicious behavior detection in android. In *International Conference on Computing Science, Communication*

*and Security, 1235*, 251–262. https://doi.org/10.1007/978-981-15-6648-6_20

[15] Sihag, V., Vardhan, M., Singh, P., Choudhary, G., & Son, S. (2021). De-LADY: Deep learning based android malware detection using dynamic features. *Journal of Internet Services and Information Security*, *11*(2), 34–45. https://doi.org/10.22667/JISIS.2021.05.31.034

[16] Vanjire, S., & Lakshmi, M. (2021). Behavior-based malware detection system approach for mobile security using machine learning. In *2021 International Conference on Artificial Intelligence and Machine Vision*, 1–4. https://doi.org/10.1109/AIMV53313.2021.9671009

[17] Aslan, Ö. A., & Samet, R. (2020). A comprehensive review on malware detection approaches. *IEEE Access*, *8*, 6249–6271. https://doi.org/10.1109/ACCESS.2019.2963724

[18] Jannat, U. S., Hasnayeen, S. M., Shuhan, M. K. B., & Ferdous, M. S. (2019). Analysis and detection of malware in android applications using machine learning. In *2019 International Conference on Electrical, Computer and Communication Engineering*, 1–7. https://doi.org/10.1109/ECACE.2019.8679493

[19] Elayan, O. N., & Mustafa, A. M. (2021). Android malware detection using deep learning. *Procedia Computer Science*, *184*, 847–852. https://doi.org/10.1016/j.procs.2021.03.106

[20] Bhatia, T., & Kaushal, R. (2017). Malware detection in android based on dynamic analysis. In *2017 International Conference on Cyber Security and Protection of Digital Servicesm*, 1–6. https://doi.org/10.1109/CyberSecPODS.2017.8074847

[21] Onwuzurike, L., Almeida, M., Mariconti, E., Blackburn, J., Stringhini, G., & De Cristofaro, E. (2018). A family of droids-android malware detection via behavioral modeling: Static vs dynamic analysis. In *2018 16th Annual Conference on Privacy, Security and Trust*, 1–10. https://doi.org/10.1109/PST.2018.8514191

[22] Feng, P., Ma, J., Sun, C., Xu, X., & Ma, Y. (2018). A novel dynamic android malware detection system with ensemble learning. *IEEE Access*, *6*, 30996–31011. https://doi.org/10.1109/ACCESS.2018.2844349

[23] Kim, Y. K., Lee, J. J., Go, M. H., Kang, H. Y., & Lee, K. (2022). A systematic overview of the machine learning methods for mobile malware detection. *Security and Communication Networks*, *1*, 8621083. https://doi.org/10.1155/2022/8621083

[24] Narudin, F. A., Feizollah, A., Anuar, N. B., & Gani, A. (2016). Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing*, *20*, 343–357. https://doi.org/10.1007/s00500-014-1511-6

[25] Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., & Liu, H. (2020). A review of android malware detection approaches based on machine learning. *IEEE Access*, *8*, 124579–124607. https://doi.org/10.1109/ACCESS.2020.3006143

[26] Ispahany, J., Islam, M. R., Islam, M. Z., & Khan, M. A. (2024). Ransomware detection using machine learning: A review, research limitations and future directions. *IEEE Access*, *12*, 68785–68813. https://doi.org/10.1109/ACCESS.2024.3397921

[27] Alzahrani, S., Xiao, Y., Asiri, S., Zheng, J., & Li, T. (2025). A survey of ransomware detection methods. *IEEE Access*, *13*, 57943–57982. https://doi.org/10.1109/ACCESS.2025.3556187

[28] Usha, G., Madhavan, P., Cruz, M. V., Vinoth, N. A. S., & Nancy, M. (2021). Enhanced ransomware detection techniques using machine learning algorithms. In *2021 4th International Conference on Computing and Communications Technologies*, 52–58. https://doi.org/10.1109/ICCCT53315.2021.9711906

[29] Ferriyan, A., Thamrin, A. H., Takeda, K., & Murai, J. (2021). Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic. *Applied Sciences*, *11*(17), 7868. https://doi.org/10.3390/app11177868

[30] Bold, R., Al-Khateeb, H., & Ersotelos, N. (2022). Reducing false negatives in ransomware detection: A critical evaluation of machine learning algorithms. *Applied Sciences*, *12*(24), 12941. https://doi.org/10.3390/app122412941

[31] Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E. B., & Turaga, D. S. (2017). Learning feature engineering for classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2529–2535.

[32] Ali, P. J. M., Faraj, R. H., Koya, E., Ali, P. J. M., & Faraj, R. H. (2014). Data normalization and standardization: A technical report. *Machine Learning Technical Reports*, *1*(1), 1–6. https://doi.org/10.13140/RG.2.2.28948.04489

[33] Singh, D., & Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, *97*, 105524. https://doi.org/10.1016/j.asoc.2019.105524

[34] Berrar, D. (2018). Cross-validation. In F. Pentimalli, A. Giordano, S. O'Neil, S. Ranganathan, S. Shhartstein, D. Yelon, . . . , & B. Dixon (Eds.), *Reference module in life sciences*. Elsevier. https://doi.org/10.1016/B978-0-12-809633-8.20349-X

[35] Noorbehbahani, F., Rasouli, F., & Saberi, M. (2019). Analysis of machine learning techniques for ransomware detection. In *2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology*, 128–133. https://doi.org/10.1109/ISCISC48546.2019.8985139

[36] Hirano, M., Hodota, R., & Kobayashi, R. (2022). RanSAP: An open dataset of ransomware storage access patterns for training machine learning models. *Forensic Science International: Digital Investigation*, *40*, 301314. https://doi.org/10.1016/j.fsidi.2021.301314

**How to Cite:** Momposhi, L., Maurushat, A., Calheiros, R. N., & Khan, S. (2025). Next-Gen Threat Hunting: A Comparative Study of ML Models in Android Ransomware Detection. *FinTech and Sustainable Innovation*. https://doi.org/10.47852/bonviewFSI52025685

## Appendix 1

| Variable Name | Variable Definition | Variable Example |
|---|---|---|
| ID | A unique identifier for each data record in the dataset. | 0 |
| Flow ID | A unique identifier for a specific network flow (sequence of packets between two IP addresses). | 172.217.2.174 - 10.42.0.211-443-51023-6 |
| Source IP | The IP address of the sender in the network flow. | 10.42.0.211 |
| Source port | The port number used by the sender in the network flow. | 51023 |
| Destination IP | The IP address of the receiver in the network flow. | 172.217.2.174 |
| Destination Port | The port number used by the receiver in the network flow. | 443 |
| Protocol | The network protocol used for communication (e.g., TCP, UDP, ICMP). | 6 |
| Timestamp | The time when the network flow was recorded. | ###### |
| Flow Duration | The total time duration of the network flow in microseconds. | 151054 |
| Total Fwd Packets | The total number of packets sent by the sender (forward direction) in the network flow. | 6 |
| Total Backward Packets | The total number of packets sent by the receiver (backward direction) in the network flow. | 8 |
| Total Length of Fwd packets | The total length (in bytes) of all packets sent by the sender (forward direction) in the network flow. | 1076 |
| Total Length of Backward packets | The total length (in bytes) of all packets sent by the receiver (backward direction) in the network flow. | 4575 |
| Fwd Packet Length Max | The maximum length of a single packet sent by the sender (forward direction) in the network flow. | 821 |
| Fwd Packet Length Min | The minimum length of a single packet sent by the sender (forward direction) in the network flow. | 0 |
| Fwd Packet Length Mean | The average length of packets sent by the sender (forward direction) in the network flow. | 179.3333333 |
| Fwd Packet Length Std | The standard deviation of the length of packets sent by the sender (forward direction) in the network flow. | 321.6219313 |
| Bwd Packet Length Max | The maximum length of a single packet sent by the receiver (backward direction) in the network flow. | 1418 |
| Bwd Packet Length Min | The minimum length of a single packet sent by the receiver (backward direction) in the network flow. | 0 |
| Bwd Packet Length Mean | The average length of packets sent by the receiver (backward direction) in the network flow. | 571.875 |
| Bwd Packet Length Std | The standard deviation of the length of packets sent by the receiver (backward direction) in the network flow. | 679.532284 |

(*Continued*)

(*Continued*)

| | | |
|---|---|---|
| Flow Byte/s | The average number of bytes per second transmitted in the network flow. | 37410.46248 |
| Flow Packet/s | The average number of packets per second transmitted in the network flow. | 92.6820872 |
| Flow IAT Mean | The average inter-arrival time (time between consecutive packets) in the network flow. | 11619.53846 |
| Flow IAT Std | The standard deviation of the inter-arrival time in the network flow. | 14541.15588 |
| Flow IAT Max | The maximum inter-arrival time in the network flow. | 49105 |
| Flow IAT Min | The minimum inter-arrival time in the network flow. | 24 |
| Fwd IAT Total | The total inter-arrival time of packets sent by the sender (forward direction) in the network flow. | 101887 |
| Fwd IAT Mean | The average inter-arrival time of packets sent by the sender (forward direction) in the network flow. | 20377.4 |
| Fwd IAT Std | The standard deviation of the inter-arrival time of packets sent by the sender (forward direction) in the network flow. | 12821.55074 |
| Fwd IAT Max | The maximum inter-arrival time of packets sent by the sender (forward direction) in the network flow. | 30425 |
| Fwd IAT Min | The minimum inter-arrival time of packets sent by the sender (forward direction) in the network flow. | 111 |
| Bwd IAT Total | The total inter-arrival time of packets sent by the receiver (backward direction) in the network flow. | 128516 |
| Bwd IAT Mean | The average inter-arrival time of packets sent by the receiver (backward direction) in the network flow. | 18359.42857 |
| Bwd IAT Std | The standard deviation of the inter-arrival time of packets sent by the receiver (backward direction) in the network flow. | 24038.55786 |
| Bwd IAT Max | The maximum inter-arrival time of packets sent by the receiver (backward direction) in the network flow. | 54822 |
| Bwd IAT Min | The minimum inter-arrival time of packets sent by the receiver (backward direction) in the network flow. | 24 |
| Fwd PSH Flags | The number of packets with the PSH flag set sent by the sender (forward direction) in the network flow. | 0 |
| Bwd PSH Flags | The number of packets with the PSH flag set sent by the receiver (backward direction) in the network flow. | 0 |
| Fwd URG Flags | The number of packets with the URG flag set sent by the sender (forward direction) in the network flow. | 0 |
| Bwd URG Flags | The number of packets with the URG flag set sent by the receiver (backward direction) in the network flow. | 0 |
| Fwd Header Length | The total length of packet headers sent by the sender (forward direction) in the network flow. | 200 |

(*Continued*)

(*Continued*)

| | | |
|---|---|---|
| Bwd Header length | The total length of packet headers sent by the receiver (backward direction) in the network flow. | 264 |
| Fwd Packet/s | The average number of packets per second sent by the sender (forward direction) in the network flow. | 39.72089451 |
| Bwd Packets/s | The average number of packets per second sent by the receiver (backward direction) in the network flow. | 52.96119269 |
| Min Packet length | The minimum length of a single packet in the network flow. | 0 |
| Max Packet length | The maximum length of a single packet in the network flow. | 1418 |
| Packet Length Mean | The average length of packets in the network flow. | 376.7333333 |
| Packet Length std | The standard deviation of the length of packets in the network flow. | 562.5149479 |
| Packet Length Variance | The variance of the length of packets in the network flow. | 316423.0667 |
| FIN Flag Count | The number of packets with the FIN flag set in the network flow. | 0 |
| SYN Flag Count | The number of packets with the SYN flag set in the network flow. | 0 |
| RST Flag Count | The number of packets with the RST flag set in the network flow. | 0 |
| PSH Flag Count | The number of packets with the PSH flag set in the network flow. | 1 |
| ACK Flag Count | The number of packets with the ACK flag set in the network flow. | 0 |
| URG Flag Count | The number of packets with the URG flag set in the network flow. | 0 |
| CWE Flag Count | The number of packets with the CWE flag set in the network flow. | 0 |
| ECE Flag Count | The number of packets with the ECE flag set in the network flow. | 0 |
| Down/Up Ratio | The ratio of the download (receiver) and upload (sender) rates in the network flow. | 1 |
| Average Packet Size | The average size of packets in the network flow. | 403.6428571 |
| Avg Fwd Segment Size | The average size of segments sent by the sender (forward direction) in the network flow. | 179.3333333 |
| Avg Bwd Segment Size | The average size of segments sent by the receiver (backward direction) | 571.875 |
| Fwd Header Length.1 | A duplicate attribute for the total length of packet headers sent by the sender (forward direction) in the network flow. | 200 |
| Fwd Avg Bytes/Bulk | The average number of bytes per bulk in the forward direction in the network flow. | 0 |
| Fwd Avg Packets/Bulk | The average number of packets per bulk in the forward direction in the network flow. | 0 |
| Fwd Avg Bulk Rate | The average rate of bulk data transmission in the forward direction in the network flow. | 0 |
| Bwd Avg Bytes/Bulk | The average number of bytes per bulk in the backward direction in the network flow. | 0 |
| Bwd Avg Packets/Bulk | The average number of packets per bulk in the backward direction in the network flow. | 0 |
| Bwd Avg Bulk Rate | The average rate of bulk data transmission in the backward direction in the network flow. | 0 |
| Subflow Fwd Packets | The total number of packets in the forward direction subflow. | 6 |

(*Continued*)

(*Continued*)

| | | |
|---|---|---|
| Subflow Fwd Bytes | The total number of bytes in the forward direction subflow. | 1076 |
| Subflow Bwd Packets | The total number of packets in the backward direction subflow. | 8 |
| Subflow Bwd Bytes | The total number of bytes in the backward direction subflow. | 4575 |
| Init_Win_bytes_forward | The initial window size in bytes for the forward direction. | 65535 |
| Init_Win_bytes_backward | The initial window size in bytes for the backward direction | 353 |
| act_data_pkt_fwd | The number of packets with at least one byte of TCP data payload in the forward direction. | 3 |
| min_seg_size_forward | The minimum segment size in the forward direction. | 32 |
| Active Mean | The average time a flow was active before becoming idle. | 0 |
| Active Std | The standard deviation of the time a flow was active before becoming idle. | 0 |
| Active Max | The maximum time a flow was active before becoming idle. | 0 |
| Active Min | The minimum time a flow was active before becoming idle. | 0 |
| Idle Mean | The average time a flow was idle. | 0 |
| Idle Std | The standard deviation of the time a flow was idle. | 0 |
| Idle Max | The maximum time a flow was idle. | 0 |
| Idle Min | The minimum time a flow was idle. | 0 |
| Label | The class or category assigned to the network flow indicates whether it is related to ransomware or not. | Benign/ SVpeng/ PornDroid/ Koler/ RansomBO/ Charger/ Simplocker/ WannaLocker/ Jisut/ Lockerpin/ Pletor |