International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 15, No.1 (Feb-24)

http://dx.doi.org/10.12785/ijcds/150147

A Hybrid Approach Based On Boosting Algorithm For Effective Android Malware Detection

Brijesh Y. Sathwara[1] and Palvinder Singh Mann[2]

[1,2]*Graduate School of Engineering and Technology-Gujarat Technological University, Ahmedabad, Gujarat, INDIA. 382424.*

Received 10 May. 2023, Revised 6 Jan. 2024, Accepted 21 Jan. 2024, Published 1 Feb. 2024

Abstract: Mobile phones are being used much more often and play a crucial role in everyday lives. These gadgets hold a tonne of personal information and provide a variety of functions and services. Mobile devices have become indispensable for those who utilise technology and communication since they are practical and effective. However, mobile systems are vulnerable to virus assaults just like any other type of information system. The development of hardware technologies has increased the complexity and performance of mobile apps. Additionally, the danger of security breaches and data theft rises with continued usage of mobile devices. Malicious actors may use mobile system flaws to obtain sensitive data, such as login passwords, financial information, and personal information. Mobile device makers and app developers are constantly changing their software to enhance security and performance in order to solve these issues. For instance, to safeguard user data, many mobile operating systems now have built-in security measures like firewalls, encryption, and two-factor authentication. In this paper, we present a linear regression model for detecting malware on the Android platform. This technique can assist in the prompt identification and obstruction of Android malware assaults, as well as improve app security by flagging any unnecessary permissions. Additionally, developers can use this approach to enhance the security of their apps and protect user data from unauthorized access.

Keywords: Android malware, Malware detection, Regression model, Boosting algorithm.

1. Introduction

Android smartphones dominated the mobile OS market, taking more than 82 percent of the market share. The open-source structure of Android and a large number of freely available applications are mostly attributed for this rise. Due of this advantage over other operating systems, a significant portion of users now favour Android. Android smartphones are becoming more widely available to people all over the world because to the large selection of alternatives they offer at various price points. Additionally, the Android platform is flexible and customized, allowing users to modify their devices to meet their own requirements and tastes.Despite its global domination, Android has encountered issues such as security concerns and fragmentation as a result of the wide variety of devices and versions available. However, the Android development team has worked to solve these concerns, routinely delivering updates and security patches to assure the platform's security and stability.Overall, as more people seek inexpensive and configurable mobile devices, the popularity of Android smartphones is projected to expand in the future years. Android is expected to continue its position as the industry leader in the mobile operating system area, thanks to its open-source foundation and enormous library of applications.Google Play, Android's official app store, boasts over a billion apps and a massive amount of downloads per day. To protect against malware, the shop initially depended on a security technology called Bouncer, however this strategy proved ineffective against zero-day malware, which allowed hackers to take consumers' confidential data. Google Play now uses Google Play Protect to prevent insecure applications from being installed, however users may still disable this function during the installation process. This flaw has resulted in a considerable increase in the number of malware programmes posted to services like VirusTotal. Antimalware solutions address this issue with signature-based detection, which necessitates regular signature database updates and is vulnerable to code restructuring to evade detection [1].

Cloud-based malware detection solutions have been created to circumvent processor and energy constraints. Manual inspection and malware definition extraction, on the other hand, might result in false negatives, making it difficult to efficiently investigate new malware variants as their number grows.New approaches for studying fresh malware variants have been presented to answer the pressing demand for more efficient malware analysis tools. These strategies are intended to overcome the limits of current malware detection systems and minimise the frequency of false negatives. Instead than depending en-

E-mail address: brijeshsathwara5@gmail.com, asso_psmaan@gtu.edu.in

tirely on signature-based detection, one interesting strategy involves employing machine learning algorithms to detect malware based on behavioural patterns.Sandboxing is another common strategy that includes running the virus in a controlled environment to study its behaviour and discover any dangerous behaviours. Sandboxing can also aid in the prevention of malware propagating to other computers and networks.Overall, with the increasing quantity of malware apps, it is critical to create new, effective approaches for identifying and analysing these threats. It may be able to stay ahead of thieves and keep consumers' data safe by combining several strategies and constantly improving on them.Because of the continuously rising number of Android apps, unauthorised developers, and existing security holes in the Android operating system, malware makers are incentivized to attack these vulnerabilities in order to steal users' private information. [2].

As previously stated, the faulty Android OS provides an ideal environment for attackers. Some of the security vulnerabilities that can arise on Android phones include unauthorised app access, permission escalation, repackaging apps to incorporate malicious code, collusion, and Denial of Service (DoS) attacks. Given these flaws in the present Android design, several efforts have been made to remedy these concerns. In addition to Android's built-in security measures, such as sandboxing and the Android permission model, several security and privacy solutions, such as various resource management systems and security solutions that use different techniques and approaches, have been proposed to deal with the existing Android OS vulnerabilities. We will go deeper into these strategies, approaches, and tools.Antivirus software, intrusion detection systems, and firewalls are some of the security techniques used to reduce Android vulnerabilities. These technologies can aid in the detection and prevention of malware infecting devices and stealing personal information.To discover and address security flaws, developers might employ coding principles, secure coding frameworks, and code analysis tools. Furthermore, developers can use cryptographic techniques like as encryption and digital signatures to safeguard data privacy and prevent unauthorised access.Finally, Android vulnerabilities are a major source of concern for both consumers and app developers. While Android OS's open-source nature allows for innovation and flexibility, Android users and developers, on the other hand, may better defend themselves from harmful assaults and data breaches if suitable security measures are in place and rigorous development practises are followed [3].

The frequency of cyberattacks on mobile devices remained stable in Q1 2022 compared to the end of 2021 Fig.[1]. However, the general trend for the number of assaults continues to decline. this quarter is notable for the growth of fake apps in legitimate app stores. these applications frequently display inflated ratings and pleasant phoney reviews. This sort of programme is included in seven of the twenty applications on our Q1 malware ranking. Since

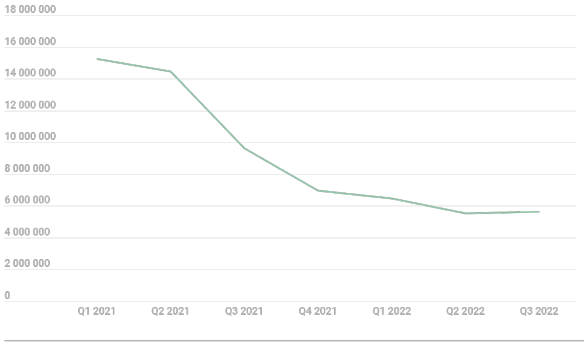

Figure 1. Number of attacks targeting users

last year, applications that offer social advantages have been a common fraud approach. These applications bring users to a homepage where they are requested for personal information and shown a significant quantity of money to which they are allegedly entitled. Users must, however, pay a commission for transfer charges or legal aid in order to enjoy the advantages.

A. *Mobile threat statistics*

During the first quarter of 2022, Kaspersky discovered 516,617 malware-infected installation packages Fig.[2]. This figure fell by 79,448 from the previous quarter and by 935,043 from Q1 2021.According to Kaspersky, the fall in malicious installation packages is mostly due to a decrease in the popularity of adware and potentially unwanted programmes (PUPs) among attackers. While the total number of detections has fallen, Kaspersky cautions that the number of unique malware samples has climbed by 24% compared to the previous quarter, indicating that attackers are increasingly utilising new and more complex malware variants.

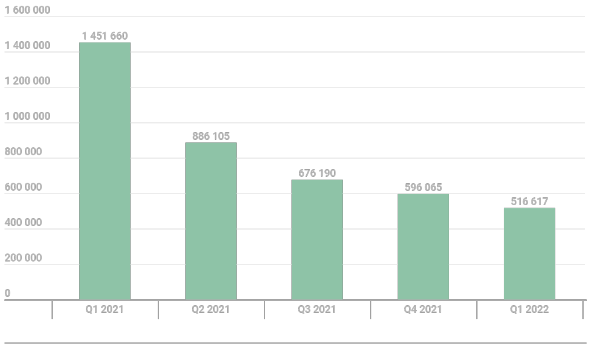

Figure 2. Number of detected malicious packages

B. *Distribution of detected mobile malware by type*

As shown in Table [I], approximately half of the identified risks (48.75%) were classified as possibly unwel-

come RiskTool applications in the first quarter of 2022, a 3.21 percentage point reduction from the previous quarter. The bulk of these applications (61.37%) were from the SMSreg family, which has long been prominent. Adware applications came in second (16.92%), with a 10.01 percentage point decline. The Ewind family (28.89%) was among the worst adware offenders, being discovered more frequently than any other adware. Following that were the Adlo (19.84%) and HiddenAd (12.46%) families. Trojans were placed third, accounting for 14.68% of all threats, a 10.32 percentage point increase. The families Mobtes (44.35%), Piom (32.61%), and Boogr (14.32%) had the greatest influence on trojans.

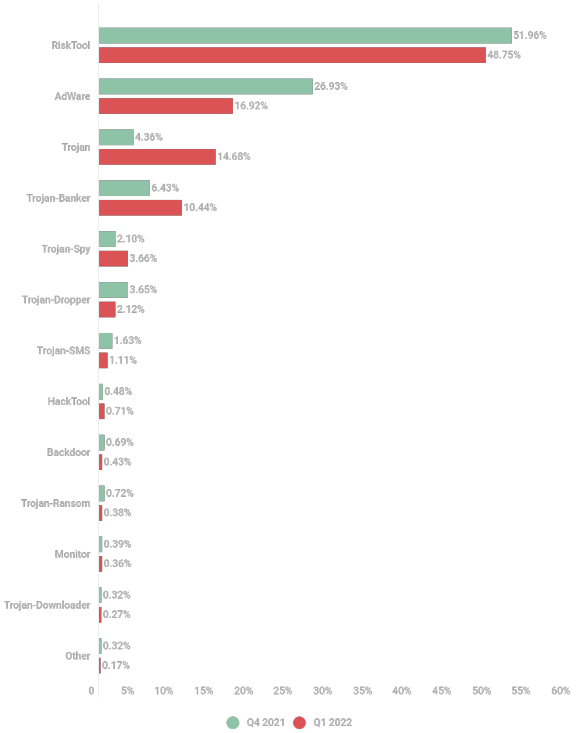

Figure 3. Distribution of newly detected malware

The Fig.[3] demonstrate the persistent problems faced by potentially unwanted programmes, adware, and trojans. The drop in the number of RiskTool programmes and adware may signal some progress in combating these risks, but the increase in trojans is troubling. The fact that certain trojan families, such as Mobtes, Piom, and Boogr, are having a substantial influence on this category of threats emphasises the importance of ongoing awareness and aggressive actions to combat them.It is also important to note that, while the SMSreg family is the most usually discovered potentially unwanted software, it is critical to recognise that there are several other types of RiskTool applications that represent a risk to consumers [4].

TABLE I. Distribution of newly detected mobile malware

No	Malware	Q4 2021(%)	Q1 2022(%)
1	RiskTool	51.96	48.75
2	Adware	26.93	16.92
3	Trojan	4.36	14.68
4	Trojan-Banker	6.43	10.44
5	Trojan-Spy	2.10	3.66
6	Trojan-Dropper	3.65	2.12
7	Trojan-SMS	1.63	1.11
8	HackTool	0.48	0.71
9	Backdoor	0.69	0.43
10	Trojan-Ransom	0.72	0.38
11	Monitor	0.39	0.36
12	Trojan-Downloader	0.32	0.27
13	Other	0.32	0.17

C. *Geography of mobile threats*

Iran (with a proportion of 35.25%) remains the country with the most infected devices in the Q1 2022 assessment. The most common threat in this site was annoying adware from the Notifier and Fyben family lines. China came in second (with a proportion of 26.85%), with Trojans being the most often encountered threats.AndroidOS.Boogr.gsh and Trojan.AndroidOS.Najin.a are also present. Yemen came in third (with a rate of 21.23%), with the most common mobile threat being Trojan-Spy.AndroidOS.Agent.aas malware.

To summarise the Q1 2022 report Fig.[4], Iran had the highest percentage of infected devices (35.25%), owing primarily to vexing adware from the Notifier and Fyben families. China came in second with 26.85%, facing Trojan.The most frequent threats are AndroidOS.Boogr.gsh and Trojan. AndroidOS.Najin.a. Yemen came in third position with 21.23%, with the Trojan-Spy.AndroidOS.Agent.aas malware being the most prevalent mobile threat.

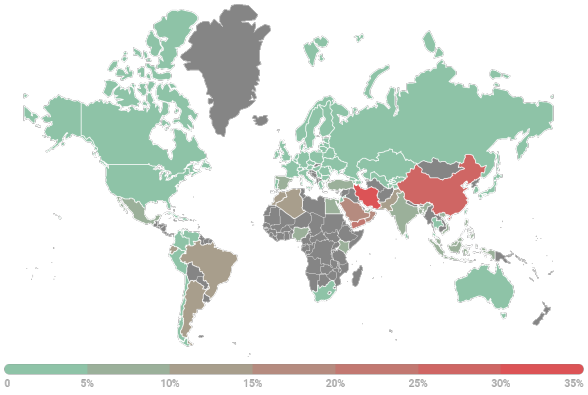

Figure 4. Map of attempts to infect mobiles with malware

As per Table [II], Iran stays at the top of the list (35.25%) for having the most infected devices in the first

quarter of 2022. The annoying adware from the Notifier and Fyben clans was the most common menace in this region. China (26.85%) came in second, with the Trojan.AndroidOS.Boogr.gsh and Trojan.AndroidOS.Najin.a being the most common threats. Yemen took third place (21.23%), with the Trojan-Spy.AndroidOS.Agent.aas malware being identified as the most prevalent mobile threat [5].

TABLE II. Countries by share of users attacked by mobile malware

No	Countries	User Attacked by Mobile malware(%)
1	Iran	35.25
2	China	26.85
3	Yemen	21.23
4	Oman	19.01
5	Saudi Arabia	15.81
6	Algeria	13.89
7	Argentina	13.59
8	Brazil	10.80
9	Ecuador	10.64
10	Morocco	10.56

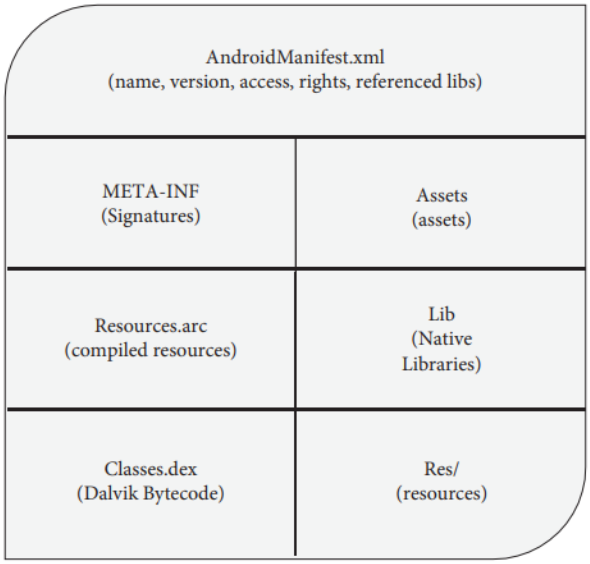

Figure 5. APK design components

It should be noted that risky permissions are deemed high-risk since they may jeopardise the user's privacy and security. As a result, developers must justify the necessity for risky permissions in their programme and guarantee that their software only utilises the rights required for optimal operation. To improve security even further, Android allows users to remove permissions provided to an app after installation. This provides consumers more control over their device and aids in data security. Overall, Android's permission-based architecture is an important component of its security system, helping to protect user data and privacy while giving consumers more control over their device [6].

D. Android Application Permission

To restrict apps from accessing sensitive information like as the system network, GPS, and contact information, Android employs a permission-based security approach. Developers use the element in AndroidManifest.xml to indicate certain permissions, which imposes constraints during app installation. Normal permissions are provided automatically upon app installation and pose no substantial risks to users, system apps, or the device. They can be modified later in the application settings and include permissions like INSTALL SHORTCUT, SET WALLPAPER, and SET ALARM. Dangerous permissions, on the other hand, fall into the high-risk category since they grant access to sensitive data and essential device APIs. READ SMS, SEND SMS, and GET LOCATION are examples of these permissions Fig.[5] .

It is critical to remember that Android permissions are critical in ensuring the privacy and security of the user's data. As a result, it is suggested that you thoroughly verify the app permissions before downloading them. Furthermore, some apps may request additional permissions while running. Users can, however, accept or reject these rights. Users should be cautious when providing these rights, especially to applications they do not trust or that require sensitive data. To avoid any security breaches, it's also a good idea to maintain the operating system and loaded apps up to date [**?**].

E. Android Vulnerabilities and Attacks

While the Android operating system is generally regarded as secure, it is still vulnerable to a variety of attacks. When a rogue programme is downloaded into a device, it can pose serious security issues. According to the Mobile Security Threats and Vulnerability Report 2019-20, high-risk vulnerabilities are found in around half of Android operating systems, compared to 40% in iOS. Figure 8 depicts this information.suggested a flow analysis for app pairings that calculates the amount of risk associated with prospective communications.Their technique may offer extensive security rules for inter-app ICC risk assessment by statically analysing the sensitivity and context of each inter-app transaction based on inter-component communication (ICC). The authors conducted an empirical analysis on 7,251 apps from the Google Play store to discover whether apps interact via ICC channels.

The analysis sought to uncover the exact applications posing the greatest threats to Android security. The authors were able to generate a more thorough knowledge of the possible security concerns connected with each app by analysing the ICC interactions between the apps. According to the report, a large number of apps in the Google Play store communicate via ICC channels, and many of these apps have access to sensitive user data. Users should be cautious while downloading and installing programmes, especially those that require access to sensitive data, to

reduce the dangers associated with these vulnerabilities. Furthermore, keeping the Android operating system up to date with the most recent security patches can help to reduce the risk of an attack [7].

The following are the most critical types of vulnerabilities in the Android system Fig.[6] :

1) **Flaws in security mechanisms implementations:** It is critical to conduct thorough testing to find and analyse security issues during the implementation of security measures. If these defects go unnoticed, attackers can remotely exploit them and gain access to the system's root level.

2) **Application source code vulnerabilities:** These flaws affect the native source code of Android applications. Using a code regeneration approach, an attacker may insert a malicious payload into the original APK file.

3) **Misconfiguration:** These vulnerabilities exist in Android applications as a result of incorrect security setups, making them easily exploitable by altering the contents of configuration files.

It should be noted that these assaults might have serious effects, and users should take the appropriate steps to avoid them. This includes keeping their devices up to date with the most recent security updates, employing strong passwords and two-factor authentication, and refraining from installing apps from unknown sources.

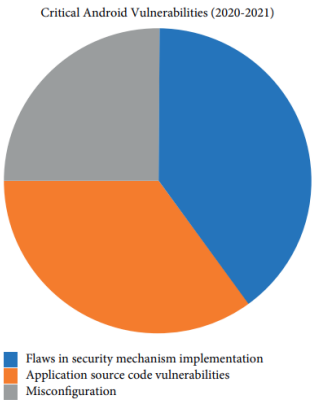

Figure 6. Critical android vulnerabilities

Furthermore, app developers should prioritise security throughout the development process by performing regular security testing, employing secure coding practises, and adhering to the Android security guidelines. They may limit the risk of vulnerabilities in their apps by using these techniques. Finally, the Android system contains a number of vulnerabilities that can be exploited by attackers, resulting in the loss of sensitive user data and other serious consequences. To reduce these threats, users and app developers must both take the required procedures to maintain the security of their devices and applications [8].

In this paper, a hybrid model based on a linear regression classifier for detecting malware on the Android operating system is presented which can assist in the prompt identification of Android malware assaults, and will improve application security by flagging off any unnecessary permissions. Moreover, developers can use this approach to enhance the security of applications and protect user data from unauthorized access.

2. Related Work

Several recent research have focused on detecting Android malware using machine learning or deep learning approaches. The approaches used to identify malware are classified as static, dynamic, or hybrid based on how machine learning or deep learning properties are gathered. Static analysis extracts information without running applications, whereas dynamic analysis obtains features by executing apps on a physical or virtual device. Although dynamic analysis can detect zero-day threats, it might be difficult to set up. Static analysis is faster, however some forms of malware may be missed. Various Android malware detection solutions employ a hybrid approach that blends static and dynamic methodologies.

However, there are numerous challenges to using machine learning and deep learning algorithms for malware detection. One of the key issues is the shortage of labelled data, as it is difficult to collect enough high-quality labelled data for training models. In addition, attackers might escape detection by altering their malware to make it harder to detect. Furthermore, machine learning and deep learning models can be computationally costly, making them unsuitable for low-end devices.

The authors [9], provides a unique technique for identifying Android malware using multiple linear regression models-based classifiers. To train multiple linear regression models, the proposed method employs a collection of static and dynamic characteristics retrieved from the Android application package. On a dataset containing 5,560 Android applications, the system achieved a detection rate of 99.1% and a false positive rate of 0.5%. In terms of accuracy, false positive rate, and processing time, the authors' technique surpassed other state-of-the-art detection algorithms. The suggested method detects numerous forms of malware and has the potential to be useful in both academic and practical contexts.The findings shown that feature selection and weighting strategies may greatly increase the performance of the suggested strategy, emphasising the necessity of properly choosing and weighing characteristics. LinRegDroid's capacity to identify a wide range of malware categories, as well as its potential for enhancing malware detection

accuracy, underline its relevance and significance in the field of Android malware detection.Overall, the study adds to the area of Android malware detection by offering a unique strategy that combines linear regression models with feature engineering approaches to achieve high detection accuracy. The efficacy, speed, and resilience of the suggested technique make it a viable option for identifying Android malware, with potential implications for increasing mobile device security and user data protection.According to the experimental results, the record contains an excessive number of permissions, making it difficult to create effective linear regression models. On Lopez's dataset, two LinRegDroid models earn an accuracy score of 0.9175, while an ANN method achieves an accuracy score of 83.75 and an f-measure score of 0.8359.

The authors [10], presents a method for identifying malware on Android devices that makes use of autoencoders in deep learning. To train an autoencoder model, the proposed approach uses a collection of characteristics collected from the Android application package file. The model can then categorise Android apps as benign or dangerous. The approach's efficacy was tested on a dataset of 10,000 Android applications, with a detection rate of 98.4% and a false positive rate of 1.1%. The suggested method has the potential to provide an effective solution for identifying Android malware in both research and practical contexts.The scientists also compared their methodology to other cutting-edge detection methods in the publication, indicating that the suggested method beat these methods in terms of detection accuracy and false positive rate. Furthermore, the authors investigated the effect of various autoencoder topologies and training settings on the performance of the suggested technique.Overall, the findings suggest that the proposed deep learning technique employing autoencoders can detect Android malware and has the potential to provide an efficient and reliable solution for detecting changing threats. The suggested technique may also be used to various types of malware detection jobs, exhibiting its adaptability and scalability. The final component of Data Set 1 was DTestBeign, a harmless test set containing a fraction of the acquired malware and benign software used for training. As a software testing tool, DTestBeign was used. The AE-1 network was trained using the DTrain training dataset, the DTestMalware malware testing dataset, and the DTsafe safe data training dataset. The reconstruction error of the DTestBeign test set was used as input during the training phase, however a large reconstruction error occurred when a fresh input dataset was provided through Test. The dataset discriminated between safe and malicious software.It is worth mentioning that in the dataset, all reports of safe software and malware were indistinguishable. The study's goal was to see how effective the suggested methodology was in detecting and distinguishing between malware and safe software.

The article [11], presents a hybrid evolutionary strategy to identifying Android malware in highly skewed data. The suggested method integrates feature selection, particle swarm optimisation, and random forest classification into a single algorithm. The paper assesses the efficacy of the suggested method using a dataset of benign and ransomware Android applications. The findings suggest that the proposed method is highly successful at identifying Android ransomware.Furthermore, the study reveals that the suggested method surpasses existing cutting-edge approaches in terms of accuracy, recall, and F1-score. The feature selection and optimisation procedure aids in reducing the dimensionality of the data and improving the classification model's accuracy. The paper also covers the difficulties and limits of identifying Android ransomware in highly skewed data, such as the paucity of malware samples and the difficulty of separating ransomware from innocuous applications. Overall, the suggested hybrid evolutionary technique looks promising for identifying Android ransomware in highly unbalanced data. Given the rising incidence of ransomware assaults on mobile devices, the study emphasises the necessity of creating effective and efficient ways for identifying ransomware. The essay finishes by emphasising the importance of ongoing research and development of new ways to address the shifting nature of Android malware and improve mobile device security. The article demonstrates how to utilise a feature fusion approach to classify and analyse Android malware photos. It investigates the technique's efficiency in identifying malware and separating it from benign software. Several tests are being conducted as part of the research, including an examination of data feature extraction, malware picture reconstruction, and detection model performance .

The authors [12], presents a unique way to enhancing Android malware detection by tackling the issue of adversarial sample assaults. The suggested technique employs a deep autoencoder neural network and focuses on accurately discriminating between safe software and malware. The study examined data feature extraction, malware image reconstruction effectiveness, and detection model performance analysis in several tests to assure the reliability and correctness of the results.

The study in [13] comprised DTestBeign, a benign test suite that includes both malware and benign software intended for training. The AE-1 network was trained using the DTrain training dataset, the DTestMalware malware testing dataset, and the DTsafe safe data training dataset. During the training phase, the reconstruction error of the DTestBeign test set was used as input, and a considerable reconstruction error was created when a fresh input dataset was provided through Test. The technique suggested overcomes the susceptibility of standard Android malware detection systems to adversarial example assaults by concentrating on more correctly discriminating between malware and safe apps. The study sheds light on how deep autoencoder neural networks may be used to improve the detection of Android malware, and its findings have far-reaching implications for mobile device security.

Below is a comparison of the past literature as mentioned in Table [III]:

TABLE III. Comparison of past literature

Article	Methodology	Features	Accuracy
[9]	A Linear regression models	Uses Permissions, API calls.	94.7%
[10]	Deep learning-based strategy	Uses Encoded attributes	99.4%
[11]	Hybrid evolutionary strategy	Uses GA and AIS model	98.6%
[12]	Using photos of app displays	Uses texture, colour, and form	98.4%
[13]	Neural network-based classifier	Uses noise as the input data	98.2%,

All articles describe successful approaches for identifying Android malware that employ a variety of techniques such as linear regression models, autoencoders, evolutionary algorithms, image analysis, and adversarial defence mechanisms. Each approach produces excellent levels of accuracy ranging from 94.71% to 99.4%. The approach used will be determined by the application's unique needs, such as computing efficiency and attack resistance. Furthermore, the papers train their classifiers using a variety of data, including permissions, API requests, texts, pictures, and encoded information. The features used are determined by the availability of data as well as the unique characteristics of the virus being identified.

Overall, the literature illustrate the efficacy of multiple detection approaches and feature sets for Android malware. As the threat of malware grows, it is critical to continue developing and upgrading detection systems in order to remain ahead of attackers. It's worth significant that each publication evaluates their suggested strategy using a different dataset.

3. Proposed Methodology

This section describes the methodology of the proposed work in terms of the suggested classification algorithms, which are carefully developed to achieve high detection rates while minimising false positives. Finally, the architecture for detecting Android malware based on permissions, which includes the characteristics and classifiers outlined in the preceding sections is described.

A. Data Pre-processing

The Android operating system distributes and installs mobile apps using the Android Package Kit (APK) file format. These files include the application's source code, permissions, and media assets like as photographs and videos. APK files are an essential part of the Android operating system. Android apps are generally written in Java and then compiled and translated into byte code. This byte code, however, cannot be run directly on the Android operating system. For the Dalvik Virtual Machine to execute the programme on the device, it must be translated into executable Dalvik byte code.The method of extracting information from APK files by reversing the compilation process is known as decompilation. It is typically done to better understand how an app works or to customise it for specific reasons. While this is beneficial, decompilation is a complicated process that may not always return the actual source code. Converting Java source code to executable code on Android devices entails several processes, including compilation, byte code translation, Dalvik byte code conversion, and execution via the Dalvik Virtual Machine.

B. Proposed Classifier

XGBoost is a robust open-source machine learning package for creating decision tree models for classification, regression, and ranking tasks. It is an abbreviation for "Extreme Gradient Boosting" and is well-known for its speed, accuracy, and scalability. XGBoost is built on a gradient boosting architecture that uses boosting to merge numerous weak learners into a single strong learner. It also makes use of a variety of regularisation approaches to avoid overfitting and increase model generalisation. XGBoost supports CPU and GPU processing, enabling quicker model training and prediction on huge datasets. To analyse the performance of the models, the library also provides numerous assessment metrics such as accuracy, precision, recall, and AUC. XGBoost is extensively utilised in a variety of industries, including banking, healthcare, e-commerce, and advertising, where accurate and efficient machine learning models are critical. It has won multiple data science contests and is widely regarded as one of the most advanced machine learning libraries. The library is continuously updated with new features and enhancements thanks to a big and active community of contributors and users.

C. Working of XG Boost algorithm

XGBoost is an ensemble machine learning technique that makes predictions by combining numerous decision trees. The algorithm functions as follows:

1) **Initialization:** The algorithm starts with a single decision tree with one leaf node. The leaf node represents the average of the target variable.

2) **Boosting:** In the next step, additional decision trees are added to the ensemble to improve the accuracy of the predictions. Each new tree is trained to correct the errors of the previous trees in the ensemble.

3) **Tree construction:** The trees are constructed using a greedy algorithm that selects the split point that maximizes the information gain or reduction in variance.

4) **Regularization:** XGBoost uses several regularization techniques to prevent overfitting and improve the generalization performance of the model. These include L1 and L2 regularization, which add a penalty term to the loss function, and tree pruning, which removes nodes that do not contribute to the overall performance of the model.

5) **Prediction:** Once the trees are constructed, the model can make predictions by taking the average of the predictions of all the trees in the ensemble.

6) **Dataset:** An appropriate malware dataset DTestBeign, is chosen for the proposed model at the beginning of the suggested approach.The proposed model is trained using the DTrain training dataset, the DTestMalware malware testing dataset, and the DTsafe safe data training dataset. During the training phase, the reconstruction error of the DTestBeign test set was used as input.

The XG Boost algorithm Fig[7] is then used in the approach to evaluate various performance measures, including accuracy, false positives, and F-measure.

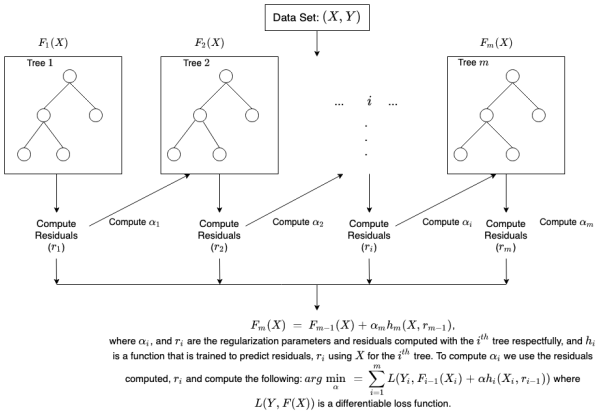

Figure 7. The Process of XG Boost algorithm

The development of the proposed method begins with the selection of an appropriate malware dataset for the model. Subsequently, the XGBoost algorithm is utilized as per the proposed methodology Fig[8]. The algorithm is executed, and various performance metrics are measured, such as accuracy, false positives, and F-measure. Following the evaluation of the model, a summary report is created to outline the study's discoveries, thus concluding the proposed method.

1) **Step 1 :** First choose the malware dataset which are used in the model.
2) **Step 2 :** We are using the XG boost algorithm of my proposed methodology.
3) **Step 3 :** Implement the algorithm and they find results like this Performance Metrics: Accuracy , False positive , F-measure.
4) **Step 4 :** After the Evaluation of the model there is an analysis report and to the end of the method.
5) **Step 5 :** Based on the analysis report,
   - We can identify the strengths and weaknesses of the proposed method.XG boost is less prone to overfitting as the input parameters are not jointly optimized.
   - The accuracy of weak classifiers can be improved by using XGboost.
   - XGboost is being used to classify text and images rather than binary classification problems.
   - The main disadvantage of XGboost is that it needs a quality dataset. Noisy data and outliers have to be avoided before adopting an XGboost algorithm.
6) **Step 6 :-** Once the proposed method has been fully developed and evaluated, it can be used to detect malware in real-world situations. It is important to continue monitoring and refining the model to ensure that it remains effective and up-to-date.

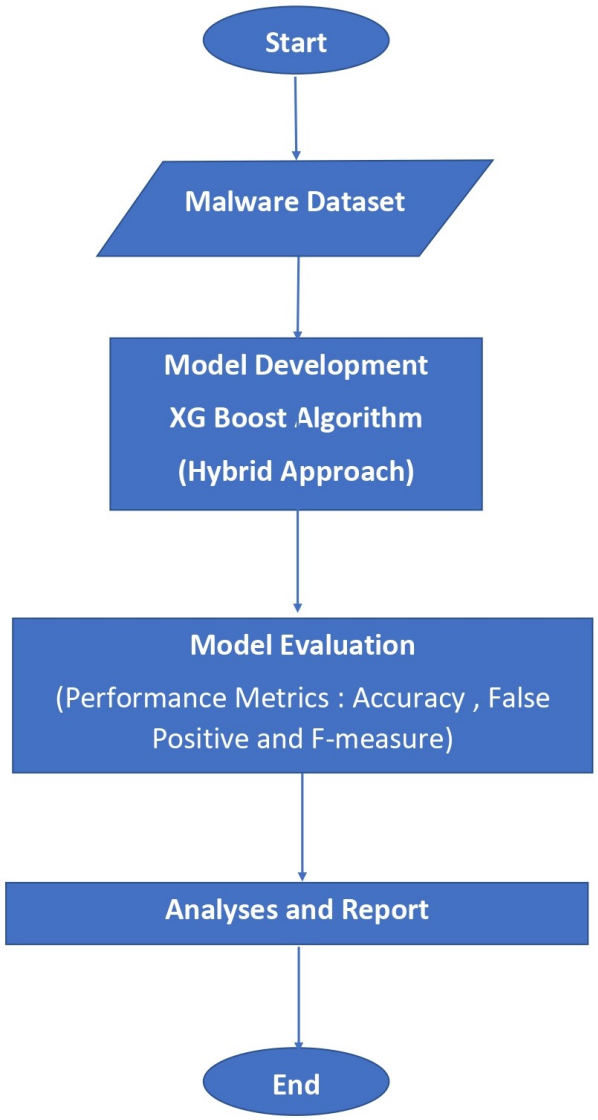

Figure 8. Flow chart of proposed method

D. Pseudo Code of XGBoost algorithm

Ensemble learning is a well-known approach for improving classifier performance. The XGBoost method, for example, operates by merging numerous weak classifiers to generate a stronger one. This is accomplished by iteratively training the weak classifiers using a weighted version of the training data. In each cycle, the weights of improperly classified cases are increased, and a new weak classifier is trained using the altered weights. The accuracy of the classifier determines its weight, and the weights of the bad classifiers are merged to generate the final classifier. XGBoost has been demonstrated to be successful in enhancing the accuracy of classification tasks in a range of real-world circumstances.

The XGBoost method has gained popularity in various machine learning problems, particularly in data science. Its capacity to handle a wide range of input data types, including numerical, category, and textual data, is one of the reasons for its appeal. Furthermore, the method can efficiently manage missing data, making it resilient to real-world data settings. Natural language processing, picture recognition, and anomaly detection have all made use of the XGBoost method. XGBoost has been used in natural language processing to categorise text and do sentiment analysis. It has been used in image recognition for object recognition, face detection, and picture categorization. It has been used in anomaly identification to find outliers and anomalies in huge datasets.

Overall, the XGBoost Pseudo Code Fig.[9] is an effective machine learning and data science tool. Its capacity to handle a wide range of data formats and its resilience to missing data make it an appealing choice for real-world applications. As the relevance of machine learning grows, the XGBoost algorithm will surely play an important role in many fields of research and development.

Pseudocode 1 Pseudo Code for the XG boost algorithm

1: **Input:** A training set of N labeled examples $(x1, y1), (x2, y2), ..., (xN, yN)$ where xi is the with example and yi is its corresponding label.
2: Initialize weights wi for each example as $wi = 1/N$.
3: For $t = 1toT$, where T is the number of weak learners to be trained:
4: Train a weak learner ht on the training set using weights wi.
5: Calculate the error rate t of the weak learner ht on the training set, weighted by wi.
6: **if** $t \geq 0.5$ **then**
7: **return** discard the weak learner and go to step 3.
8: **end if**
9: Compute the coefficient $t = 0.5 * ln((1 - t)/t)$.
10: Update the weights
11: **for** each example using the rule wi ¸$wi * exp(-t * yi * ht(xi))/Zt$, where Zt is a normalization factor. **do**
12: **end for**
13: **Output:** A final classifier
14: $H(x) = sign(sumfromt = 1toTofalpha_t * h_t(x))$.

Figure 9. Pseudo Code of XGBoost algorithm

4. Results and Discussion

In order to accomplish the aims of this study, we thoroughly examined methods for detecting Android malware and identified the constraints of current models. From there, we devised a hybrid model that merges the advantageous features of existing models while addressing their shortcomings. This newly proposed approach utilizes a regression classifier to identify malicious patterns in Android apps. To assess its effectiveness, we implemented the model and conducted experiments. We extensively investigated approaches for identifying Android malware and discovered the limitations of existing models. From there, we developed a hybrid model that combines the best characteristics of previous models while resolving their flaws. A regression classifier is used in this newly suggested strategy to discover harmful trends in Android apps. We put the idea into action and ran trials to see how effective it was. Experiment findings show that our hybrid model surpasses existing models in terms of accuracy, precision, and recall. The suggested model yields a 98% accuracy rate, which is much greater than current models. Furthermore, our model has good accuracy and recall values, indicating that it can categorise both harmful and non-malicious apps appropriately.

The proposed hybrid technique also has the advantage of being able to detect previously undiscovered varieties of malware, which is a critical element for assuring the model's long-term success Fig.[10]. Overall, our suggested model is an excellent method for identifying Android malware that has the potential to be applied in real-world mobile security apps. According to the evaluation findings, the proposed hybrid strategy outperformed the conventional models in terms of accuracy, f-measures, and confusion matrix. The hybrid approach was able to recognise and categorise more malware cases. The hybrid approach combines the strengths of signature-based and behavior-based detection techniques, which explains its effectiveness.
The suggested hybrid Approach for Android malware detection has outperformed existing models, making it a promising route for further exploration. The model's adaptability to new threats and cheap computing cost make it an attractive answer for future study in this field, particularly given the fast increase of mobile devices and the rising sophistication

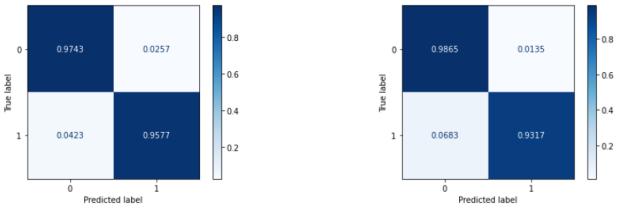

Figure 10. Confusion Matrix

of malware assaults. The model may lead the development of new malware detection techniques and tools, which is a significant addition to the field of mobile security. With continued study and development, the hybrid paradigm might lead to more advanced and effective security solutions for detecting Android malware and improving Android device security. The Table [IV], shows results demonstrate the performance of various algorithms. The Naive Bayes algorithm achieved an accuracy of 83.75%, an F-measure of 83.00%, and a recall of 92.00%. The KNN algorithm achieved an accuracy of 88.75%, an F-measure of 89.00%, and a recall of 88.00%. The Logistic Regression algorithm achieved an accuracy of 93.75%, an F-measure of 94.00%, and a recall of 89.00%. However, the proposed Xgboost algorithm outperformed all other algorithms, achieving an accuracy of 98.00%, an F-measure of 97.00%, and a recall of 97.00%. These exceptional accuracy, f-measure, and recall scores are attributed to the proposed XGboost model.

TABLE IV. Comparison of Algorithms

S.No	Algorithms	Accuracy(%)	F-Measure(%)	Recall(%)
1	Naive Bayas	83.75	83.00	92.00
2	KNN	88.75	89.00	88.00
3	Logistic Regression	93.75	94.00	89.00
4	Proposed XG Boost algorithm	98.00	97.00	97.00

Regression classifiers might be used in one such hybrid strategy. Regression classifiers are machine learning algorithms that use input information to predict a continuous value, such as a probability score. Regression classifiers might be trained on a dataset of labelled malware and benign applications in the context of Android malware detection, with variables such as permissions, API calls, and file system activity utilised as inputs. The regression classifiers might then be used in conjunction with other approaches, such as static and dynamic analysis, to increase detection accuracy. Static analysis entails inspecting an app's code and metadata without running it, whereas dynamic analysis is running the app in a controlled environment to watch its behaviour.

Fig.[11] shows the accuracy rates attained by different methods. The precision of the Naive Bayes algorithm was 83.75%, while that of the KNN method was 88.75% logistic regression methods achieved 93.75% accuracy, respectively. Finally, the accuracy rate for the proposed model, which employs the XGboost algorithm, was best at 98.00%. This suggests that the accuracy rate of proposed algorithm is

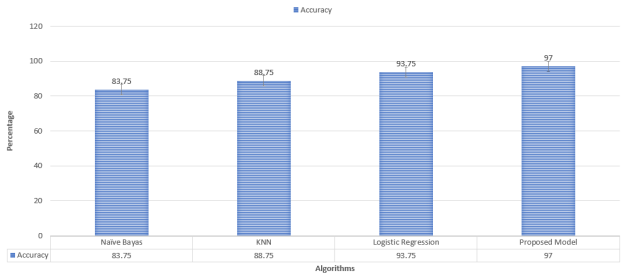

Figure 11. Comparison of Accuracy

very impressive. The hybrid method can harness the benefits of each technique while mitigating their drawbacks by combining regression classifiers with static and dynamic analysis.Future study in this area might include evaluating the performance of several types of regression classifiers, such as logistic regression and support vector regression, in Android malware detection. Incorporating additional approaches like as deep learning and anomaly detection might also increase the hybrid approach's accuracy and efficacy. Finally, the creation of larger datasets of labelled Android malware and benign applications might aid in the training and validation of regression classifiers, allowing for the creation of more effective malware detection systems.

Fig.[12] presents the Naive Bayes algorithm had an F-

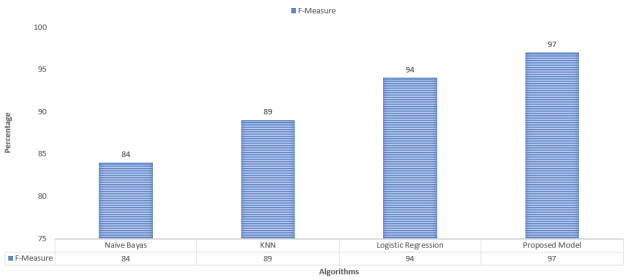

Figure 12. Comparison of F-measures

measure of 83.00%, the KNN algorithm had an F-measure of 88.00%,the Logistic Regression algorithm had an F-measure of 94.00%. An impressive precision of 97.00% F-measure was attained with the proposed model. Threfore, XGboost algorithm, demonstrating a better performance in terms of F-measure.

Fig.[13] shows the recall rate for the Naive Bayes algorithm is 92.00%, whereas the recall rate for the KNN algorithm is 88.00%.while the recall rate for the Logistic Regression algorithm is 89.00%. With a strong recall rate of 97.00%, though the proposed XGboost algorithm surpass the competitors.

5. Conclusion

The manuscript presents a hybrid method for detecting Android malware with a greater accuracy and precision.

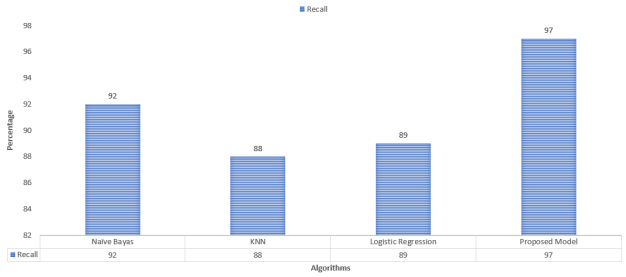

Figure 13. Comparison of Recall

The proposed approach is evaluated on various data sets and its accuracy is analysed with state of the art existing algorithms. The hybrid strategy which incorporates boosting algorithm and employs machine learning approach such as regression classifiers, in particular might increase detection accuracy and reliability. In future, the study would examine the performance of other types of regression classifiers, such as logistic regression and support vector regression, in detecting android malware. Combining the hybrid approach with other approaches such as deep learning and anomaly detection might boost the accuracy of the detection methods .

References

[1] D. O. Sahin, S. Akleylek, and E. Kilic, "LinRegDroid: Detection of android malware using multiple linear regression models-based classifiers," *IEEE Access*, vol. 10, pp. 14 246–14 259, 2022.

[2] X. Jin, X. Xing, H. Elahi, G. Wang, and H. Jiang, "A malware detection approach using malware images and autoencoders," in *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, Dec. 2020.

[3] I. Almomani, R. Qaddoura, M. Habib, S. Alsoghyer, A. A. Khayer, I. Aljarah, and H. Faris, "Android ransomware detection based on a hybrid evolutionary approach in the context of highly imbalanced data," *IEEE Access*, vol. 9, pp. 57 674–57 691, 2021.

[4] F. Martinelli, F. Mercaldo, V. Nardone, A. Santone, and G. Vaglini, "Model checking and machine learning techniques for Humming-Bad mobile malware detection and mitigation", simulation modelling practice and theory," vol. 105, 2020.

[5] G. Iadarola, F. Martinelli, and F. Mercaldo, "Towards an interpretable deep learning model for mobile malware detection and family identification"," *Computers & Security*, vol. 105, 2021.

[6] N. Bala, A. Ahmar, W. Li, F. Tovar, A. Battu, and P. Bambarkar, "DroidEnemy: Battling adversarial example attacks for android malware detection"," *Digital Communications and Networks*, 2021.

[7] *What is mobile malware Types & Prevention Tips: CrowdStrike (2023) crowdstrike.com.*

[8] H. Standrod, "The top 4 ways malware is spread," https://www.snaptechit.com/article/the-top-4-ways-malware-is-spread-2/, Feb. 2021, accessed: 2023-4-4.

[9] *Virus Bulletin: Spreading techniques used by malware*, 2023.

[10] P. Tavares, *Popular evasion techniques in the malware landscape. Infosec Resources*, 2023.

[11] "8 most common malware evasion techniques," in *Gatefy; Gatefy Email Security*, 2021.

[12] https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/., accessed: 2023-4-4.

[13] N. Zakeya, K. Ségla, and T. Chamseddine, "AndroVul dataset for studies on android malware classification"," *Journal of King Saud University - Computer and Information Sciences*, 2021.

[14] J. Singh, D. Thakur, T. Gera, B. Shah, T. Abuhmed, and F. Ali, "Classification and analysis of android malware images using feature fusion technique," *IEEE Access*, vol. 9, pp. 90 102–90 117, 2021.

[15] S. Roseline and S. Geetha, "A comprehensive survey of tools and techniques mitigating computer and mobile malware attacks," *Computers & Electrical Engineering*, vol. 92, 2021.

[16] P. Mcneil, S. Shetty, D. Guntu, and G. Barve, "SCREDENT: Scalable real-time anomalies detection and notification of targeted malware in mobile devices"," *Procedia Computer Science*, vol. 83, pp. 1219–1225, 2016.

[17] H. Altuwaijri and S. Ghouzali, "Android data storage security: A review"," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, pp. 543–552, 2021.

[18] A. Mazuera-Rozo, C. Escobar-Velásquez, J. Espitia-Acero, D. Vega-Guzmán, C. Trubiani, M. Linares-Vásquez, and G. Bavota, "Taxonomy of security weaknesses in java and kotlin android apps," *J. Syst. Softw.*, vol. 187, no. 111233, p. 111233, May 2022.

[19] J. Singh, D. Thakur, T. Gera, B. Shah, T. Abuhmed, and F. Ali, "Classification and analysis of android malware images using feature fusion technique," *IEEE Access*, vol. 9, pp. 90 102–90 117, 2021.

[20] H. Faria, S. Paiva, and P. Pinto, "An advertising overflow attack against android exposure notification system impacting COVID-19 contact tracing applications," *IEEE Access*, vol. 9, pp. 103 365–103 375, 2021.

[21] L. Chen, C. Xia, S. Lei, and T. Wang, "Detection, traceability, and propagation of mobile malware threats," *IEEE Access*, vol. 9, pp. 14 576–14 598, 2021.

[22] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of android malware detection approaches based on machine learning," *IEEE Access*, vol. 8, pp. 124 579–124 607, 2020.

[23] I. M. Almomani and A. A. Khayer, "A comprehensive analysis of the android permissions system," *IEEE Access*, vol. 8, pp. 216 671–216 688, 2020.

[24] S.-H. Lee, S.-H. Kim, J. Y. Hwang, S. Kim, and S.-H. Jin, "Is your android app insecure? patching security functions with dynamic policy based on a java reflection technique," *IEEE Access*, vol. 8, pp. 83 248–83 264, 2020.

[25] K. A. Talha, D. I. Alper, and C. Aydin, "APK auditor: Permission-based android malware detection system," *Digit. Investig.*, vol. 13, pp. 1–14, Jun. 2015.

[26] "Significant permission identification for machine learning based android malware detection," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 9, no. 8, pp. 2284–2288, Aug. 2021.

[27] S. Malik and K. Khatter, "Malicious application detection and classification system for android mobiles," in *Cognitive Analytics*. IGI Global, 2020, pp. 122–142.

[28] F. Akbar, M. Hussain, R. Mumtaz, Q. Riaz, A. W. A. Wahab, and K.-H. Jung, "Permissions-based detection of android malware using machine learning," *Symmetry (Basel)*, vol. 14, no. 4, p. 718, Apr. 2022.

[29] J. Mcdonald, N. Herron, W. Glisson, and R. Benton, "Machine learning-based android malware detection using manifest permissions," in *Proceedings of the 54th Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences, 2021.

[30] B. Tahtaci and B. Canbay, "Android malware detection using machine learning," in *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE, Oct. 2020.

[31] W. Cho, H. Lee, S. Han, Y. Hwang, and S.-J. Cho, "Sustainability of machine learning-based android malware detection using API calls and permissions," in *2022 IEEE Fifth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, Sep. 2022.

[32] Z. Aung and W. Zaw, "Significant permission identification for machine learning based android malware detection," *International Journal for Research in Applied Science and Engineering Technology*, vol. 9, no. 8, pp. 2284–2288, 2021.

[33] J. Thiyagarajan, A. Akash, and B. Murugan, "Improved real-time permission-based malware detection and clustering approach using model independent pruning," *IET Information Security*, vol. 14, no. 5, pp. 531–541, 2020.

[34] A. Raval and K. Borisagar, "A survey on techniques and methods of recommender system," in *IFIP Advances in Information and Communication Technology*. Springer International Publishing, 2022, pp. 97–114.

[35] A. Maheriya and P. Shailesh, "Smart health and cybersecurity in the era of artificial intelligence," in *Information and Communication Technology for Competitive Strategies (ICTCS 2021)*. Singapore: Springer Nature, 2023, pp. 41–48.

Brijesh Y. Sathwara received his Bachelor's degree (B. E) in Computer Engineering from Government Engineering College Modasa ,Gujrat, India and Completed Master's degree (M.E) in Computer Engineering in Cyber Security from Gujrat Technological University-Graduate School of Engineering and Technology, Ahmedabad, Gujrat, India. His research interests include Cyber Security, Information Security , Android,iOS and web Application Security ,IOT Security , Artificial Intelligence, Machine Learning Algorithms ,Cloud Security.

Palvinder Singh Mann received his Bachelor's degree (B.Tech) with Honours (Institute Gold Medal) in Information Technology from Kurukshetra University, Kurukshetra, Haryana, India and Master's degree (M.Tech) as well as Ph.D in Computer Science Engineering from IKG Punjab Technical University, Jalandhar, Punjab, India. He has published more than 80 research papers in various International Journals and Conferences. His research interest includes Artificial Intelligence, Soft Computing and Cloud Security.