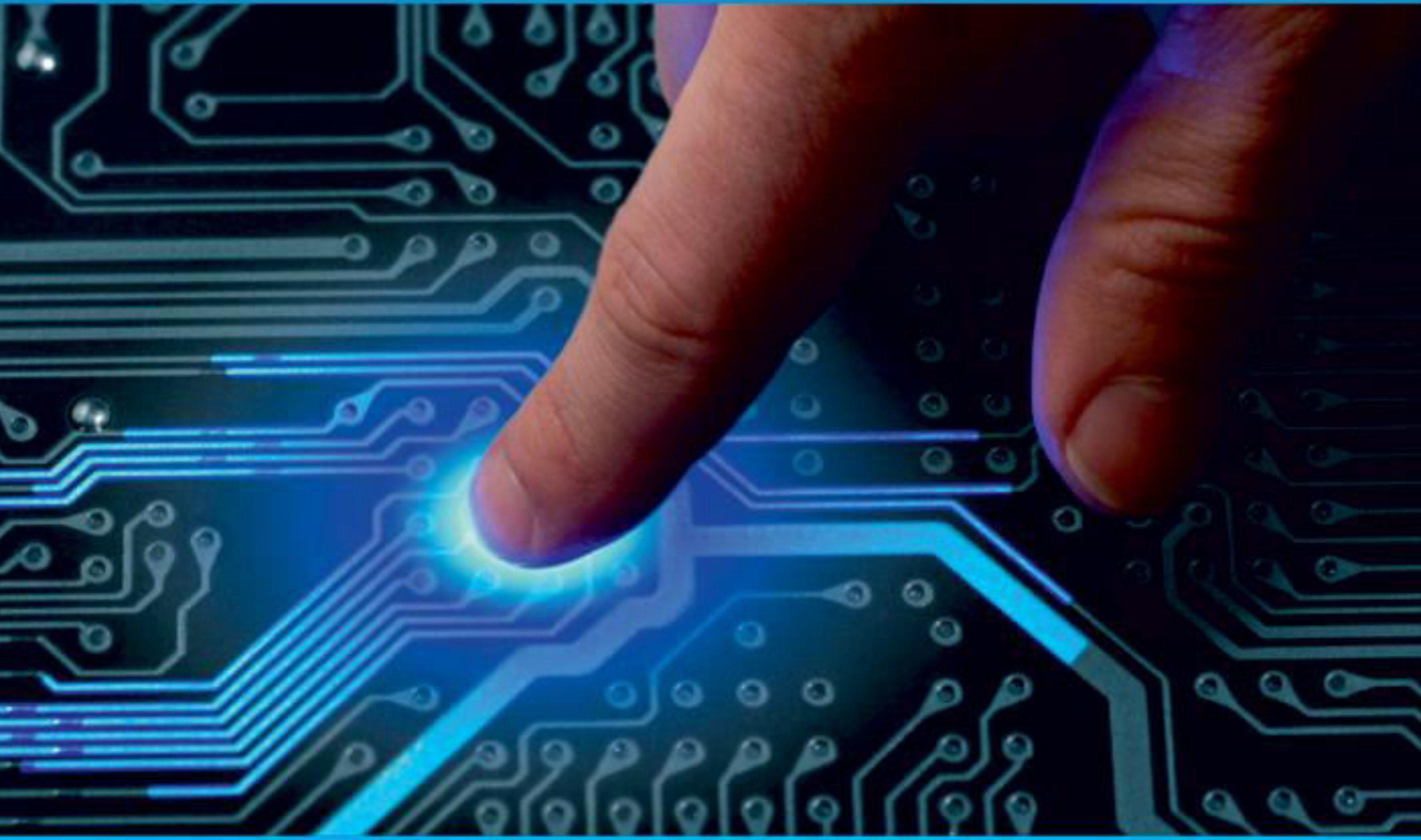
e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 4, April 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 4, April 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1204158 |

A Lightweight and Multi-Stage Approach for Android Malware Detection Using Non-Invasive Machine Learning Techniques

R.VIJAY, MATHAN KUMAR J, ARAVINTHAN S, KAVIN SHANKAR S

Assistant Professor, Department of CSE, Muthayammal Engineering College (Autonomous), Rasipuram, Tamil Nadu, India

Department of CSE, Muthayammal Engineering College (Autonomous), Rasipuram, Tamil Nadu, India

Department of CSE, Muthayammal Engineering College (Autonomous), Rasipuram, Tamil Nadu, India

Department of CSE, Muthayammal Engineering College (Autonomous), Rasipuram, Tamil Nadu, India

Department of CSE, Muthayammal Engineering College (Autonomous), Rasipuram, Tamil Nadu, India

ABSTRACT: Malware has long been employed in cyberattacks. Due to their widespread usage, malicious software developers target Android smartphones, which may store a lot of sensitive data. As the main mobile OS, Android has always attracted malware developers. Thus, several Android malware species target susceptible people everyday, making manual malware analysis unfeasible. ML and DL methods for malware identification and categorization might help cyber forensic investigators curb the spread of malicious software. Applying DL methods helps safeguard applications. Cybersecurity issues including intrusion detection, malware classification and identification, phishing and spam detection, and spam recognition have been addressed using DL approaches. ECNN uses the BP (Back Propagation) model for every layer between several intermediate layers, making it faster and more accurate than other methods. SVM Learning with Weighted Features and CNN with SGD optimization for static analysis of mobile apps are presented in this research. The ECNN model has the highest accuracy of 96.92, 96.14, and 95.8 for Android Malware Dataset-1, 2, and 3. On the three datasets, the ECNN model has 96%, 94%, and 94% precision. Smartphone malware analysis is faster and more accurate using this method.

KEYWORDS: android; malware; ranking; reduct; rough sets; prediction

I.INTRODUCTION

Smartphones, in reality, are the personal desktop computers of today's world. With smartphones, we can do almost anything we intend to do with personal desktop computers. Smartphones have become integral to modern society, impacting various aspects of our lives. Their versatility and functionality have revolutionized how we communicate, work, entertain ourselves, and access information. In addition to being a communication device to place calls and send SMSs, smartphones are used for internet browsing, social media, emails, photography and videography, navigation, online shopping, banking, health and fitness, education and learning, personal organization, and intelligent home control. The current mobile market share is 20% higher than desktops, meaning smartphones are leading the way ahead of desktops in terms of usage [1].

Among all the types of smartphones available in the market, smartphones with the Android operating system are the most popular. The reason for the popularity is that Android is an open-source platform many manufacturers adopt. As per the report of Statcounter [2], the global market share of mobile operating systems is entirely dominated by the Android operating system with a 70% share. The high market share of the Android operating system is one of the primary reasons for many malware attacks on the Android platform over the past few years. As per the report [3], several mobile trojan subscribers were found on Google's official app marketplace in 2022. As per the blog post by renowned antivirus firm McAfee [4], 60 Android apps with 100 million downloads were found to be spreading a new malware strain to unsuspecting users. According to the news article on TechRadar [5], a new ransomware, "Daam", capable of hiding from antivirus software, was detected. The report on the development of new Android malware shared by

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 4, April 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1204158 |

Statistica [6] shows that 5.6 million Android malware samples were seen in 2020. Moreover, millions of Android malware have been detected yearly from 2016 to 2020. Hence, there is a dire need to develop Android malware detection mechanisms to see malicious applications.

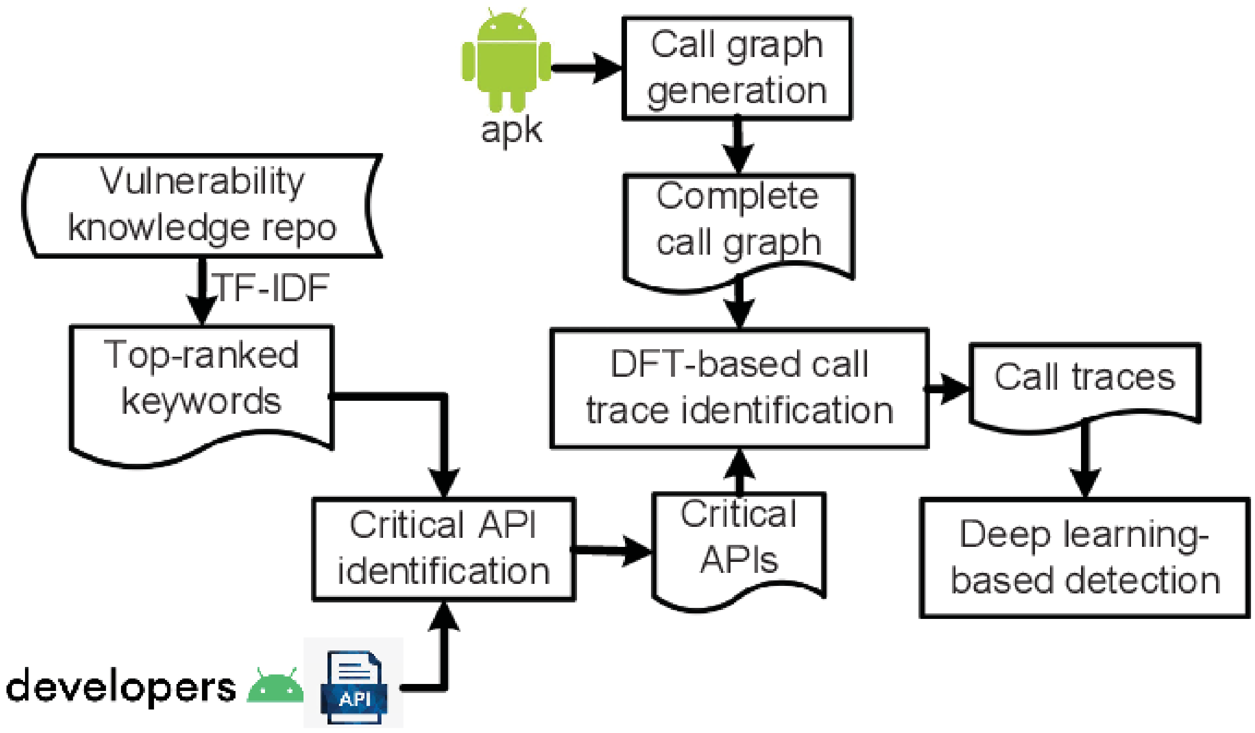

apk
Call graph generation
Complete call graph
Vulnerability knowledge repo
TF-IDF
Top-ranked keywords
DFT-based call trace identification
Call traces
Critical API identification
Critical APIs
Deep learning-based detection
developers
API

Fig 1: DeepCatra: Learning Flow

Malware analysis is a technique by which we can analyze the functionality and source of malware. Malware analysis can be broadly classified into three types [7]: static analysis, dynamic analysis, and hybrid analysis. These different analysis techniques can form a detection model to classify an Android application as malicious or benign. Three types of detection models could be possible: static, dynamic, and hybrid. In static detection, features are extracted using static analysis, which is the art of malware analysis that extracts features from the application without installing or executing the application. In dynamic detection, features are extracted using dynamic analysis, performed by executing or running the application and capturing the features at run time. The hybrid detection model is based upon hybrid analysis, which extracts both static and dynamic features from the application.

In static detection methods, the standard static techniques employed for extracting static features are manifest file-based detection, API calls-based detection, and Java code-based detection methods. In the manifest file-based detection technique, features are extracted from the manifest file of the Android application. Li et al. [8] used permissions from the manifest file and achieved an accuracy of 90%. Arora et al. [9] used permissions pairs from the manifest file and achieved an overall accuracy of 95.44%. IPDroid [10] used both permissions and intents from the manifest file and achieved an accuracy of 94.73% with a Random Forest classifier.

API calls-based feature detection is based on extracting APIs to be called by Android applications. Droidmat [11] used API calls along with manifest file components to detect malicious applications with an accuracy of 97.87%. Elish et al. [12] extracted sensitive API calls invocation by the user to build a detection model. Zhang et al. [13] created association rules between API calls and achieved an accuracy of 96%.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 4, April 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1204158 |

II.RELATED WORK

This section describes the work on static malware detection methods based on manifest file-based detection, API calls-based detection, and Java code-based detection. Therefore, this section is divided into three subsections: manifest file-based detection, API calls-based detection, and Java code-based detection.

In this sub-section, we review the works that have analyzed features from manifest files for Android malware detection. Grace et al. [17] established the relationship between embedded ad libraries and host apps as a significant risk factor for Android-based smart devices. Enck et al. [18] developed a lightweight certification model based on security configuration within an application to identify risky applications. A malware detection system, SIGPID, was proposed by Li et al. [8], in which three-level pruning is applied to filter out a minimal permission set that can be used for malicious app identification. Talha et al. [19] developed a client–server-based tool known as APK-auditor to detect malicious applications that maintain the Android profile database based on permission analysis.

The authors in [20] developed context category ontology based on permissions to detect the potential risk of information leakage with the help of a malicious activity. A prototype ASE was developed by Song et al. [21], which applied four levels of filtering based on static analysis to detect an application as benign or malicious. DroidChain [22] is another malware detection approach that applies static analysis and a behavior chain model to detect four types of malicious behaviors, i.e., privacy leakage, SMS financial charges, malware installation, and privilege escalation. ProDroid [23] is another behavior-based malware detection model that uses the biological sequence technique and Markov chain model to match the classes and API of decompiled applications to the stored malicious behavior models. Moonsamy et al. [24] used both requested and used permissions to mine contrasting permission sets, which were further used to detect an application as benign or malware.
used intent filters and permissions to identify an application as benign and malicious. work is based on rankings of requested permissions based on risk, and the selection of risky permission subset is made to train machine learning models. DroidRanger was developed by the authors in as a malicious application detection tool using permission behavior and heuristic filtering, which also successfully discovered zero-day malware did a unique work of annotating the capabilities of detected malware regarding security and privacy concerns. APIs, intents, and permissions were used in to perform similarity associations with malware samples and detect malicious applications using hamming distance.
tried to develop a fast malware detection model using multiple features such as permissions and opcode sequences. is a lightweight malware detection technique that can be deployed on the smartphone. The developed method can detect malicious applications on the smartphone within ten seconds of download. work is based on selecting prominent features from various sets of static features. sed ten different feature selection techniques to find the best possible combination of features to detect malicious applications effectively. is a genetic search-based technique, in which a genetic algorithm minimizes the number of features.

III.METHODS

First, each of the features mentioned above are considered individually to be pre-processed. The feature correlation score is calculated for each of the separate features, i.e., permissions, API calls, system commands, and opcodes. This feature correlation score is the basis for eliminating such features that are highly correlated with the existing features. The elimination of closely interconnected attributes occurs due to their strong correlation, resulting in a high degree of linear dependence. As a result, they exert nearly identical influences on the dependent variable. Thus, in cases where two attributes exhibit significant correlation, it is feasible to omit one of them. A correlation score of 90 per cent or more is used to select one feature out of two and eliminate the other. After this round, we received 4428 permissions out of 5501 permissions, 1589 API calls out of 2128 API calls, 93 System commands out of 103 System calls, and 159 Opcodes out of 224 Opcodes.

The formula for the Chi-square test involves several key terms and calculations, as shown in Equation is the expected frequency of category *i* under the null hypothesis. The sum is taken over all categories in the contingency table.Initially, the null hypothesis assumes that the two variables, i.e., the feature variable and the class variable, do not have any association. Then, the Chi-square test is applied to those variables using Equation (1) to calculate the Chi-square test statistic value. The Chi-square distribution table maps the estimated chi-square test statistic value to the corresponding *p*-value. The *p*-value less than 0.05 means that the null hypothesis assumed is false, and the alternate hypothesis that the class variable and the feature variable are associated with each other is true. All those features in each feature set that fall in the rejection region whose *p*-value is less than 0.05 are selected as a new feature subset for each

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 4, April 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1204158 |

feature set. Choosing a *p*-value of less than 0.05 means the features falling in this rejection region have 95% confidence of dependency on the class variable. Hence, in the end, we got a subset of each feature space with the minimized feature set from each feature space, i.e., permissions, API calls, system commands, and opcodes' feature space.

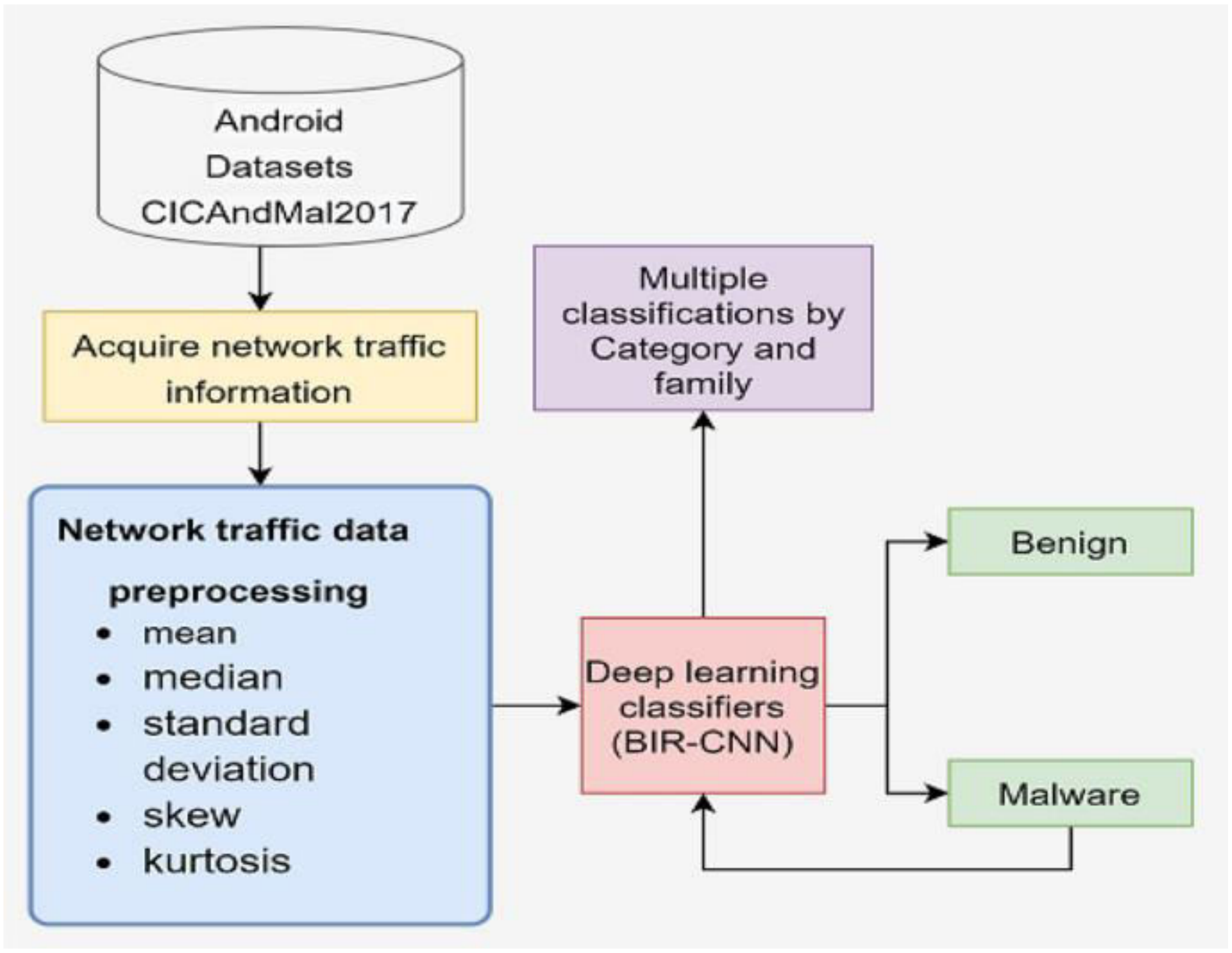

Android Datasets CICAndMal2017
Acquire network traffic information
Network traffic data preprocessing
mean
median
standard deviation
skew
kurtosis
Multiple classifications by Category and family
Deep learning classifiers (BIR-CNN)
Benign
Malware

Fig 2: Convolution Neural Network

Fourthly, we can combine two types of feature sets out of permissions, opcodes, API calls, and system commands in six ways. We experimented on all these six combinations and observed that opcodes and API calls combinations give the highest accuracy among all the six types of combinations. The reason seems to be obvious from the fact that opcodes constitute low-level instructions executed by a processor. In the realm of Android malware detection, scrutinizing the sequence of opcodes within an app's code can unveil discernible patterns suggestive of malicious behavior. Simultaneously, examining the app's API calls offers insights into its conduct. Specific API calls may serve as markers for malicious activities, and analyzing their patterns can assist in identifying malware. Consequently, the amalgamation of opcodes and API examines a more detailed perspective on an app's behavior at the code level. This approach proves advantageous for pinpointing nuanced and sophisticated malware that may employ obfuscation techniques. In contrast, permissions offer a higher-level overview based on declared capabilities, potentially offering less specificity.

Similarly, we combined all these four features in the group of three. The total number of combinations possible are four. Experimenting with all these four combination establishes that combining permissions, opcodes, and API calls is best among them. The reason seem to be that permissions offer insights into an app's officially declared capabilities. Integrating permissions with opcodes and API calls enhances our comprehension of an app's intended functionality and the possibility of misuse. Specific combinations of permissions can act as early indicators of potential issues, even before delving into the details of code execution. While opcodes and API calls provide a granular view of an app's behavior at the code level, incorporating permissions into the analysis contributes to a broader context, fostering a more comprehensive understanding of an app's intentions and the associated risks.

III.RESULT ANALYSIS

the results of four sets of permissions, i.e., all permissions in the data set, reduced permissions obtained after applying correlation feature elimination as permission correlation, permission chi as a set of permissions obtained after applying chi-square test on permission correlation, and finally permission reduct as the reduced permission feature set obtained after applying rough set reduct on permission chi. All four permission sets are used to train all six classifiers, i.e., Support Vector Machines (SVM), K-nearest neighbours, Random Forest, Logistic regression, ANN, and CNN. The results indicate that, for each type of classifier, as the feature set is changed from all permissions to permission correlation then to permission chi and finally to permission reduct, the accuracy increases. This means that reduct feature sets are the best for building malware detection systems.

In this subsection, we summarize the reasoning behind the detection results achieved from the proposed model. The proposed model gives us the highest accuracy of 97% with all four features. We observe that the detection accuracy significantly improves when we combine all four types of features for detection. The reason behind this observation is that different categories of features capture different behavioral aspect of malicious application. Hence, combining different types of features gives us wholesome characteristics of a malicious application under consideration. Therefore, increasing the number of features in combinations helps to improve the overall accuracy.

Secondly, we observed that API calls give us better results as compared to other features individually. Malicious software developers might try to avoid being detected by obscuring their code. Nevertheless, concealing the need for specific API calls is challenging. Identifying these calls can aid in the identification of malware that is intentionally concealed. Hence, results with API calls are better when compared to other features.

Thirdly, Random Forest gives us the better results when compared to other classifiers. The reason lies in the fact that Random Forest is a technique that employs a collection of decision trees to formulate predictions. This method of ensembling is beneficial for mitigating the variance and overfitting risks inherent in individual decision trees. Through the consolidation of predictions from numerous trees, there is a consistent enhancement in overall performance.

IV.CONCLUSIONS

In this work, we have proposed a novel Android malware detection model based on rough set theory. We have considered the combination of four static features, namely, permissions, opcodes, API calls, and system commands. Initially, a data pre-processing stage was executed, during which correlated features, as well as features that exhibited no dependence on the class variable, were eliminated. A ranking score was computed to establish the significance of each feature by employing the concept of Discernibility Matrices from rough set theory. Subsequently, an algorithm for computing rough set reducts was employed to reduce the number of features within each category based on the ranking score and the Discernibility Matrix concept of rough set theory. Following this reduction step, machine learning algorithms were applied to assess the detection accuracy using the calculated reducts. To gauge the effectiveness of the proposed model, a comparison was drawn against other advanced detection techniques. The results of this comparison underscored the superior performance of the proposed model over other state-of-the-art models in the field.

In our future work, we will try to work on the limitations of static analysis, i.e, we will be using dynamic analysis techniques along with static analysis techniques to build hybrid malware detection methods. Currently, the proposed model is an off-device model, and hence it can not be installed on smartphones for real-time detection. In future, we will propose a client–server-based model to integrate in smartphones.

REFERENCES

1. Petrov, C. 51 Mobile vs. Desktop Usage Statistics for 2023; Technical Report. Available online: **https://techjury.net/blog/mobile-vs-desktop-usage/** (accessed on 27 July 2023).
2. Statcounter: Mobile Operating System Market Share Worldwide. 2023. Statcounter. Available online: **https://gs.statcounter.com/os-market-share/mobile/worldwide** (accessed on 27 July 2023).
3. SHISHKOVA, T. The Mobile Malware Threat Landscape in 2022. Technical Report. SECURELIST by Kaspersky (February 2023). Available online: **https://securelist.com/mobile-threat-report-2022/108844/** (accessed on 27 July 2023).

4. McAfee. Goldoson: Privacy-Invasive and Clicker Android Adware Found in Popular Apps in South Korea. 2023. McAfee. Available online: **https://www.mcafee.com/blogs/other-blogs/mcafee-labs/goldoson-privacy-invasive-and-clicker-android-adware-found-in-popular-apps-in-south-korea/?AID=11552066&PID=9129747&SID=tomsguide-in-8524612056782596000** (accessed on 27 July 2023).
5. Fadilpašić, S. This Dangerous New Malware Also Has Ransomware Capabilities. Techradar. 2023. Available online: **https://www.techradar.com/news/this-dangerous-new-malware-also-has-ransomware-capabilities** (accessed on 27 July 2023).
6. Petrosyan, A. Development of Android Malware Worldwide 2016–2020. Statista. 2020. Available online: **https://www.statista.com/statistics/680705/global-android-malware-volume/** (accessed on 27 July 2023).
7. Tam, K.; Feizollah, A.; Anuar, N.B.; Salleh, R.; Cavallaro, L. The evolution of android malware and android analysis techniques. *ACM Comput. Surv. (CSUR)* **2017**, *49*, 1–41. [**Google Scholar**] [**CrossRef**]
8. Li, J.; Sun, L.; Yan, Q.; Li, Z.; Srisa-An, W.; Ye, H. Significant permission identification for machine-learning-based android malware detection. *IEEE Trans. Ind. Inf.* **2018**, *14*, 3216–3225. [**Google Scholar**] [**CrossRef**]
9. Arora, A.; Peddoju, S.K.; Conti, M. Permpair: Android malware detection using permission pairs. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 1968–1982. [**Google Scholar**] [**CrossRef**]
10. Khariwal, K.; Singh, J.; Arora, A. Ipdroid: Android malware detection using intents and permissions. In Proceedings of the 2020 Fourth World Conference on Smart Trends in Systems, Security, and Sustainability (WorldS4), London, UK, 27–28 July 2020; pp. 197–202. [**Google Scholar**]
11. Wu, D.-J.; Mao, C.-H.; Wei, T.-E.; Lee, H.-M.; Wu, K.-P. Droidmat: Android malware detection through manifest and api calls tracing. In Proceedings of the 2012 Seventh Asia Joint Conference on Information Security, Tokyo, Japan, 9–10 August 2012; pp. 62–69. [**Google Scholar**]
12. Elish, K.O.; Shu, X.; Yao, D.D.; Ryder, B.G.; Jiang, X. Profiling user-trigger dependence for android malware detection. *Comput. Secur.* **2015**, *49*, 255–273. [**Google Scholar**] [**CrossRef**]
13. Zhang, H.; Luo, S.; Zhang, Y.; Pan, L. An efficient android malware detection system based on method-level behavioral semantic analysis. *IEEE Access* **2019**, *7*, 69246–69256. [**Google Scholar**] [**CrossRef**]

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING