iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** I **Month of publication:** January 2024

DOI: https://doi.org/10.22214/ijraset.2024.57885

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue I Jan 2024- Available at www.ijraset.com

Detection and Analysis of Android Ransomware Using the Support Vector Machines

Hardlife Ngirande[1], Martin Muduva[2], Ronald Chiwariro[3], Austin Makate[4]
[1, 2]*Zimbabwe National Defence University, Zimbabwe*
[3]*Jain University, India*
[4]*University of Zimbabwe, Zimbabwe*

Abstract: ***Ransomware is a malicious code designed to encrypt and lock personal data such as documents and photos, in order to create opportunity for extorting money from the victims. Android operating systems are particularly targeted due to their large market share. Previous studies have primarily relied on signature-based detection methods, which require sufficient data samples and labelled signatures. However, modern ransomware utilizes obfuscation techniques that make it challenging to analyse using static methods. This project proposes a hybrid analysis approach for Android ransomware, employing the SVM algorithm for detection. The novelty lies in the limited exploration of SVM algorithms for ransomware analysis. The dataset used in the study was obtained from CICA and Mal 2017. Static features, including permissions, intents, encoding methods, and API calls were used, along with dynamic features such as network activities and system calls. The SVM model achieved good performance, with 81% accuracy and 90% precision using static features, and 100% accuracy with dynamic features.***
Keywords: API calls, Ransomware, SVM Algorithm, Package name, Static and Dynamic Analysis.

I. INTRODUCTION

Android ransomware has become a prevalent and significant threat in recent years, posing serious risks to smartphone users and their personal data. Ransomware is a malicious code that infiltrates devices and encrypts essential files or locks the entire device, demanding a ransom from the victim in exchange for restoring access to their data. With the widespread adoption of Android operating systems and the increasing reliance on smartphones for various tasks, cybercriminals have recognized the lucrative potential of targeting Android devices.

The alarming rise in Android ransomware incidents has highlighted the need for efficient and effective incident response processes to mitigate the impact of such attacks. Traditional methods of incident response, such as signature-based detection mechanisms, have shown limitations in detecting modern ransomware variants that employ sophisticated obfuscation techniques [1]. These techniques make it challenging to rely solely on static analysis approaches for accurate detection and analysis of Android ransomware. To address these challenges and optimize the response process to Android ransomware incidents, researchers and cybersecurity professionals have turned to machine learning techniques, specifically the Support Vector Machine (SVM) algorithm. SVM is a supervised machine learning approach known for its ability to handle complex problems in both linear and nonlinear applications [2]. Through leveraging SVM and its capability to classify and analyse data, the study aims to develop a model that can effectively detect and analyse Android ransomware.

The prevalence of Android ransomware poses a significant threat to individuals, organizations, and the overall cybersecurity landscape. Reports indicate that the average ransom payment has been steadily increasing, demonstrating the financial impact of these attacks [3], [4]. As the average ransomware payment nearly doubled in the period between 2022 and 2023, it is evident that conventional detection methods are insufficient to counter this growing menace.

Machine learning techniques, such as SVM, offer promising solutions by leveraging the power of data analysis and pattern recognition. By training the SVM model on a dataset comprising static features (such as permissions, intents, encoding methods, and API calls) and dynamic features (including network activities and system calls), the study aimed to enhance the accuracy and efficiency of Android ransomware detection [2]. The utilization of both static and dynamic attributes enables a comprehensive analysis of ransomware behaviour, providing a more robust defence against evolving ransomware variants.

Optimizing the response process to Android ransomware incidents through machine learning techniques is crucial in minimizing the impact of cyberattacks and ensuring the security of personal and sensitive data. By developing a model that can accurately detect and analyse Android ransomware, cybersecurity professionals can respond swiftly and effectively, thereby mitigating the financial and emotional repercussions experienced by victims.

II. LITERATURE REVIEW

Prevailing ransomware detection techniques used by anti-viruses are signature-based and behaviour-based to detect and defend devices against malware [5], [6]. The former detects by searching for a pattern of bytecode that matches the patterns stored in the database. This technique may positively detect previously known malware, but it cannot recognize new unseen malicious behaviours. The behaviour-based techniques do not require a predefined knowledge of malware, a set of rules are defined and any program violating the ruleset is considered malicious. These techniques are based on static and dynamic approaches to malware analysis.

Static analysis decompiles the APK file and examines the code together with any associated information. It extracts multiple attributes from the Java code and manifest file [7]. Code errors, signatures, permissions, API calls excessive permission, hardcoded credentials, weak cryptographic functions, workflow bypass and hidden features are some of these features. Static analysis features detect unknown and new malicious samples while signature-based anti-malware relies on malware behaviours [8].

The debugger, disassembler, and program analyser are used to perform static analysis. Tools used are Apktool, Jadx-GUI and Dex2jar. A command-line tool called Apktool is used to decompile APKs into resource folders, Dalvik Executable (dex) files, and AndroidManifest.xml files [8], [9]. These files can be examined to retrieve Java methods and permissions that criminals have taken advantage of. The Jadx-GUI tool is a graphical user interface tool for Java decompiler that extracts Java archive files and parses the Java source code to extract malicious codes that lock and encrypt files. The Dex2jar tool is used to create Java archive files from Dex files by converting the Dalvik bytecode into Java bytecode.

Dynamic analysis is carried out to set off the application's behaviour. The dynamic analysis malware detection includes network traffic, memory and registry utilization, instruction traces, and API call traces. CrowDroid, TaintDroid, ParanoidAndroid, Aurasium, DroidBox, and DriodScope are the tools that are for dynamic analysis [10], [11]. Android emulators, debuggers, and network analysers, are needed to observe file actions and interactions. Android Studio and Genymotion are the Android emulators used the simulate real phones [12]. The emulator instance is connected using ADB Android Debug Bridge. The DroidBox comes with Monkey Runner, which is used to emulate random UI interactions. Then record the resulting system calls using the monitoring tool strace. The strace tool monitors and tampers with interactions between processes and the Linux kernel, which include system calls, signal deliveries, and changes in process state.

Support Vector Machines (SVM) solves complex problems in linear and nonlinear applications. It works by finding a hyperplane that separates the two classes. It is a useful method for resolving issues with classification and regression [2], [13]. It makes use of the kernel, the kernel function receives input data and transforms it into the required format. Types of kernel functions are linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid [14], [15]. It divides feature space into the hyperplane to maximize the classification margin.

Several studies have focused on classifying malicious behaviour of Android applications using static analysis. Gupta [16] classifies and characterizes malicious behaviour of Android applications using static features. The static features used were data flow, permissions, components and inter-component communication along with unique source-sink pairs obtained from data flow analysis. In a similar type of study, [17] analysed 24 dangerous permissions detected by Android applications from the different malware samples in the Drebin 2014 dataset. The findings confirm that the applications' behaviour has been classified to their respective malware family. However, they did not offer a thorough overview of the detection and classification of a large number of Android malware belonging to varied malware families.

Some researchers [18], and [19], have proposed the dynamic analysis of Android malware detections using network attributes. They examined malicious traffic encrypted on host computers to come up with traffic fingerprints. The experiment results showed an accuracy of 95.2% in categorizing malware. A significant amount of research was on permissions and API package calls. The researchers like Almomani and Khayer; Alsoghyer and Almomani [20], [21] proposed a permissions-based ransomware detection system, based on the outcome of this analysis based on the Random forest (RF), Decision trees (J48), and Naıve Bayes (NB) algorithms. After the evaluation of the ransomware detection service, the results revealed a high detection rate that reached 96.9%. The model was good however, other types of ransomware attributes are to be analysed with permissions.

Sharma et al., [22] use machine-learning models to distinguish between benign and malicious applications by comparing them to determine how long it will take the algorithms to detect Android ransomware. With an accuracy of 99.59% using Logistic Regression, the researcher framework identified both lockers and crypto-ransomware in 177 milliseconds on the Graphics Processing Unit (GPU) and 235 milliseconds on the Central Processing Unit (CPU), respectively. He then proposed a framework to classify Android ransomware and benign apps by using supervised machine learning models from static features.

The attributes used were data flows, permissions, system calls and network traffic. Permission and API calls have been widely analysed as compared to other attributes like network traffic. There is a need to combine attributes for static and dynamic analysis.
Android forensics and detection is an emerging research domain where topics of publications are concentrated on the malware family in general rather than ransomware. Several studies have explored the effectiveness of machine learning techniques in detecting Android malware and ransomware. Hyoil Han et al. (2020) conducted a static analysis of Android applications' API calls and achieved an impressive SVM accuracy of 99.75% and a recall rate of 99.97%. Mahdi Moodi et al. (2020) developed the SAPSO-SVM technique for Android botnet detection, achieving a high accuracy of 98.3% by extracting 85 features from Android phone traffic. Santosh Jhansi K. (2020) focused on identifying influential permissions and achieved a randomisable filtered classifier accuracy of 94.47% using 330 permissions. Ahmed S. Shatnaw et al. (2020) presented a static base classification approach based on Android permissions and API calls, achieving an 88.75% F1 rate with SVM. Talal A.A (2020) proposed a hybrid analysis approach utilising machine learning to detect unknown and zero-day Android malware apps, achieving an impressive accuracy of 0.9829 with the SVM model. Huijuan Zhu and Yang Li (2020) utilized static analysis to detect Android malware and achieved an SVM accuracy of 92.47% with a precision of 95.15%. Ruei-Hau Hsu (2020) implemented privacy-preserving federated learning for Android malware detection. The study synthesized different research works on Android ransomware analysis and detection methodology and algorithms used by different researchers.

III. METHODOLOGY

The proposed framework for enhancing Android ransomware incident response is a two-phased architecture. It consists of forensic analysis and detection of ransomware using the SVM algorithm.

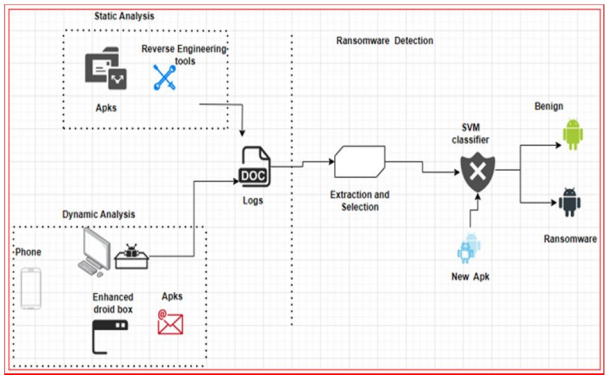


Figure 1: Proposed framework for analysis and detection of APKs

AndroPyTool is an Android dynamic analysis tool that is a modified version of DroidBox that comes with the Strace. The tool also used is for static analysis. AndroPyTool handles many APKs at once. The second is the machine learning part, where feature extraction and selection techniques are applied to the generated logs. The dataset is split into training and testing sets for the SVM model.

Table 1: Tools used

Tool	Version
Windows 10	22H2
Kali	2023.3
Genymotion	3.5.1
Apktool	2.9.0
Jadx-GUI	1.4.7
AndroPyTool	-
Strace	6.6

A. Data Sources

The APK files used in this study were obtained from the Canadian Institute for Cyber Security. The dataset represents live ransomware and benign APKs. It comprises 500 Benign and 70 ransomware applications from selected 10 families of ransomware, which are Charger, Jusuit, Koler, lockerPin, and Pletor.

1) Dynamic Analysis

Run-time features are extracted after executing the Android ransomware in a sandbox, this type of analysis works well if the attackers apply obfuscation techniques to make the source code unclear. Droidbox and strace are widely used tools to monitor system calls, network data, file read and write permissions, cryptographic operations performed, and send SMS and phone calls.

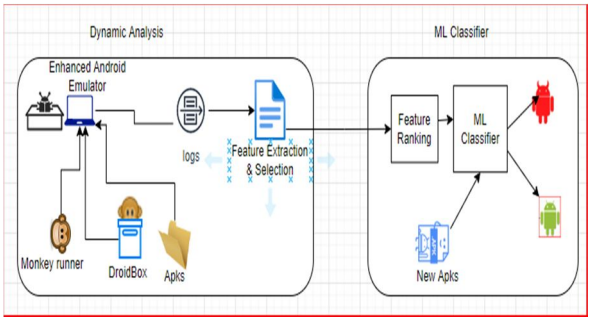


Figure 2: Dynamic analysis lab setup

AndroPyTool framework provides tools and techniques for extracting static and dynamic analysis. It combines different Android application analysis tools such as DroidBox, FlowDroid, Strace, AndroGuard and VirusTotal [23]. AndroPyTool uses all of these tools to analyse APK files from a source directory. It then creates feature files in CSV and JSON formats. To achieve this, Android APK files follow the six steps.

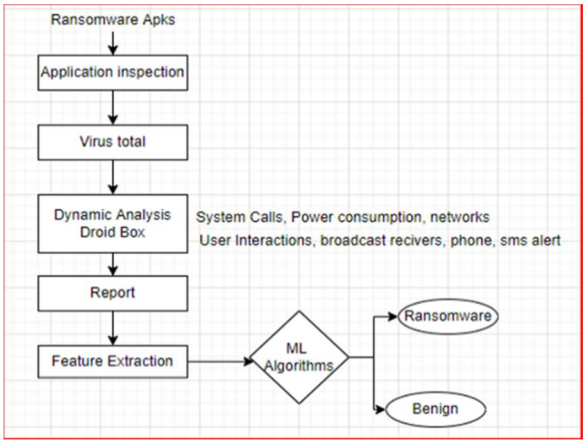


Figure 3: Lab experimental setup for dynamic analysis

2) APK Filtering

Using the AndroGuard tool, each sample is inspected in the first stage to see if it is a legitimate Android application. Screenshot 1 below shows that all the are valid APKs.

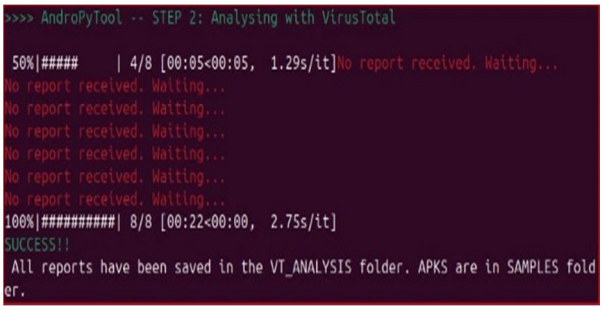
```
ubuntu@ubuntu-VirtualBox:~/Desktop$ sudo docker run --volume=/home/ubuntu/Deskto
p/Ransomware/RansomBO/:/apks alexmyg/andropytool -s /apks/ -all -vt 04d8ad9978a8
d8a7008ff9bbf12d73b6582ccceb3962043f322005a00fd74d2a
[sudo] password for ubuntu:


>>>> AndroPyTool -- STEP 1: Filtering apks

  0%|          | 0/8 [00:00<?, ?it/s]8 apks found. Processing...
 88%|########7 | 7/8 [00:01<00:00,  4.54it/s]TOTAL VALID APKS: 8
TOTAL INVALID APKS: 0
100%|##########| 8/8 [00:01<00:00,  4.55it/s]
```

Screenshot 1 shows AndropPyTool filtering APKs

3) Virus Total Analysis

Virus Total online web application tool reports the application status. The scan findings from more than 60 different anti-malware engines are included in the report.

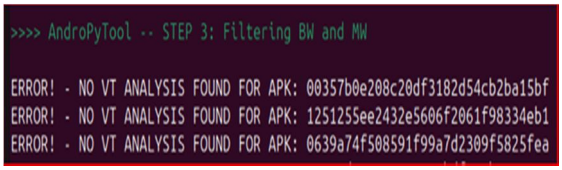
Screenshot 2 below shows the success of the Virus total analysis.



Screenshot 2 APK analysis using Virus Total

4) Dataset Partitioning

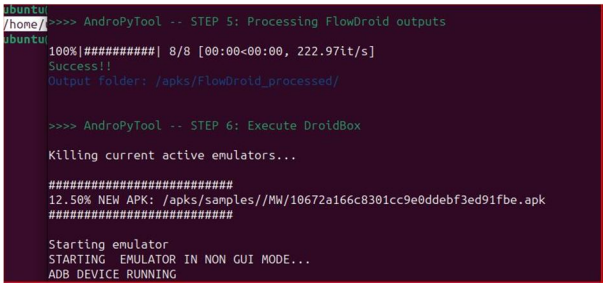
If at least many antivirus programs flag APK as malicious, the Virus Total report will classify them as malicious applications. The threshold is based on the researcher's perception.



Screenshot 3 shows the filtering of benign and ransomware APKs

5) FlowDroid Results Processing

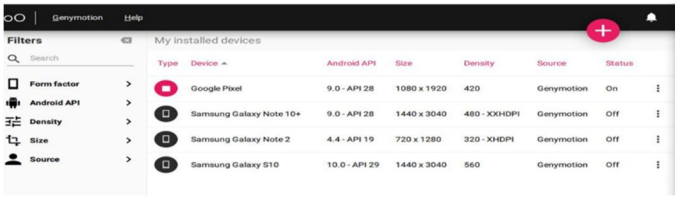
The FlowDroid extracts connections between sources and sinks. This tool is based on taint analysis, and runs against every sample, features extracted are Opcodes, API calls, system commands, and strings are also extracted. Screenshot 4 shows the process's success.



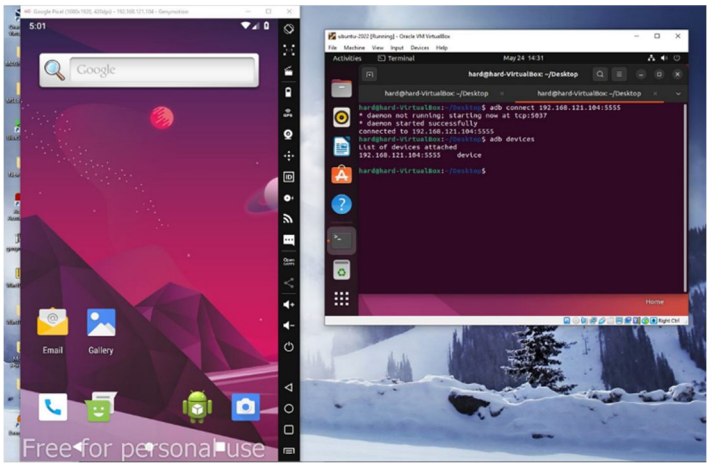
Screenshot 4 shows FlowDroid and DroidBox executing APKs

6) DroidBox Executing

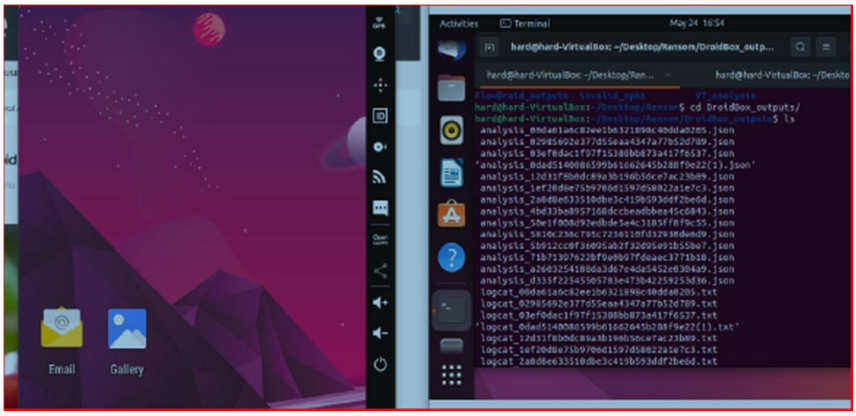
AndroPyTool is an Android dynamic analysis tool that is a modified version of DroidBox. It includes the Strace tool extracting system calls, monkey tool aims at stimulating the sample under analysis with a higher number of simulated user actions on the screen and buttons.



Screenshot 5 shows the Android instance in genymotion



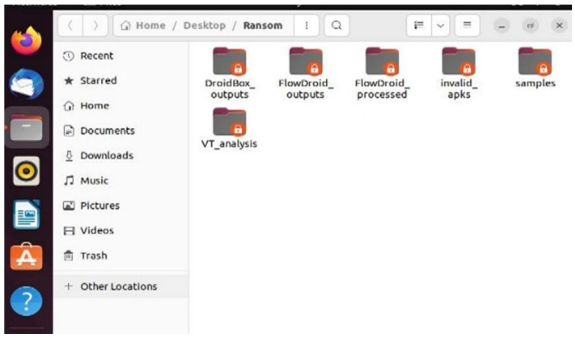
Screenshot 6 shows the ADB connection on Android devices



Screenshot 7 Shows the Droidbox outputs

The results of the AndroPyTool are served in separate folders as shown below on screenshot 7. Folder for Virus Total analysis, valid and invalid APKs, and outputs from DroidBox and FlowDroid.

7) Output

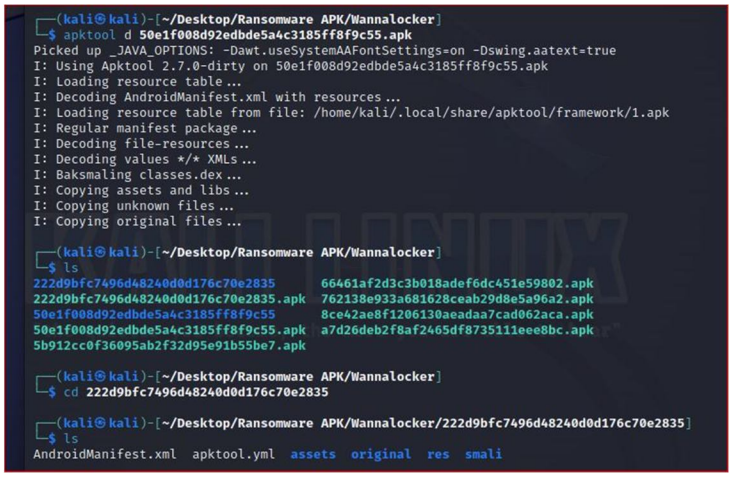


Screenshot 8: AndroPyTool outputs

B. Static Analysis

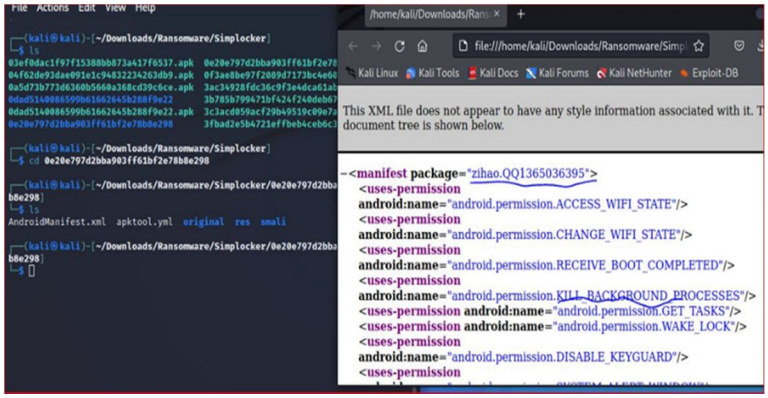
In the static analysis part, reverse engineering tools such as Apktool, Jadx-GUI, and Dex2jar tools are used to decompile the APK files. The Apktool converts the Android application into an Android manifest file, resources and dex file. Dex2jar converts dex to jar file and Jadx-gui finally converts the jar files into java codes. The files are analysed for permissions, intents, encoding, encryptions and locking methods.

Apktool decompiles APK files into AndroidManifest.xml, assets, smali, res, and original. The command used is *apktool d application name.*



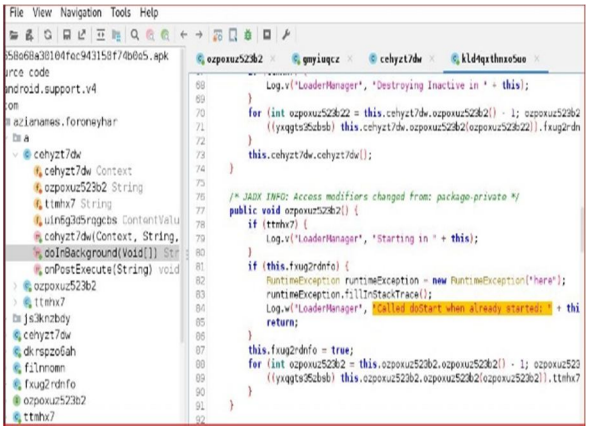
Screenshot 9 Show Apktool disassemble APKs

The Android manifest xml file shows the package name, intents and permissions. From the xml file, normal, dangerous and signature permissions are identified.



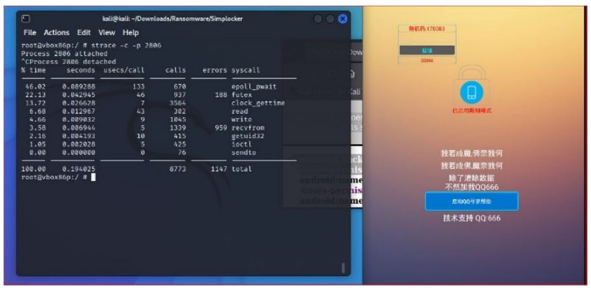
Screenshot 10: Shows APKs and package names from the Android Manifest xml file

From the Java files, is where malicious codes are identified. Figure 11 shows the Java class and methods. JADX GUI tool was used to show the Java codes.



Screenshot 11 Shows Java classes and methods using Jadx-GUI

The features of Android ransomware mentioned above are examined and stored in a database. These features will be used as input for supervised machine-learning methods to detect Android malware in the following stage.



Screenshot 12 Shows System Calls using the Strace tool

1) Feature Extraction of Android Ransomware.

Table 2: Shows features extracted

	Attributes
Static	331 permission, intents and API calls
Dynamic	85 olumns of network activities and system calls

a) SVM Model

Support Vector Machines are models that perform supervised learning on data for classification and regression. When given a labelled training dataset, it computes the optimal hyperplane that categorizes the test data. The SVM classifier model uses two SVM kernels, fitness and predict function

b) Model Evaluation

The evaluation is done using the confusion matrix, precision, recall, F1- score, and AUC.

➢ *Confusion Matrix*

A confusion matrix summarises the performance of a classification algorithm. It gives a clear picture of classification model performance and the types of errors, providing an overview of the correct and incorrect predictions in every category. The metrics computed from The F1 score, recall, precision, and TN, FP, and FN values are shown in Figure 4.

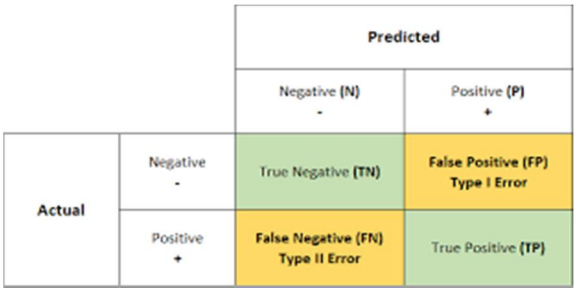


Figure 4: Shows Confusion Matrix

➢ *Precision*

Precision can be defined as the percentage of correctly predicted positive outcomes out of all the predicted positive outcomes. Precision is the number of correct positive results, divided by the number of positive results predicted.

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

The result in the [0,1] range shows how accurate the classifier's predictions are; the higher the better.

- *Recall*

Recall is the fraction of accurately predicted good outcomes among all actual positive outcomes. The recall is the number of correct positive results, divided by the number of all relevant samples

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

A recall is a number in the [0,1] range that indicates the percentage of correctly classified samples over all the samples of that class.

- *F1 - score*

The weighted harmonic mean of recall and precision is known as the F1-score. F1 scores can range from 0.0 to 1.0, with 1.0 being the best attainable score.

$$F1 - Score = 2.\frac{1}{\frac{1}{Precision} + \frac{1}{Recall}}$$

The greater the F1 score, the better the overall performance of the model.

- *The Area Under the Curve*

AUC is a metric that has values between 0 and 1. A model with 100% incorrect predictions would have an AUC of 0.0, and one with 100% correct predictions would have an AUC of 1.

IV. SUMMARY OF RESULTS

Throughout the study, the SVM model classifier was used for both static and dynamic features. The model selects six evaluation techniques for performance evaluation. The evaluation metrics are confusion matric, accuracy, recall, f1-score, precision and AUC. From the study, the model performed well in detecting, ransomware applications. The recall f1-score improved from the previous studies that showed a reduction in false alarms.

A. Classification Report

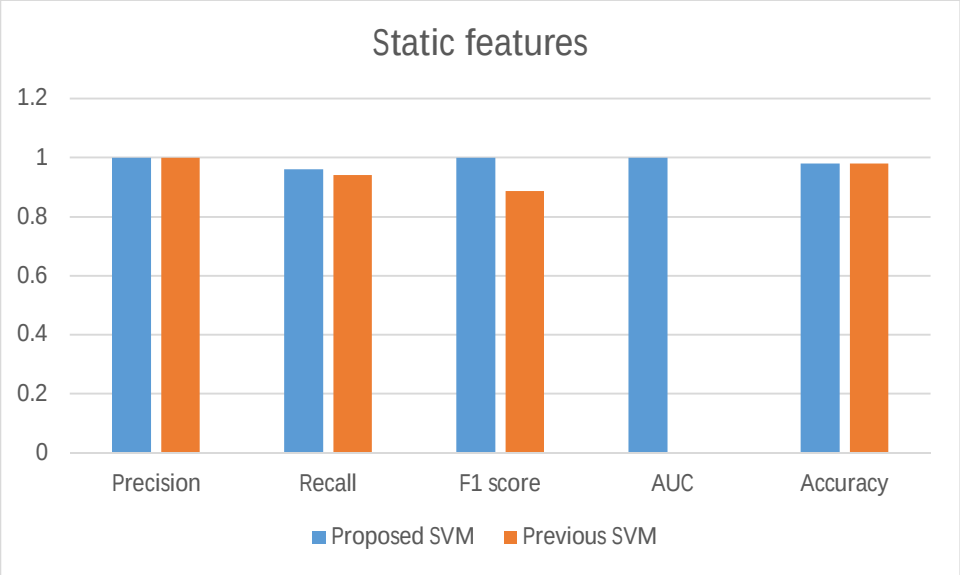
Table 3: Shows the classification report

Static features	Accuracy	Precision	Recall	F1-Score	AUC
SVM	0.9	0.9	0.89	0.9	0.89
Dynamic features	Accuracy	Precision	Recall	F1-Score	AUC
SVM	0.98	1	0.963	1	1

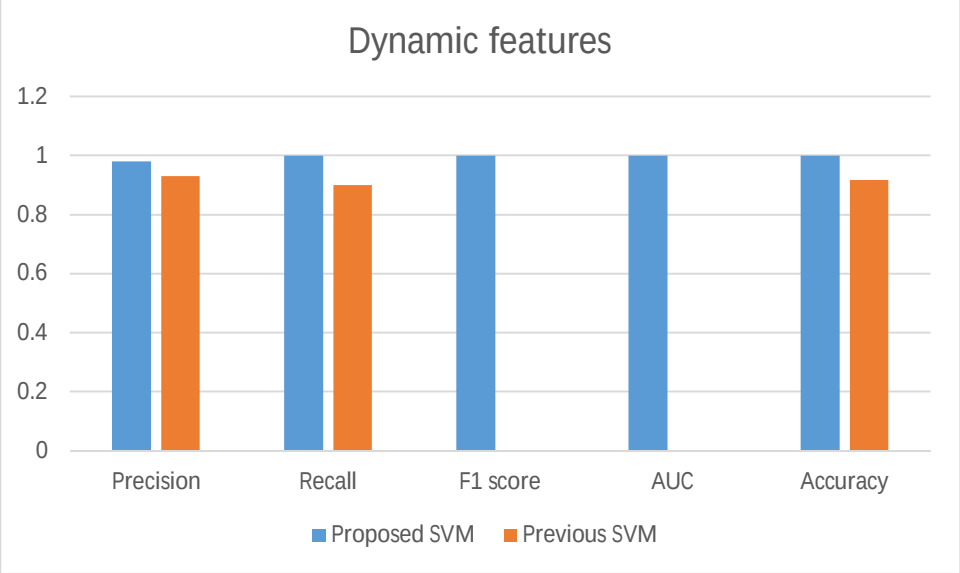
The values of the performance metrics acquired following the implementation of the suggested methodology to detect Android ransomware are displayed in Table 3 above. The results show the classification model effectively detect ransomware apps with a small margin using the dynamic features and static feature. Using the static feature the model accuracy, precision, and fi-score was 0.9, and recall and AUC was 0.89. While using dynamic feature shows the model performed 100% on all metrics.

V. COMPARATIVE ANALYSIS

Bar graph 1 and 2 below compare the performance metrics and features used in the proposed model and also the same metrics and feature from the previous studies.



Bar graph 1 shows the comparative analysis of static features using SVM model



Bar graph 2 shows a comparative analysis of dynamic features using SVM model

Table 4: Shows analysis of attributes

Attributes	Previous SVM	Proposed SVM
Static	2089 permission features and 6022 API calls features. [20]	331 permission, intents, and API-calls
Dynamic	86 Network traffic features.[24]	85 columns of network activities and system calls
Attributes	Previous SVM	Proposed SVM
Static	2089 permission features and 6022 API calls features. [20]	331 permission, intents, and API-calls
Dynamic	86 Network traffic features.[24]	85 columns of network activities and system calls

The two bar graphs compares the performance metrics of precision, recall, F1 score, AUC, and accuracy for the proposed SVM model and the previous SVM model. In terms of static features, the proposed SVM model achieves a precision of 1, recall of 0.963, F1 score of 1, AUC of 1, and accuracy of 0.98. In contrast, the previous SVM model achieves a precision of 1, recall of 0.94357, F1 score of 0.8875, and an accuracy of 0.98. Unfortunately, the exact F1 score and AUC values for the previous model are not provided in the table. Moving to dynamic features, the proposed SVM model achieves a precision of 0.98, recall of 1, F1 score of 1, AUC of 1, and accuracy of 1. On the other hand, the previous SVM model achieves a precision of 0.93 and recall of 0.90. The F1 score and AUC values are not available for the previous model, but the accuracy is reported as 0.917. Table 8 provides an analysis of the number of attributes used in the SVM models. The previous SVM model utilizes 2089 permission features and 6022 API call features for static analysis, while the proposed SVM model employs 331 permission, intents, and API call features. For dynamic analysis, the previous SVM model uses 86 network traffic features, whereas the proposed SVM model utilizes 85 columns of network activities and system calls.

VI. FUTURE WORKS

The model was trained using the SVM algorithm, a supervised classification that means that the data need to be labelled. It is recommended that other supervised algorithms such as Naïve Bay and Decision Tree be used. A combination of two or more algorithms may be used in the research. It is recommended to use unsupervised algorithms to detect ransomware APKs without labelled datasets. It is also recommended to use the same model to detect other malware like botnets and spyware.

VII. CONCLUSION

Throughout the study, the SVM algorithm was used for ransomware application detection. The study concluded that ransomware applications could be detected using dynamic and static analysis. The static features used are permission, intents, and calls, and the dynamic features used are network activities and system calls. The features performed well in training and evaluating the model. The model was evaluated using the six metrics and its performance was good. The confusion matrix, accuracy, precision, recall, F1 Score, and AUC all show positive performance. The SVM model was able to classify ransomware and benign Apk accurately. Ransomware detection is one of the widely researched domains in malware detection due to the large amounts of money in the form of ransom. Even if you pay the ransom there is no guarantee that the criminals will return the data. This study shows how permissions, intents, API calls and network activities play in detecting ransomware.

REFERENCES

[1] S. Sharma, C. R. Krishna, and R. Kumar, “RansomDroid: Forensic analysis and detection of Android Ransomware using unsupervised machine learning technique,” Forensic Science International: Digital Investigation, vol. 37, Jun. 2021, doi: 10.1016/j.fsidi.2021.301168.

[2] H. Han, S. Lim, K. Suh, S. Park, S. J. Cho, and M. Park, “Enhanced android malware detection: An SVM-based machine learning approach,” in Proceedings - 2020 IEEE International Conference on Big Data and Smart Computing, BigComp 2020, Institute of Electrical and Electronics Engineers Inc., Feb. 2020, pp. 75–81. doi: 10.1109/BigComp48618.2020.00-96.

[3] S. E. Madnick, “The Continued Threat to Personal Data: Key Factors Behind the 2023 Increase,” 2023.

[4] “9100 Cyber Report June 23 v04”.

[5] S. Baek, Y. Jung, D. Mohaisen, S. Lee, and D. H. Nyang, “SSD-Assisted Ransomware Detection and Data Recovery Techniques,” IEEE Transactions on Computers, vol. 70, no. 10, pp. 1762–1776, Oct. 2021, doi: 10.1109/TC.2020.3011214.

[6] M. N. Olaimat, M. Aizaini Maarof, and B. A. S. Al-Rimy, “Ransomware Anti-Analysis and Evasion Techniques: A Survey and Research Directions,” in 2021 3rd International Cyber Resilience Conference, CRC 2021, Institute of Electrical and Electronics Engineers Inc., Jan. 2021. doi: 10.1109/CRC50527.2021.9392529.

[7] F. Manavi and A. Hamzeh, “Static Detection of Ransomware Using LSTM Network and PE Header,” in 26th International Computer Conference, Computer Society of Iran, CSICC 2021, Institute of Electrical and Electronics Engineers Inc., Mar. 2021. doi: 10.1109/CSICC52343.2021.9420580.

[8] V. Ajiri, S. Butakov, and P. Zavarsky, “Detection Efficiency of Static Analysers against Obfuscated Android Malware,” in Proceedings - 2020 IEEE 6th Intl Conference on Big Data Security on Cloud, BigDataSecurity 2020, 2020 IEEE Intl Conference on High Performance and Smart Computing, HPSC 2020 and 2020 IEEE Intl Conference on Intelligent Data and Security, IDS 2020, Institute of Electrical and Electronics Engineers Inc., May 2020, pp. 231–234. doi: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00049.

[9] F. Manavi and A. Hamzeh, “Static Detection of Ransomware Using LSTM Network and PE Header,” in 26th International Computer Conference, Computer Society of Iran, CSICC 2021, Institute of Electrical and Electronics Engineers Inc., Mar. 2021. doi: 10.1109/CSICC52343.2021.9420580.

[10] B. Qin, Y. Wang, and C. Ma, “API Call Based Ransomware Dynamic Detection Approach Using TextCNN,” in Proceedings - 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering, ICBAIE 2020, Institute of Electrical and Electronics Engineers Inc., Jun. 2020, pp. 162–166. doi: 10.1109/ICBAIE49996.2020.00041.

[11] R. Thangaveloo, W. W. Jing, C. K. Leng, and J. Abdullah, “DATDroid: Dynamic analysis technique in android malware detection,” Int J Adv Sci Eng Inf Technol, vol. 10, no. 2, pp. 536–541, 2020, doi: 10.18517/ijaseit.10.2.10238.

[12] M. Pasetto, N. Marastoni, and M. D. Preda, "Revealing Similarities in Android Malware by Dissecting their Methods," in Proceedings - 5th IEEE European Symposium on Security and Privacy Workshops, Euro S and PW 2020, Institute of Electrical and Electronics Engineers Inc., Sep. 2020, pp. 625–634. doi: 10.1109/EuroSPW51379.2020.00090.

[13] M. Gracea and M. Sughasiny, "Malware detection for Android application using Aquila optimizer and Hybrid LSTM-SVM classifier," EAI Endorsed Transactions on Scalable Information Systems, vol. 10, no. 1, 2023, doi: 10.4108/eetsis.v9i4.2565.

[14] T. A. A. Abdullah, W. Ali, and R. Abdulghafor, "Empirical Study on Intelligent Android Malware Detection based on Supervised Machine Learning," 2020. [Online]. Available: www.ijacsa.thesai.org

[15] H. Alkahtani and T. H. H. Aldhyani, "Artificial Intelligence Algorithms for Malware Detection in Android-Operated Mobile Devices," Sensors, vol. 22, no. 6, Mar. 2022, doi: 10.3390/s22062268.

[16] C. Gupta, R. K. Singh, S. K. Bhatia, and A. K. Mohapatra, "Decadroid classification and characterization of malicious behaviour in android applications," International Journal of Information Security and Privacy, vol. 14, no. 4, pp. 57–73, Oct. 2020, doi: 10.4018/IJISP.2020100104.

[17] C. Nugent, W. Guo, P. Müller, and Y. Ji, "Bayesian Approaches to Subgroup Analysis and Related Adaptive Clinical Trial Designs," JCO Precis Oncol, no. 3, pp. 1–9, 2019, doi: 10.1200/po.19.00003.

[18] R. Thangaveloo, W. W. Jing, C. K. Leng, and J. Abdullah, "DATDroid: Dynamic analysis technique in android malware detection," Int J Adv Sci Eng Inf Technol, vol. 10, no. 2, pp. 536–541, 2020, doi: 10.18517/ijaseit.10.2.10238.

[19] J. Zhou, W. Niu, X. Zhang, Y. Peng, H. Wu, and T. Hu, "Android Malware Classification Approach Based on Host-Level Encrypted Traffic Shaping," 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2020, doi: 10.1109/ICCWAMTIP51612.2020.9317429/20/$31.00.

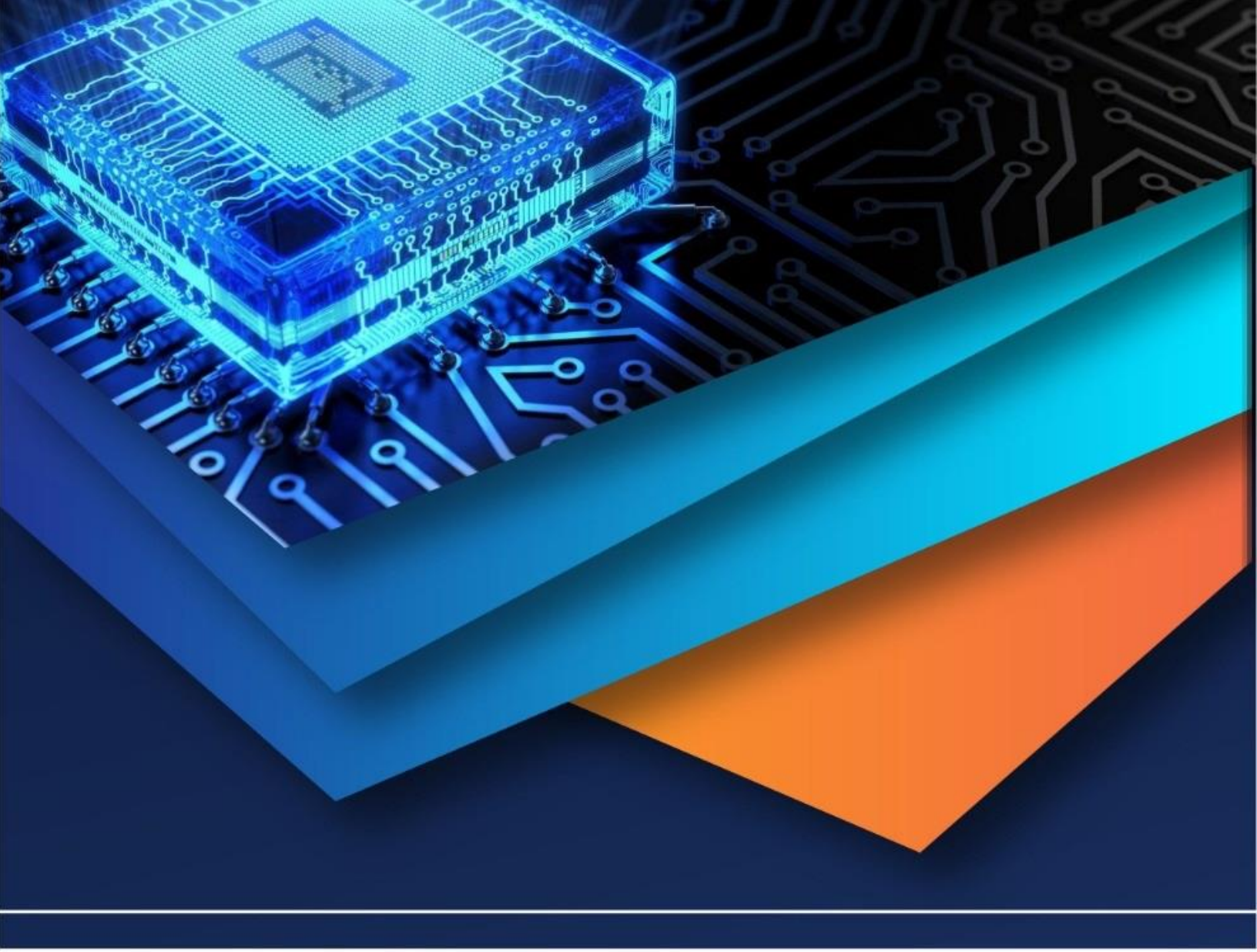
[20] I. Almomani, A. Alkhayer, and M. Ahmed, "An Efficient Machine Learning-based Approach for Android v.11 Ransomware Detection," in 2021 1st International Conference on Artificial Intelligence and Data Analytics, CAIDA 2021, Institute of Electrical and Electronics Engineers Inc., Apr. 2021, pp. 240–244. doi: 10.1109/CAIDA51941.2021.9425059.

[21] S. Alsoghyer and I. Almomani, "On the Effectiveness of Application Permissions for Android Ransomware Detection," in Proceedings - 2020 6th Conference on Data Science and Machine Learning Applications, CDMA 2020, Institute of Electrical and Electronics Engineers Inc., Mar. 2020, pp. 94–99. doi: 10.1109/CDMA47397.2020.00022.

[22] S. Sharma, C. R. Krishna, and R. Kumar, "Android Ransomware Detection using Machine Learning Techniques: A Comparative Analysis on GPU and CPU," in Proceedings - 2020 21st International Arab Conference on Information Technology, ACIT 2020, Institute of Electrical and Electronics Engineers Inc., Nov. 2020. doi: 10.1109/ACIT50332.2020.9300108.

[23] A. Martín, R. Lara-Cabrera, and D. Camacho, "Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset," Information Fusion, vol. 52, pp. 128–142, Dec. 2019, doi: 10.1016/j.inffus.2018.12.006.

[24] R. Chiwariro and L. Pullagura, "Malware Detection and Classification Using Machine Learning Algorithms", International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 11 Issue VIII Aug 2023, doi: https://doi.org/10.22214/ijraset.2023.55255.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY