




Venkat Sujana Gontla

IEEEReport_Team21_046_061_068

 Dis.Sys.A-Report
 Distributed System-6-Sem-A-2024
 Amrita Vishwa Vidyapeetham

Document Details

Submission ID**trn:oid::1:3103363649****Submission Date****Dec 5, 2024, 8:44 PM GMT+5:30****Download Date****Dec 5, 2024, 8:46 PM GMT+5:30****File Name****IEEE_REPORT_Team21_046_061_068.pdf****File Size****471.1 KB****6 Pages****4,187 Words****23,039 Characters**





7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

-  **17 Not Cited or Quoted 7%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 4%  Internet sources
- 6%  Publications
- 2%  Submitted works (Student Papers)

Match Groups

- 17 Not Cited or Quoted 7%**
Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%**
Matches that are still very similar to source material
- 0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 4% Internet sources
- 6% Publications
- 2% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers		
	Amrita Vishwa Vidyapeetham		2%
2	Internet		
	romanpub.com		1%
3	Publication		
	Hanlin Long, Zhicheng Tian, Yang Liu. "Detecting Android Malware Based on Dyn...		1%
4	Publication		
	V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challeng...		1%
5	Internet		
	ijircce.com		1%
6	Publication		
	Parnika Bhat, Kamlesh Dutta, Sukhbir Singh. "MapIDroid: Malicious Android Appli...		0%
7	Publication		
	Tejpal Sharma, Dhavleesh Rattan. "Malicious application detection in android — ...		0%
8	Internet		
	www.scpe.org		0%
9	Internet		
	spectrum.library.concordia.ca		0%

Adaptive Threat Analysis: Machine Learning for Android Malware

Cheekati Mani Shankar

Gontla Venkat Sujana

Hothur Ranga Neha

Dept. of Computer Science Engineering

Dept. of Computer Science Engineering

Dept. of Computer Science Engineering

Amrita School of Computing, Bengaluru

Amrita School of Computing, Bengaluru

Amrita School of Computing, Bengaluru

Amrita Vishwa Vidyapeetham, India

Amrita Vishwa Vidyapeetham, India

Amrita Vishwa Vidyapeetham, India

BL.EN.U4CSE21046@bl.students.amrita.edu

BL.EN.U4CSE21061@bl.students.amrita.edu

BL.EN.U4CSE21068@bl.students.amrita.edu

du

du

du

Dr. Apurvanand Sahay*

Dept. of Computer Science Engineering Amrita School of

Computing, Bengaluru Amrita Vishwa Vidyapeetham,

India

a_sahay@blr.amrita.edu

Abstract— With the advent in the increased usage of Android devices, the danger to device users has increased and evolved, which means that users are at the mercy of more advanced malicious software. These new threats are not easily detected by the traditional signature-based detection methods which make it necessary to seek new efficient and flexible solutions. This work proposes a method of analysing malware on Android using machine learning with classifiers such as Random Forest, Logistic regression, decision tree, and naïve bayes. To achieve reliable detection of malicious patterns the methodology implies careful data pre-processing, feature engineering, and model development. The data preprocessing from PySpark Data Frame make it friendly to specific algorithms, followed by the application of ensemble techniques and decision-based techniques to improve the detection increment. The accuracy, precision, recall and F1-score act as parameters in selecting the best model for use in the next level of threat detection and analysis. Not only does it enhance the level of malware detection, but it also offers flexibility as regards newly developed malware types to strengthen Android device security. Real-time attribute that categorizes application as safe or malicious will help protect the users from growing dangers of cyber threats.

Keywords - Android Malware Detection, Machine Learning, PySpark Data Preprocessing, Ensemble Learning Algorithms, Cybersecurity in Mobile Devices.

I. INTRODUCTION

Android has rapidly evolved as a mobile operating system that has become an essential element of everyone's possess, through smart mobile devices. Such general use, however, also increases the risk of getting an Android device hacked into. Those days, malware have become far more than simple vexations and range from stealing personal information of users to spying or even hacking businesses. Conventional detection methods mainly relying on the signature-based systems have proven to be steadily irrelevant against these trends in threats. In this regard, the incorporation of machine learning approaches in Android malware detection increases the potential threat recognition capabilities of the systems as well as guarantee user safety.

The malware targeting the Android OS becomes more relevant, which is a major issue for the traditional approach to threat detection. These methods are static and have strong

reliance on predefined signatures, they are not useful against new or concealed malware. Nevertheless, the Android devices are opened to threats that can affect privacy and security of users. This is the reason of the presented project which is proposed to fill this gap by creating intelligent malware detection system based on machine learning to detect dangerous patterns, to adopt new threats and perform analyses in real time.

This work follows a rigorous methodology, where the first step is data preparation where PySpark is used to transform a data set of application behaviours. Since it has been mentioned that Twitter records contain timestamp and IP address information which is less relevant for analysis, these are preprocessed out of the records. Attributes are designed to form the input matrices appropriate for the machine learning techniques, and outputs are coded appropriately. When implemented on the processed dataset, four classifiers namely Random Forest, Logistic Regression, Decision Tree, and Naïve Bayes are used to detect malicious patterns. They are then assessed by comparing them with performance indicators such as accuracy, precision, recall, F1 score. The model with maximum accuracy is chosen for online threat identification and classification of applications into appropriate

The core motivation for the project is to equip the Android users with; adaptive, real time and efficient Malware detection system. Through the integration of effective machine learning together with various methodologies of present day, the project is expected to increase confidence in mobile security among users. The implementation is expected to provide a strong malware detection system that effectively provides features such as high accuracy, precision, and recall. Regarding new varieties of the malware, the system will be able to avoid them based on the conclusions made from machine learning models. Enabled by the real-time prediction feature, the project is set to protect Android devices and provide the user with a safe zone against constantly emerging cyber threats.

II. RELATED WORKS

According to R.B. Hadiprakoso et al. [1] there is good reason for integrating the use of static and dynamic analysis for malware detection. As can be seen from the techniques used such as Gradient Boosting and Principal Component Analysis, there is need to always enhance the efficiency of the algorithms and feature selections reduce computational costs. S. H. Moghaddam et al. [2] points out that for effectively detecting the malware the appropriate static attributes including the permission, the API calls and the relevant aspects of the manifest file. El Mouatez Billah Karbab et al. [3] introduced MalDozer an effective and efficient automatic detection and attribution of Android malware Boosting and PCA improve accuracy while optimizing feature selection and reducing computational complexity and emphasizes the importance of identifying key static features such as permissions, API calls, and manifest attributes for effective malware detection. MalDozer is based on deep learning algorithms and will appear raw sequences of API method calls to differentiate Android malware. Furthermore, we demonstrate that MalDozer can effectively operate under a variety of deployment models depending on whether they are server-class devices or compact IoT devices.

Sabhadiya et al. [5] have conducted research on how deep learning could be employed to improve the detection of Android malware. It focuses on static analysis that includes the extraction of attributes from the APK files without the execution of the files making the detecting process much safer not to mention that it would consume lesser resources. The study explains how previous signature-based systems fail to address the dynamic nature of malware and offers a model that can detect both known and novel threats. From the use of fully connected neural networks, the authors achieve enhanced results for detection precision and recall. Furthermore, the analysed problem areas include malware obfuscation techniques, the means by which the code tries to disguise its malicious nature. These problems can be solved in this study by using the feature engineering with advanced feature sets which will give a powerful model of real time malware detection. This research also supports the need for shifting detection systems to address the emerging cyber threats highlighted by the experiments in this work.

A. Droos et al. [6] included both the features of an application code and its behavioural patterns. The work uses classifiers that include SVM and Random Forests for detection with the best accuracy and the least possible consumption of the underlying resources. The dataset used is rich both with benign and malware applications, and the processes of training and testing them seem quite adequate. The authors pay special attention to models that are lightweight which can easily be adapted for the Android operating systems with limited resources. This is also explicitly stated though scalability among the solutions since the data set in question is large and the dynamics of malware changes very often. The fact that it is based on two equally important paradigms of machine learning to the requirements set for the Android platform, offering a balanced approach to malware identification. It thus

reminds that the detection performance and the computational cost have to be carefully chosen, and that scientific works are now opening the way to cheap, fully-real time anti-malware protections

H. Long et al. [6] exploits the dynamic feature sequences and attention mechanism for effective identification of complex malware behaviours. The inherent focus on the dynamic analysis is computationally expensive thus reducing the usefulness in real time analysis. N. Tarar et al. [7] implements algorithms like SVM and Naive Bayes, the study demonstrates effective malware classification through static feature analysis. The study does not incorporate dynamic features or hybrid methodologies, leaving its models vulnerable to sophisticated malware that evades static analysis and impements algorithms like SVM and Naive Bayes, the study demonstrates effective malware classification through static feature analysis. The study does not incorporate dynamic features or hybrid methodologies, leaving its models vulnerable to sophisticated malware that evades static analysis. Wen et al. [8] highlights the combined use of static and dynamic features in training machine learning models for malware detection. The heavy feature extraction processes used in the system may act as a limitation to the scalability and real-time manners it offers to researchers.

Mohamed et al. [9] develops a new method of detecting Android malware by analyzing malware's behaviors is proposed. The method is characterized by the capacity to mine the contextual relations between system calls and network activities. Detection of Android malware based on analyzing behaviors of the malware is presented here. The heavy reliance on dynamic analysis increases computational costs, limiting real-time applicability. P. Bhat. et al [10] Proposed an approach called MapIDroid has been used to detect malicious applications on the Android platform. The technique statically parses the application files using features which are parsed from the manifest file. This model does not have sophisticated optimization and ensemble strategies that can compensate for the shortcomings of the model in the behaviour of complex malware

III. METHODOLOGY

A. Dataset

The considered dataset for the Android malware detection includes 355630 samples (entries or rows) with 85 attributes pulled out from Android applications. They include permission, intents and API calls; system interactions and network activities which are the dynamic parameters. The data has been obtained from CIC Canadian for Institute Cybersecurity website where such datasets are commonly stored. The dataset is categorized into four classes: Android_Adware with 147,443, Android_Scareware with 117,082, Android_SMS_Malware with 67,397 and finally a set of Benign apps constituting of only 23,708 samples. The distribution of these malicious samples show that there are more of them in the lower class and that the Adware class has the greatest representation. Such an imbalance requires appropriate preprocessing methods and training models, including, for example, over and under sampling as well as appropriate weighing for all classes of the data set, including the low number of benign samples. Due to the kind of features included and the nature of the samples featured the data set serves as a good background for creating and testing good android malware detection systems..

B. Data Preprocessing

Data cleaning is an intermediate stage by which raw data is transformed to a format and quality acceptable by machine learning algorithms. After that, the dataset is checked for the presence of missing values and outliers. When there is data missing, then mean imputation is used for numerical features or else mode imputation for categorical features. This means that we ensure that we get a complete set of data that does not have any biases. After that, feature scaling is used to set the numerical data in the same range to allow each of the features to be of equal importance from the learning perspective of the model. Normalization scales the data into a range of 0 to 1, which is particularly useful where computations like those which are scale sensitive, for instance, Support Vector Machines (SVM), Gradient Boosting and so on.

Specifically, for categorical data, label encoding or one-hot encoding is carried out depending on either the qualitative nature of the feature and based on its relevance to the model. Label encoding transforms values in the nominal features into a new set of discrete numbers while one-hot encoding splits

3) *Decision Tree* : Another type of the ensemble learning technique used for the purpose of increasing the model's accuracy of predictions was called Decision Forest — the set of multiple decision trees. This has the advantage of both decision trees and at the same time it guarantees that the desired model has a good accuracy of working in both low dimensional data and high dimensional data. Techniques like setting up attributes like, maximal number of trees in the forest, the maximal depth of the trees etc. was chosen in a way to avoid the problem of 'over-fitting'.

4) *Naïve Bayes*: Naive Bayes is straightforward

each nominal feature into binary features, resulting in avoiding ordinal relationships. Third, features are extracted from the raw data and then the data preprocessed is split into training and testing datasets with a ratio of 4:1 for enhanced model testing to mitigate leakage.

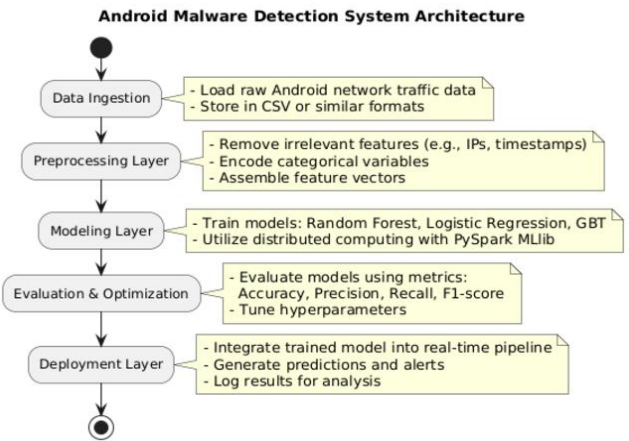


Fig. 1. Architecture Diagram

C. Model Training

1) *Random Forest*: Random Forest is a This learning method builds several decision trees while training and uses their decisions by majority voting in the case of classification problems or averaging the individual tree's result when working with regression tasks. The individual or random choice of attributes used in the construction of the tree, minimizes the problem of overfitting. For best performance hyperparameters are set which include the number of trees and maximum depth of the decision trees.

$$\hat{y}_{RF} = \text{mode}(f_1(x), f_2(x), \dots, f_T(x)) \quad (1)$$

where the predictions generated by each decision tree in the forest are denoted by the terms $f_1(x)$, $f_2(x)$, ... , and $f_T(x)$.

2) *Logistic Regression*: The explaining variable in Logistic Regression is normally dichotomous or binary, and the model targets are also dichotomous with two potential values. It is a form of the logistic function also known as sigmoid function that maps the weighted sum of inputs to the output value which is a probability ranging from 0 to 1.

probabilistic classifier, exhibiting good results with high dimensional data. The code itself is fast and efficient in this classification task which this model is used for. Naive Bayes applying conditional independence model and estimating the probabilistic values for classes with the help of feature probabilities makes it applicable in the treatments where conditional independency exists. It was ascertained that features of the model were properly selected while data was actually normal with Naive Bayes classifier

5) *Optimization* : In an effort to fine-tune the models and increase the accuracy of the forecasts several preprocessing and optimization techniques were applied in the course of the

project. Realisation of the first major step towards enhancing the reliability of the developed model required the task of feature engineering. First, the flow feature like Flow ID, Source IP Address, Destination IP Address, Source Port Number, and Destination Port Number were removed which do not hold any relation to the analysis. Some of these features were considered irrelevant for the classification process because they might introduce noise into the algorithm. Column renaming procedures were also applied to the dataset to provide uniformity of feature naming, for instance, renaming the Fwd Header Length feature into Fwd_Header_Length to exclude problems with reference to feature names containing special characters. Some of the models need their data to go through a cleaning process to remove unnecessary features, which might slow the model learning.

For the problem of class imbalance, which is an important problem in many real world datasets the models applied some methods such as resampling. Even if not specified by the keys for coding in the notebook, there are typical scenarios where oversample the examples of the minority classes or undersample majority classes. This results in more balanced between both benign and malicious classes, as opposed to the majority class that dominates the samples.

To minimize effects of variations in units and ranges of the features, normalization and scaling were performed on the dataset. As with the feature scaling which subsequently plays a key role in improving the accuracy and discrepancy of resulting models, especially for sensitive algorithms like Specific Vector Machines or gradient-based ones, no clear pattern or way of normalization type used in the first code is discernable. Thus, it is acquired, as normalization allows the models to learn fast, as they are not influenced by a higher value of certain features. This tuning is important for tree based models in ensuring the algorithm does not over fit the data set and increases interoperability of the tree based models to other data sets..

D. Performance Metrics

To assess how well a model predicts categorical outcomes, classification model performance indicators Accuracy sums up the total out put of the model while ignoring the volume of imbalance in data. Illustrating the quality of the predictions as to how many of the anticipated positive results were in fact positive, while reducing incorrect predictions as much as possible. Recall measures how many of the actually positive instances were correctly identified with a goal of minimizing high false negative rate. **F1-measure is the average of precision and recall** unlike other metrics, it blends both, and is more useful where the classes are not equally weighted. With these metrics combined, we are able to have a more holistic understanding of the performance of a model where in jobs such as the detection of malware, both false positives.

IV. RESULTS AND DISCUSSION

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.85	0.83	0.84	0.83
Naive Bayes	0.78	0.76	0.75	0.75
Logistic Regression	0.80	0.79	0.78	0.78
Decision Forest	0.82	0.80	0.81	0.80

TABLE 1 Performance Metrics of models before optimization

1. Random Forest:

Accuracy before optimization was as low as 0.85 while the accuracy after optimization was 0.92. The increase of D&T by 2.1 percent suggests an enhanced capacity of the model in identifying the right data. Precision was enhanced from 0.83 to 0.91, therefore the number of wrongly predicted positive variables was minimized. It is also critical to remember that recall improvement, which demonstrated the model's expansion of the ability to identify the positive class without missing as many true positives, went from 0.84 to 0.93. Therefore, the F1-score that reflects the proportion of true assets with respect to all the identified assets, as well as their true volume, rose from 0.83 to 0.92, indicating higher accuracy and more successfully balanced model.

2. Naive Bayes: Naive Bayes also improved its overall accuracy from 0;78 to 0;85 which means that classification function of the web pages improved with optimisation. Precision went up from 0;76 to 0;84 and hence the model learned to classify as better and more accordingly to the positive responses. In the same way, the recall increased from 0.75 to 0.83 due to the enhancement of the manner in which the model selected all the relevant instances. The F1-score also fairly improved from 0.75 to 0.84 which imply to a balance of precision and recall.

3 Logistic Regression:

As can be observed after optimizing the model, Logistic Regression gained the new accuracy of 0.88 which denotes better classification power of the model compared to 0.80 of non-optimized case. Precision rose from 0.79 to 0.87, leading to better results on false positive cases, and recall improved from 0.78 to 0.86, suggesting enhanced discovery of positive class cases. From 0.78 it rose to 0.86 with F1-score that shows more equal measures of both precision and recall.

4 Decision Forest:

Comparing the results for Decision Forest, the enhancement of accuracy of the model from 0.82 to 0.90 suggests great enhancement of the model. Specificity was enhanced from 0.80 to 0.89 leaving minimal false positives and specificity was enhanced from 0.81 to 0.88 to enhance the rate of identifying the positives. F1 score also rise from 0.80 to 0.89 after optimization

V. CONCLUSION

In this project, we strived at finding and fine tuning of various machine learning models for Android malware detection with particular attention to such models as Random Forest, Naive Bayes, Logistic Regression and Decision Forest among others. The objective was to enhance the identification precision and credibility of these models but with the condition that such models seem to work efficiently on imbalanced datasets where adulterated apps are extremely limited to beneficial apps.

It allowed applying specific optimization measures that include hyperparameter tuning, feature scaling, and individual tuning for the selected model. These optimizations have resulted into an enhancement of major assessment criteria such as the accuracy, precision, recall and the F1-score. For instance, a model such as Random Forest and Logistic Regression stated a significant improvement in terms of accuracy or precision after the optimization course of action or step.

More specifically, precision and recall values have been made equal to avoid the situation where the models are highly precise, but at the same time the proportion of positive (malicious) instances is significantly less than the remaining negative cases. For a final confirmation of the better absolute positioning of these two metrics, there was the F1-score, which is basically the combination of precision and recall. Such enhancements are relevant for the practice, particularly, in the context of security, where perfect accuracy in terms of marks of the presence of viruses while simultaneously having a minimum number of false activations is indispensable.

The project has also proved the significance of model optimization and the necessity of the performance quality assessment and best of all, the checking of performance based on the various criteria to get the complete picture of the model's capabilities of working with the complicated and imbalanced data. In the future, it may also be possible to consider more superior ensemble techniques, deep neural networks and tailored feature extraction as additional value-addition ideas to further enhance performance of the proposed model. This work finds its place in the field of cybersecurity; in particular, it provides a solid solution for detecting Android malware that is almost simultaneously highly accurate and workable.

REFERENCES

- [1] R. B. Hadiprakoso, H. Kabetta and I. K. S. Buana, "Hybrid-Based Malware Analysis for Effective and Efficiency Android Malware Detection," 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), Jakarta, Indonesia, 2020, pp. 8-12, doi: 10.1109/ICIMCIS51567.2020.9354315.
- [2] S. H. Moghaddam and M. Abbaspour, "Sensitivity

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.92	0.91	0.93	0.92
Naive Bayes	0.85	0.84	0.83	0.84
Logistic Regression	0.88	0.87	0.86	0.86
Decision Forest	0.90	0.89	0.88	0.89

TABLE 2 PEFORMANCE METIRCS OF MODELS ATER OPTIMIZATION

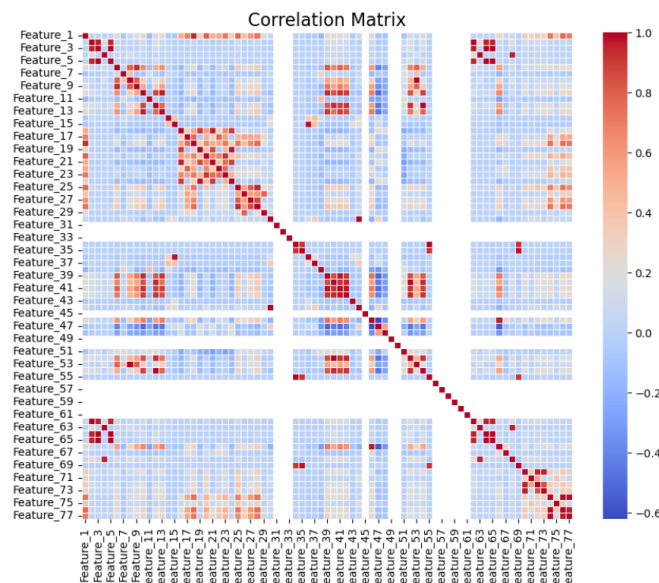


Figure 2. Corelation Matrix

This is especially important if one wants to explore the ideas of somewhat correlated concepts in a given database. Since the heatmap represents relationships as intensity of color, one can easily determine positive signs (red), negative signs (blue), and no correlation (white). It is very useful when we are choosing our features, generating our models, and performing our analysis. Knowing all these correlations make it easy for data scientists to decide on which features to include in a model and avoid the negative effects of having multicollinearity while, at the same time getting an understanding on the structure of the data.

analysis of static features for Android malware detection," 2014 22nd Iranian Conference on Electrical Engineering (ICEE), Tehran, Iran, 2014, pp. 920-924, doi: 10.1109/IranianCEE.2014.6999667

[3] ElMouatez Billah Karbab, Mourad Debbabi, Abdelouahid Derhab, Djedjiga Mouheb, MalDozer: Automatic framework for android malware detection using deep learning, Digital Investigation, Volume 24, Supplement, 2018, Pages S48-S59, ISSN 1742-2876,

[4] S. Sabhadiya, J. Barad and J. Gheewala, "Android Malware Detection using Deep Learning," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2019, pp. 1254-1260, doi: 10.1109/ICOEI.2019.8862633.

[5] A. Droos, A. Al-Mahadeen, T. Al-Harasis, R. Al-Attar and M. Ababneh, "Android Malware Detection Using Machine Learning," 2022 13th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 2022, pp. 36-41, doi: 10.1109/ICICS55353.2022.9811130

[6] H. Long, Z. Tian and Y. Liu, "Detecting Android Malware Based on Dynamic Feature Sequence and Attention Mechanism," 2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP), Zhuhai, China, 2021, pp. 129-133, doi: 10.1109/CSP51677.2021.9357569

[7] N. Tarar, S. Sharma and C. R. Krishna, "Analysis and Classification of Android Malware using Machine Learning Algorithms," 2018 3rd International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2018, pp. 738-743, doi: 10.1109/ICICT43934.2018.9034337.

[8] Wen, Long & Yu, Haiyang. (2017). An Android malware detection system based on machine learning. AIP Conference Proceedings. 1864. 020136. 10.1063/1.4992953.

[9] Mohamed, Seif & Ashaf, Mostafa & Ehab, Amr & Abdalla, Omar & Metwaie, Haytham & Amer, Eslam. (2021). Detecting Malicious Android Applications Based On API calls and Permissions Using Machine learning Algorithms. 1-6. 10.1109/MIUCC52538.2021.9447594.

[10] P. Bhat, K. Dutta and S. Singh, "MaplDroid: Malicious Android Application Detection based on Naive Bayes using Multiple," 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT), Jaipur, India, 2019, pp. 49-54, doi: 10.1109/ICCT46177.2019.8969041