International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-11 Issue-1, May 2022

Detection of Android Malware using Machine Learning and Deep Learning Review

Kiran K. Joshi

Android apps are fast evolving throughout the mobile ecosystem, yet Android malware is always appearing. Various researchers have looked at the issue related with detection of Android malware and proposed hypothesis and approaches from various angles. According to existing studies, machine learning and deep learning seems to be an effective and promising method for detecting Android malware. Despite this, machine learning is used to detect Android malware from various angles. By evaluating a broader variety of facets of the issue, the review work complements prior evaluations. The review process undertakes a systematic literature review to discuss a number of machine learning and deep learning technology that might be used to detect and prevent Android malware from infecting mobile devices. This is a strategy to cope with the rising threat of malware in the Android apps.

Keywords: Android, Malware, Machine Learning, Deep Learning

I. INTRODUCTION

The use of smartphones is expanding at a faster rate than it has ever been previously. Globally, the number of smart mobile devices will reach 3.8 billion by 2021.Furthermore, more than 72% of these smartphones are having android operating system as shown in figure1 [41].As antivirus software is seldom installed on Android smartphones. Even those that install it may not be able to utilize it to identify malware very successfully [16] because of the enormous number of users and the large quantity of important information they may access on these devices, these aspects may make the Android system more appealing to cyber attackers [16]. Malware is one of the major security threats in computer and network environment[26].Specifically, as the no of users increases, the quantity of important information that a cyber attacker may access increases. As of the 3rd quarter of year 2020, over 2.86 million Android apps were available for download [16]. Furthermore, 482,579 malicious apps were identified in June 2016 to March 2020 [36]. More complex malware detection systems are required because of the large number of malicious apps.

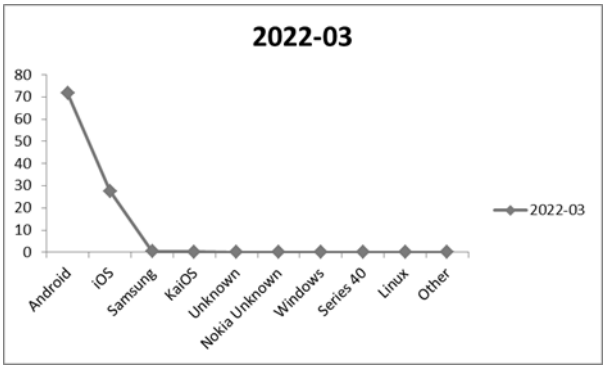

Figure1: Market of android os worldwide

The proposed work examines various machine learning(ML) and deep learning(DL) approaches that may be used to identify and mitigate Android malware access to Android phones as a solution to the issue of escalating Android application malware dangers.

II. RELATED WORK

Since the discovery of the 1st malicious Android application in early decade, several Android malwares are detected [2]. According to Kaspersky Lab, a firm that produces anti-malware software, there will be various harmful Android applications in 2020, marking a three-year rise [43]. Android malware apps are getting more common as technology lets developers all around the globe create more Android applications. According to StatCounter [41], the influence of the Android OS, which has a more than 70 per cent worldwide market share as of 2022, is predicted to increase the proliferation of harmful applications as shown in figure 1. This underscores the necessity to develop a solution that fully protects Android phone users by detecting applications which are malicious and assisting them in blocking their access to the system before any severe damage is done. Consequently, a comprehensive examination of the most effective strategies has been carried out. In 2020, Liu et al. [17] released their study on ML-based malware detection of Android. Lee et al.[16] investigated DP-based Android malware detection techniques. Experiments with 12 different machine learning algorithms to determine the most effective malware detector lead Rana et.al. [24] to an algorithm that detects malware better. Anti-malware tools and frameworks for recognizing harmful real-world apps have been created as a consequence of these findings. Ahmadi et al. [3] proposed an IntelliAV system that employs machine learning to detect harmful applications.

Manuscript received on 23 April 2022.
Revised Manuscript received on 30 April 2022.
Manuscript published on 30 May 2022.
* Correspondence Author
Kiran K. Joshi*, Ph.D. Student, Department of Computer Engineering & IT, VJTI, Mumbai (Maharashtra), India. Email: kkjoshi@it.vjti.ac.in

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

They developed an anti-malware tool that outperformed commercial alternatives and released it on the Google Play store. Mahindru[18] proposed the MLDroid framework. Experiments with over 500,000 applications using various machine learning algorithms revealed that 98.8 per cent of them could be located. As per Liu et al. [17], several researchers have utilized feature selection and information gain and other ways of limiting the features used in machine learning to increase normalization performance and operational efficiency. These efforts are crucial since they have shown that selection of feature may be used efficiently to identify Android malware using comparison trials with multiple feature selection methodologies [25]. According to Merceedi et al., [21] however, each feature selection procedure Consequently, research into implementing feature selection using genetic algorithms, which are more sophisticated than earlier methodologies, has been conducted. Fatima et al. [9] affirmed it by strengthening an SVM classifier and neural networks of 33–40 features chosen from a set of 99 features using genetic algorithms. Lee et. al.[15] were the first to present Android malware detection due to genetic selection using a genetic algorithm. Yaldiz et al. [30] presented experimental results of selecting 152 features using Android permission information, 16 features utilizing a genetic automated process selection of feature and evaluated performance with a decision tree, SVM,Naive Bayes. A genetic algorithm and simulated annealing were combined with a classification strategy to improve performance [20].

Lee [15] introduced a genetic algorithm and ensemble learning was used to develop SEdroid, a method of detecting Android malware. Wang[28] suggested an evolutionary algorithm-based solution for Android malware classification difficulties. There were no comparisons to normal feature selection trials provided by the authors. Fatima et al. [9] discovered that employing genetic algorithms lowers accuracy. This conclusion implies that more studies will be required to determine if detection systems based on genetic algorithms are more successful than other methods. Furthermore, since most research restricts the number of features to fewer than 200, it is vital to investigate whether employing a genetic algorithm on feature sets with a sufficient number of features affects how effectively they perform. Alzaylaee [5] described a technique for detecting Android malware using a deep CNN. Static analyzers were used to categorize the malware based on the raw opcode sequence from the disassembled Smali program. The benefit of this strategy is that it automatically learns the characteristics of malware. N-gram-based approaches influenced this study. The Android Malware Genome Project dataset was utilized for training the models [14]. The categorization system has a precision of 0.87 and a recall accuracy of 0.87. When the dynamic analysis is conducted, malware detection accuracy may be boosted.

III. ANALYSIS OF THE ANDROID MALWARE

A. The structure of the android application.

The extension "APK," which stands for "Android Package," is often used for Android application files. The APK file is a compressed zip file that executes programs on the Android operating system. After decompressing the APK file, one can observe the unique structure of the APK file, which connects together numerous files required to execute the software. The APK package's schematic structure is shown in Figure 2[21].

Manifest: This section includes fundamental information about the program and is found all APK files contain a binary XML file called AndroidManifest.xml under the name AndroidManifest.xml of crucial app data such as the component, device compatibility, application permissions, and package name.

i. Signatures: This component comprises the application signature with the META-INF directory.
ii. Assets: For real-time applications, assets are required to store static files, which are implemented as asset directories, and are accessed using the Asset Manager class.
iii. Compiled Resources: It includes information and resources that have been precompiled and saved as an arcs file. Resources in an application are located, accessible, and utilized with the help of this file, which keeps track of resource IDs and file files [30].

Figure 2: the Android application structure

iv. Native libraries: These are the libraries that are utilized in an app. The lib directory contains library files developed in C/C++ suited to the targeted device's CPU instruction set.
v. Dalvik bytecode: This component includes the byte code for JAVA. It is performed in the classes. dex file inside the application provides source code converted into bytecodes that Dalvik VMs may build. Each program has single DEX file by default, although some apps have many DEX files [16].
vi. Resources: This component contains the collection of resources that are utilized in a program. The Android application's resources are stored in the res directory.

The Manifest and Dalvik bytecode are the locations to check for when detecting Android malware. The Manifest is a crucial part of APK file where you may extract and evaluate information, including the app's version number, permissions, and performance [16]. Dalvik bytecode is a bytecode generated by the virtual environment that includes the application's primary code. The analyzer can examine which classes and objects the programs utilize and where hazardous codes are invoked by studying them using decompiled files. After all, such domains may be retrieved from the machine learning characteristics and directly impact the app's execution.

© Copyright: All rights reserved.

As a result, the Manifest and Dalvik bytecode sections are the focus of Android software analysis.

B. Static and dynamic analysis

Figure 3 below[22] shows the features of static , dynamic used for malware detection system, in which the model(static) consist of the resource features, features which are semantic, and dynamic models consist of the features which are behavioral. Static analysis is an automated way of reasoning about a program's runtime properties without actually running it [14]. It analyzes the program's source code and resources without actually running it. It is based on a dependence on source code analysis [6]. Static analysis is used to detect Android malware by AndroidManifest.xml, smali files, and a collection of static characteristics obtained by decompiling APK files, including permissions, Dalvik opcodes, API calls, and other components [11][34-37].

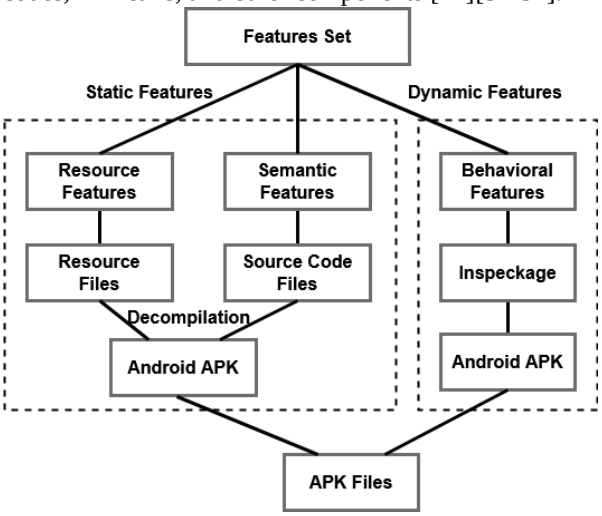

Figure 3: Static and Dynamic Analysis

It has the benefit of taking less time and using less computing power than dynamic analysis. Furthermore, independent of time, it can evaluate the complete code, allowing for effective processing, such as evaluating just the relevant bits [12]. However, there are certain drawbacks. It's hard to analyses the malware's static parts that can't be retrieved properly, and it's difficult to deal with harmful code that has been processed in a sophisticated manner due to obfuscations. Dynamic analysis is all about analyzing the attributes of a program running [15]. Data collector programs, crowdsourcing, and data collecting scripts are used to capture all of the user's input traces [10]. Program execution, API calls, internet traffic, and CPU statistics from Android apps are among the items [4]. Nonetheless, dynamic analysis is possible. Analysis and detection take longer than static processes and need a large number of resources. According to Lee [16], programs implementing a behaviour may only be utilized to identify malicious actions if they are performed during analysis, causing dynamic analysis to waste numerous CPU cycles by examining irrelevant areas of the application.

Static analysis vulnerabilities may be addressed by including dynamic analysis, while dynamic analysis' flaws can be complemented by static analysis. As a result, determining which of these two studies is superior is difficult, and suitable procedures must be used based on the detection objective and context [13]. Choosing features for machine learning should, of course, be taken into account while examining these aspects. In this case, the shortcomings in the dynamic analysis are exposed. During the dynamic analysis, the results of running the application directly during the analysis process are crucial. However, an extensive dataset can be difficult to run due to the difficulty of running all of the data. As a result, characteristics of ML algorithm are applied.

IV. MACHINE LEARNING TECHNIQUES

A. Feature selection

Selection of feature is a machine learning strategy that involves picking important characteristics to predict data and applying them for learning. Variable and feature selection has a number of possible advantages, including improving prediction performance by providing visualization tools and understanding, lowering measurement and storage needs, shortening training and usage duration, and overcoming the curse of dimensionality [16]. Feature extraction is not the same as feature selection, even though it has the same effect. Feature set development, measurement, and learning algorithms are the three primary processes of a filter approach. After generating a new set of characteristics, the information measurement is repeated until the termination criteria have been met. Unlike the filter approach, the wrapper method employs a black box that employs a learning algorithm to apply to selected features. Feature sets are entered into the black box and the most highly rated ones are created as a result of the learning algorithm [16]. The embedding approach selects features and is generally customized to the learning machine during the training process. Suppose selection of feature is properly implemented even on the dataset provided in this research. In that case, the machine learning performance will be enhanced since it uses fewer resources than learning the original dataset. According to Lee[15], feature selection considerably impacts malware detection experimental outcomes. It is consequently vital to choose an exceptional feature for increased efficacy. The connection between the chosen characteristics must be minimized [15]. A genetic algorithm was employed in this research to identify the best characteristics that best met the specified parameters. Figure 4 shows process of machine learning[14].

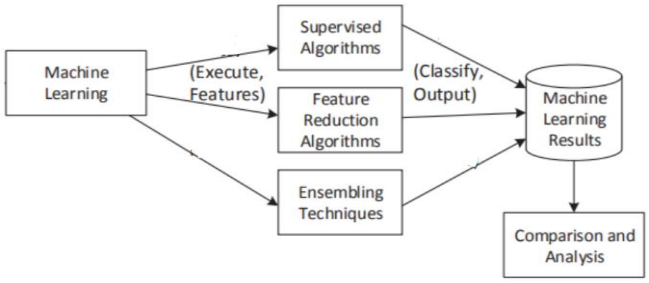

Figure 4: Machine Learning process

B. Genetic Algorithm.

Genetic algorithms cross over, mutate, and create new genes that diverge from existing ones. They use crossover and mutation techniques to identify the best solutions to develop things.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

As the generations pass, the answer determined by a genetic algorithm converges in specified ways. Due to the absence of established rules controlling it, the algorithm contains flaws that make it unable to provide the desired result. Furthermore, since the solution simply converges and does not advance, the method cannot guarantee the optimal solution, resulting in poorly optimized solutions. It does, however, have the benefit of assisting in the security and exploration of potentially large search regions, the discovery of optimum combinations, and the discovery of difficult-to-find answers. As a result, it's appropriate for solving NP-hard problems like feature selection [16]. As a result, using a genetic algorithm in this investigation was also suitable. Mutations with a certain probability may be formed once a crossover creates new genes. This step is continued until the algorithm's termination requirements are fulfilled.

C. Machine learning (ml) algorithms

ML is a technology that uses instances of data or experience to maximize performance criteria. Kalmegh [13] defined machine learning as an interdisciplinary discipline applied in various real-world applications. ML is accomplished by using a variety of algorithms to tackle issues in a variety of domains. As a result, different methods are used to assess the performance of selecting features. The ML techniques utilized in this work are described in detail below. 1) Supervised 2) Feature Reduction and 3) Ensembling Techniques. The algorithms which are supervised as below. RF,DT, KNN, ,Linear SVM,Logistic Regression, and N B[32-34]. The Feature Reduction Techniques used are Nonnegative Matrix Factorization (NMF),Linear Discriminant Analysis (LDA), and Principal Component Analysis (PCA) [4][31]. The few of algorithms are discussed below. Decision Trees: Typical decision trees are devised by using classification rules to evaluate and classify data. A training scenario guides a tree, where a tuple of data is conveyed, and the attribute's classifier is established. Due to a large amount of data, maps are typically based on greedy, top-down, and iterative processes. The tree is divided into sub-sets that satisfy the separating attribute's value using techniques that will optimally divide the training data [14]. While the algorithm is running, certain trees may be pruned or eliminated for more generic results. This is because the trees are easy to overfit.

a. **Random Forest**: Random forests use unconnected trees as part of their classification algorithm. A learning algorithm taught using the bagging approach identifies solutions while combining a broad range of input values [25]. This results in a large number of more accurate and reliable decision trees, which the algorithm can then combine to yield a higher level of accuracy.

b. **Naive Bayes**: This is a classification algorithm based on the Bayes theorem's probabilistic reasoning. The Naive Bayes standalone model determines and analyzing probabilities; a considerably higher likelihood suggests that the actual tag is more likely to be the higher probability's class label value [23]. The method assumes that the predictive qualities are class-wide events with no hidden as well as important latent variations in the prediction process. This makes it difficult to apply data that must be based on specific characteristics for a variety of classes.

c. **Multi-Layer Perceptron (MLP):** For learning, this technique employs bidirectional ANN [1]. The input data, the convolution layers, and the output layer are the three layers that make up the neural network architecture. The input nodes obtain the data, the hidden layer is produced using a non-linear activation, and the output unit displays the classification or regression results. Despite the fact that the model's output signals layers exist separately, the hidden layer may be layered as numerous layers. In contrast to a superficially stacked model, it is also recognized that the deep a prototype is, the more generic it may be [14]. Among the other classification approaches, the gradient boost classification model is efficient for android malware detection. Its pseudo-code model is illustrated below

D. Algorithm for gradient boost

Algorithm – Gradient Boost Classification

I. Purpose:To perform gradient boost classification.

II. Inputs: Input data $(a, b)_{i=1}^{N}$*, number of iterations S, loss function* $\psi(b, f)$*, and base learner model* $l(a, \theta)$*;*

III. Procedure:

IV. Output classified data

Step 1: Initialize the parameter $\widehat{x_0}$ *with the constant;*

Step 2: for $k = 1$ *to S do*

Step 3: Estimate the negative gradient $p_k(a)$ *by using the following equation:*

$$p_k(a) = E_y\left[\frac{\partial\psi(b, x(a))}{\partial x(a)}|a\right]_{x(a)=\hat{x}^{k-1}(a)}$$

Step 4: Estimate the new learner function $l(a, \theta_k)$*;*

Step 5: Identify the best gradient descent step g_k*;*

$$g_k = argmin_g \sum_{i=1}^{N} \psi[b_i, \hat{x}_{k-1}(a_i) + gl(a_i, \theta_k)]$$

Step 6: Update the function estimate by using the following model:

$$\widehat{g}_k \leftarrow \widehat{g}_{k-1} + g_k l(a, \theta_k)$$

Step 7: end for

E. Benefits and limitations of the machine learning models.

The table1 below presents various machine learning techniques and their benefits and limitations[7][8][10][11][29][31][38-40].

Table1.Machine Learning techniques

Sr. No.	ML Techniques	Benefits	Limitations
1	SVM	• Effectively solves problems of nonlinear classification	• Overhead is Increasing
2	K-means	• Easy	• Sensitivity to noise
3	KNN	• For solving multi-classification problems preferred • Increased efficiency	• Computational overhead
4	NN	• Increased Detection accuracy Fault tolerance	• Needs a complete data training • Requires selection of network topology

© Copyright: All rights reserved.

5	EL	• More accuracy • More processing speed	• Increased time consumption • Complexity of computation
6	DT	• Reduced cost • Handles multiclass problems	• Reduced accuracy
7	Random Forest	• Efficient • Reduction in overfitting	• Reduction in accuracy of Prediction
8	CNN	• More accuracy • Reduction in Error rate	• More time required for training of classifier • Complex computational operations
9	DNN	• Parallel processing support • Fault tolerance	• Difficult

V. THE DEEP LEARNING BASED TECHNIQUES

Deep learning algorithms may also be used to identify malware in the android smartphones. After completing a dynamic analysis, Android malware detection framework which was a web-based presented [19]. In this study, ML and DL algorithms were employed to identify malware with an overall detection rate of 98.8 per cent. The model suggested employing a semantic-based DL methodology to identify malware and create a tool named DeepRefiner [24]. Because the RNN involves a gradient vanish issue, this approach prefers the LSTM over the RNN. Using this method, malware could be identified with 97.4 percent accuracy and a 2.54 percent false positive rate. When compared to conventional methods, it was more efficient and accurate. Because this model employs a static analysis technique, several issues may arise. Using this model in conjunction with the hybrid analysis technique, these issues may be identified. The MOCDroid concept presented a evolutionary classifier which is multiobjective for malware detection in Android [19]. It used third-party call group behavior to construct a classifier utilizing multiobjective optimization and clustering. The accuracy of this approach was 95.15 per cent. This procedure includes three steps: import phrase extraction, clustering, and the use of a genetic algorithm. The decompress utility was used to extract the DEX files from the APK, and the JADX tool was used to extract the Java codes. The word matrix of the manuscript was then transformed. The genetic algorithm was utilized in the next phase, and K-Means clustering, which was regarded as the best accurate model for this.Other clustering approaches might be considered to increase the performance of the method [8]. The DL-Droid framework- is based on deep learning methods and suggests a novel method of identifying Android malware using dynamic analysis techniques [5]. In addition, this report included evaluations of detection accuracy and code coverage. The performance of traditional ML classifiers was also compared. DL, NB, J48, RF, SVM, PART, and SL were outperformed by this unique technique. In addition to this effort, investigating the idea of including an intrusion detection system into the DL-Droid would be a useful addition. The AdMat model is CNN Matrix-based solution for detecting Android malware. Apps were characterized and processed as pictures in this approach presented [27]. The adjacency matrix for applications was then built, and it was simplified with a size of 219*219 to improve data processing performance after converting decompiled code into the call-graph of the GML format. These matrices served as the CNN's input pictures, and the model was trained to distinguish between malicious and benign programs. This model has a 98.2 percent accuracy. Although the model is extremely accurate, there are several limits to this study, such as only doing static analysis; performance is dependent on the no. of characteristics included. For the evaluation of machine learning, deep learning algorithms the different types of measures are computed as below as shown in equations 1-6 as below.TP is True positive,FP is False positive,TN is True negative,FN is False Negative.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Sensitivity = \frac{TP}{TP+FN} \quad (2)$$

$$Specificity = \frac{TN}{TN+FP} \quad (3)$$

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

$$Recall\ or\ TPR = \frac{TP}{TP+FN} \quad (5)$$

$$F1-Score = \frac{2TP}{2TP+FP+FN} \quad (6)$$

VI. CONCLUSION

For Android OS users, detecting Android malware quickly and effectively is critical. Many studies have employed ML to identify malicious software, and selection of feature has been used to fasten the process. Experiments were carried out to pick authorization and API related information features for ML-based on this research's findings. The results demonstrated that evolutionary automated process feature selection was also helpful compared to a frequently used information gain. Despite the fact that the genetic algorithm's attribute selection performance was lowered by less than 3% in general, it still outperforms non-selection in terms of model generation time. ML & DLs ultimate objective is to provide location and time budgets and accuracy criteria to ML & DL systems, with the system determining an operational point that meets these requirements. This study revealed that evolutionary operators might assist ML/DL detect Android spyware, albeit further research and debate are required. If the additional study is done in the near future, using genetic algorithms as a technique to detect malware that is more accurate and takes less time will be highlighted.

REFERENCES

1. Abirami, S. and Chitra, P., 2020. Energy-efficient edge based real-time healthcare support system. In Advances in Computers (Vol. 117, No. 1, pp. 339-368). Elsevier.
2. Adebayo, O.S. and Aziz, N.A., 2019. The trend of mobile malwares and effective detection techniques. In Multigenerational Online Behavior and Media Use: Concepts, Methodologies, Tools, and Applications (pp. 668-682). IGI Global.
3. Ahmadi, M.; Sotgiu, A.; Giacinto, G. Intelliav: Toward the feasibility of building intelligent anti-malware on android devices. In Cross-Domain Conference for Machine Learning and Knowledge Extraction; Springer: Cham, Switzerland, 2017; pp. 137–154
4. Alam, S., Alharbi, S.A. and Yildirim, S., 2020. Mining nested flow of dominant APIs for detecting android malware. Computer Networks, 167, p.107026.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

5. Alzaylaee, M.K., Yerima, S.Y. and Sezer, S., 2020. DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, 89, p.101663.
6. Amro, B., 2017. Malware detection techniques for mobile devices. International Journal of Mobile Network Communications & Telematics (IJMNCT) Vol, 7.
7. D. Gupta, and R. Rani, "Improving malware detection using big data and ensemble learning," Computers & Electrical Engineering, vol. 86, pp. 106729, 2020.
8. Cunningham, P. and Delany, S.J., 2021. k-Nearest neighbour classifiers-A Tutorial. ACM Computing Surveys (CSUR), 54(6), pp.1-25.
9. Fatima, A., Maurya, R., Dutta, M.K., Burget, R. and Masek, J., 2019, July. Android malware detection using genetic algorithm based optimized feature selection and machine learning. In 2019 42nd International conference on telecommunications and signal processing (TSP) (pp. 220-223). IEEE.
10. Garg, S. and Baliyan, N., 2019. A novel parallel classifier scheme for vulnerability detection in android. Computers & Electrical Engineering, 77, pp.12-26.
11. Garg, S. and Baliyan, N., 2021. Android malware classification using ensemble classifiers. In Cloud Security (pp. 133-145). CRC Press.
12. Garg, S. and Baliyan, N., 2021. Comparative analysis of Android and iOS from security viewpoint. Computer Science Review, 40, p.100372.
13. Kalmegh, S.R., 2018. Comparative Analysis of the WEKA Classifiers Rules Conjunctiverule & Decisiontable on Indian News Dataset by Using Different Test Mode. International Journal of Engineering Science Invention (IJESI), 7(2Ver III), pp.2319-6734.
14. Kim, S., Yeom, S., Oh, H., Shin, D. and Shin, D., 2020. Automatic malicious code classification system through static analysis using machine learning. Symmetry, 13(1), p.35
15. Lee, J., Jang, H., Ha, S. and Yoon, Y., 2021. Android Malware Detection Using Machine Learning with Feature Selection Based on the Genetic Algorithm. Mathematics, 9(21), p.2813.
16. Li, L., Bissyandé, T.F., Papadakis, M., Rasthofer, S., Bartel, A., Octeau, D., Klein, J. and Traon, L., 2017. Static analysis of android apps: A systematic literature review. Information and Software Technology, 88, pp.67-95.
17. Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D. and Liu, H., 2020. A review of android malware detection approaches based on machine learning. IEEE Access, 8, pp.124579-124607.
18. Mahindru, A. and Singh, P., 2017, February. Dynamic permissions based android malware detection using machine learning techniques. In Proceedings of the 10th innovations in software engineering conference (pp. 202-210).
19. Martín, A., Menéndez, H.D. and Camacho, D., 2017. MOCDroid: multi-objective evolutionary classifier for Android malware detection. Soft Computing, 21(24), pp.7405-7415.
20. Meimandi, A., Seyfari, Y. and Lotfi, S., 2020, October. Android malware detection using feature selection with hybrid genetic algorithm and simulated annealing. In Proceedings of the 2020 IEEE 5th Conference on Technology In Electrical and Computer Engineering (ETECH 2020) Information and Communication Technology (ICT), Tehran, Iran (Vol. 22).
21. Merceedi, K.J., Ahmed, A.J., Salim, N.O., Hasan, S.S., Kak, S.F., Ibrahim, I.M., Yasin, H.M. and Salih, A.A., A State of Art Survey for Understanding Malware Detection Approaches in Android Operating System.
22. T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," Security and Communication Networks, vol. 2020,
23. M. Hossain, S. Rafi, and S. Hossain, "An Optimized Decision Tree Based Android Malware Detection Approach Using Machine Learning." pp. 115-125.
24. Rana, M.S.; Gudla, C.; Sung, A.H. Evaluating machine learning models for Android malware detection: A comparison study. In Proceedings of the 2018 VII International Conference on Network, Communication and Computing, Taipei City, Taiwan, 14–16 December 2018; pp. 17–21
25. Şahin, D.Ö., Kural, O.E., Akleylek, S. and Kılıç, E., 2021. A novel permission-based Android malware detection system using feature selection based on linear regression. Neural Computing and Applications, pp.1-16.
26. Mujumdar, Ashwini, Gayatri Masiwal, and B. B. Meshram. "Analysis of signature-based and behavior-based anti-malware approaches." International Journal of Advanced Research in Computer Engineering and Technology 2.6 (2013): 2037-2039.
27. Vu, L.N. and Jung, S., 2021. AdMat: A CNN-on-matrix approach to Android malware detection and classification. IEEE Access, 9, pp.39680-39694.
28. Wang, L., Gao, Y., Gao, S. and Yong, X., 2021. A New Feature Selection Method Based on a Self-Variant Genetic Algorithm Applied to Android Malware Detection. Symmetry, 13(7), p.1290.
29. S. A. Roseline, S. Geetha, S. Kadry, and Y. Nam, "Intelligent vision-based malware detection and classification using deep random forest paradigm," IEEE Access, vol. 8, pp. 206303-206324, 2020.
30. T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," Security and Communication Networks, vol. 2020, 2020.
31. Prerna Agrawal, Bhushan Trivedi. "Evaluating Machine Learning Classifiers to detect Android Malware", 2020 IEEE International Conference for Innovation in Technology (INOCON), 2020
32. P. Agrawal, and B. Trivedi, "Machine learning classifiers for android malware detection," Data Management, Analytics and Innovation, pp. 311-322: Springer, 2021.
33. M. Wadkar, F. Di Troia, and M. Stamp, "Detecting malware evolution using support vector machines," Expert Systems with Applications, vol. 143, pp. 113022, 2020
34. A. G. Kakisim, M. Nar, N. Carkaci, and I. Sogukpinar, "Analysis andevaluation of dynamic feature-based malware detection methods." pp. 247-258.
35. Damodaran, F. D. Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," Journal of Computer Virology and Hacking Techniques, vol. 13, no. 1, pp. 1-12, 2017.
36. I. Santos, J. Devesa, F. Brezo, J. Nieves, and P. G. Bringas, "Opem: A static-dynamic approach for machine-learning-based malware detection." pp. 271-280.
37. R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," Journal of Network and Computer Applications, vol. 36, no. 2, pp. 646-656, 2013.
38. Meenakshi Garg, Kiran Joshi "Machine learning approach for feature classification using supervised learning algorithms on "2011 in National Journal on Advances in Computing and Management,Volume 2,Issue1
39. Kailas K Devadkar, Meenakshi Garg, Kiran Joshi "Machine Learning Approach for Feature Selection using Naive Bayesian Variants "in 2nd International Conference on Information and Multimedia Technology,IEEE (ICIMT 2010),VI378-VI381
40. Meenakshi Garg, Kiran Joshi, Shubha Putharan "Classification Accuracy and Performance of Naïve Bayesian (NB), J48,ID3 and Decision Stump – Comparative Study " in Conference Recent Trends in Information Technology and Computer Science, December, 2011
41. https://gs.statcounter.com/os-market-share/mobile/worldwide
42. Chebyshev, V. Mobile Malware Evolution 2020. https://securelist.com/mobile-malware-evolution-2020/101029/
43. https://www.statista.com/statistics/680705/global-android-malware-volume/#statisticContainer

AUTHORS PROFILE

Prof. Kiran Joshi, pursuing his Ph.D.in Computer engineering from Veermata Jijabai Technological Institute, Mumbai, Maharashtra ,INDIA. He has completed Master of Technology from Government Engineering College ,Pune (COEP). He has completed Bachelor of Engineering (Computer Science and Engineering) from Government College of Engineering Aurangabad. His papers are also published in IEEE conference proceedings. His research areas are security,cloud computing ,machine learning. He is having 15 years of academic experience at esteemed institutes. Presently he is working as Assistant professor at Veermata Jijabai Technological institute in the computer engineering & information technology department, Mumbai. He has attended various workshops. He has guided students on patented product smarthelmet. He has won best Paper award for "Improved Authentication protocol for GSM Security" in National Conference ETCS-07.

© Copyright: All rights reserved.