Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 4, April : 2023

THE USE OF MACHINE LEARNING ALGORITHMS IN MALWARE DETECTION: A FRAMEWORK FOR REVERSE ENGINEERING ANDROID APPLICATIONS

[1]MEESALA PRANATHI,[2]H.MADHUSUDHANA RAO MCA,
[1]Student,[2]Assistant professor, MCA,M.Phill,(Ph.D)
Department of CSE

ABSTRACT:

Android has quickly become one of the most popular mobile operating systems. Because of this, Android has become a popular target for cybercriminals. Due to the increasing sophistication of malicious software buried in Android apps, detecting and recognising an application as malware has become the hardest task for security companies. Android malware has become more sophisticated, making it difficult to detect using traditional methods. The complexity and novelty of new Android threats have led to the rise of machine learning-based approaches. To perform their role, they first detect typical behaviour patterns shown by malware, then use this data to differentiate between recognised threats and those with unknown behaviour. In order to discover security flaws in mobile apps, the authors of this study use a combination of Machine Learning algorithms and attributes gleaned from reverse-engineered Android apps. Our help is dual in nature. We begin by suggesting a model that improves upon standard approaches by combining the most extensive available malware sample databases with novel static feature sets. Second, we have employed ensemble learning to boost our model's accuracy using a variety of machine learning methods (AdaBoost, SVM, etc.). The extracted malware from Android apps may be detected with a 96.24% success rate, using our testing results and discoveries, and a 0.3 FPR. The suggested model is trained using a single random feature acquired through reverse engineering, and it takes into account potentially harmful aspects like permissions, intents, API calls, etc.

Keywords: Android apps, safe, feature extraction, malware detection, reverse engineering, machine learning.

I.INTRODUCTION:

To this degree, it is guaranteed that mobile devices are an integral part of most people's daily lives. Furthermore, Android now controls the vast majority of mobile devices, with Android devices accounting for an average of 80% of the global market share over the past years. With the ongoing plan of Android to a growing range of smartphones and consumers around the world, malware targeting Android devices has increased as well. Since it is an open-source operating system, the level of danger it poses, with malware authors and programmers implementing unwanted permissions, features and application components in Android apps. The option to expand its capabilities with third-party software is also appealing, but this capability

comes with the risk of malicious device attacks. When the number of smartphone apps increases, so does the security problem with unnecessary access to different personal resources. As a result, the applications are becoming more insecure, and they are stealing personal information, SMS frauds, ransomware, etc. In contrast to static analysis methods such as a manual assessment of AndroidManifest.xml, source files and Dalvik Byte Code and the complex analysis of a managed environment to study the way it treats a program, Machine Learning includes learning the fundamental rules and habits of the positive and malicious settings of apps and then data-enabling. The static attributes derived from an application are extensively used in machine learning methodologies and the tedious task of this can be relieved if the static features of reverse-engineered Android Applications are extracted and use machine learning SVM algorithm, logistic progression, ensemble learning and other algorithms to help train the model for prediction of these malware applications [1]. Machine learning employs a range of methodologies for data classification. SVM (Support Vector Machine) is a strong learner that plots each data item as a point in n-dimensional space (where n denotes the number of features you have), with the value of each feature becoming the vector value. Then it executes classification by locating the hyper-plane that best distinguishes the two groups, leading to an improvement identification property for any two parameters. Conversely, boosting or ensemble techniques like Adaboost are assigned higher weights to rectify the behavior of misclassified variables in conjunction with other machine algorithms. When combined alongside weak classifiers, our preliminary model benefits from deploying such models since they have a high degree of precision or classification. [2], [3], [4], supports classifiers in their system models to find the highest accuracy. Although using ensemble or strong classifiers can cause problems like multicollinearity, which in a regression model, occurs when two or more independent variables are strongly associated with one another. In multivariate regression, this indicates that one regression analysis may be forecasted from another independent variable. This scope of the study can be presented as a detection journal analysis itself and can present several experimentations and results based on machine learning models [5], [6]. When an app has access to a resource in the most recent versions of Android OS, it must ask the OS for approval, and the OS will ask the user if they wish to grant or refuse the request via a pop-up menu. Many reports have been performed on the success of this resource management approach. The studies showed consumers made decisions by giving all requested access to the applications to their privileges requests [7]. In contrast to this, over 70% of Android mobile applications seek extra access that is not needed. They also sought a permit that is not needed for the app to run. A chess game that asks for photographs or requests for SMS and phone call permits, or loads unwanted packages are an example of an extra requested authorization. So, trying to assess an app's vindictiveness and not understanding the app is a tough challenge. As a result, successful malicious app monitoring will provide extra information to customers to assist them and defend them from information disclosure [8]. Figure 1 elaborates the android risk framework through the Google Play platform, which is then manually configured by the android device developers. FIGURE 1. Android

Security Framework Contrary to other smartphone formats, such as iOS, Android requires users to access apps from untrusted outlets like filesharing sites or third-party app stores. The malware virus problem has become so severe that 97 % of all Smartphone malware now targets Android phones. In a year, approximately 3.25 million new malware Android applications are discovered as the growth of smartphones increases. This loosely amounts to a new malware android version being introduced every few seconds [9]. The primary aim of mobile malware is to gain entrance to user data saved on the computer and user information used in confidential financial activities, such as banking. Infected file extensions, files received via Bluetooth, links to infected code in SMS, and MMS application links are all ways that mobile malware can propagate [10]. There are some strategies for locating apps that need additional features. Hopefully, by using these techniques, it would be possible to determine whether the applications that were flagged as questionable and needed additional authorization are malicious. Static analysis methodologies are the most fundamental of all approaches. Until operating programs, the permissions and source codes are examined [11]. For many machine learning tasks, such as enhancing predictive performance or simplifying complicated learning problems, ensemble learning is regarded as the most advanced method. It enhances a single model's prediction performance by training several models and combining their predictions. Boosting, bagging, and random forest are examples of common ensemble learning techniques [12].

In summary, the main contributions of our study are as follows:

1) We present a novel subset of features for static detection of Android malware, which consists of seven additional selected feature sets that are using around 56000 features from these categories. On a collection of more than 500k benign and malicious Android applications and the highest malware sample set than any state-of-the-art approach, we assess their stability. The results obtain a detection increase in accuracy to 96.24 % with 0.3% false-positives.

2) With the additional features, we have trained six classifier models or machine learning algorithms and also implemented a Boosting ensemble learning approach (AdaBoost) with a Decision Tree based on the binary classification to enhance our prediction rate.

3) Our model is trained on the latest and large time aware samples of malware collected within recent years including the latest Android API level than state-ofthe-art approaches.

This research paper incorporates binary vector mapping for classification by allocating 0 to malicious applications and 1 for non-harmful and for predictive analysis of each application fed to the model implemented in the study. The technique eases the process by reducing fault predictive errors. Figure 2 shows the procedure for a better understanding of the concept applied later in our study. The paper passes both the categories of applications through static analysis and then is further processed for feature extraction. We presented features in 0’s and 1’s after extraction. Matrix displays the extraction characteristics of each application used in the dataset.

There are major issues to be addressed to incorporate our strategy. High measurements of the features will make it difficult to identify malware in many real-world Android applications. Certain features overlap with innocuous apps and malware [13]. In comparison, the vast number of features will cause high throughput computing. Therefore, we can learn from the features directly derived from Android apps, the most popular and significant features. The paper implements prediction models and various computer ensemble teaching strategies to boost and enhance accuracy to resolve this problem [14]. Feature selection is an essential step in all machine-based learning approaches. The optimum collection of features will not only help boost the outcomes of tests but will also help to reduce the compass of most machine-based learning algorithms [15]. Studies have extensively suggested three separate methods for identifying android malware: static, interactive meaning dynamically, and synthetic or hybrid. Static analysis techniques look at the code without ever running it, so they're a little sluggish if carried out manually and have to face a lot of false positives [16]. Data obfuscation and complex code loading are both significant pitfalls of the technique. That is why automated operation helps to achieve reliability, accuracy, and lesser time utilization [17]. Reverse engineer Android applications and extract features and do static analysis from them without having to execute them. This method entails examining the contents of two files: AndroidManifest.xml and classes.dex, and working on the file with the .apk extension. Feature selection techniques and classification algorithms are two crucial areas of featurebased types of fraudulent applications. Feature filtering methods are used to reduce the dimension size of a dataset. Any of the functions (attributes) that aren't helpful in the study are omitted from the data collection because of this. The remaining features are chosen by weighing the representational strength of all the dataset's features [18]. Parsing tools can help learn which permissions, packages or services an application offers by analyzing the AndroidManifest.xml file, such as permission android.permission.call phone, which allows an application to misuse calling abilities. The paper elaborates exactly what sort of sensitive API the authors could name by decoding the classes.dex file with the Jadx-gui disassembler [19]. In certain cases, including two permissions in a single app can signify the app's possible malicious attacks. For example, an application with RECEIVE SMS and WRITE SMS permissions can mask or interfere with receiving text messages [20] or applying sensitive API such as sendTextMessage() can also be harmful and lead to fraud and stealing. Until we started our main idea of the project. The fact explained that Android applications pose a lot of threats to its user because of the unnecessary programs compiled inside them and explained why it is necessary to automate the process of static analysis for the efficient detection of malware applications based on the extracted features. The rest of the paper is planned as follows. Related works are examined in Section II. Section III will present the design and method of our model. Section IV elaborates the assessment findings and future threats. The experiments and results will be dilated and performed in Sections V and VI. Section VII includes our research issues, recommendations, and conclusions for the future.

II.LITERATURE SURVEY:

Linux (Android core) keeps key aspects of the security infrastructure of the operating system. The Android displays to the administrator a list of features, sought to reinstall an update. The program installs itself on the computer after they issue access. Figure 3 shows the integrated core parts of Android architecture. It comprises applications at the top layer and also includes an application framework, libraries or a Runtime layer, and a Linux kernel. These levels are further divided into their components, which make an Android Application. The Linux Kernel is the key part of Android that provides its OS functionality to phones, and the Dalvik Virtual Machine (DVM) is to manage a mobile device. Application is the Android architecture's highest layer. Native and third-party apps such as contacts, email, audio, gallery, clock, sports, and so on are located only in this layer. This framework gets the classes often used to develop Android apps. It also handles the user interface and device infrastructure and provides a common specification for hardware entry. To facilitate the development of Android, the Platform Libraries include many C/C++ core libraries and Java-based libraries such as SSL, libc, Graphics, SQLite, Webkit, Media, Surface Manager, OpenGL, and others. The taxonomy helps understand the viewer with a logical algorithmic approach for grasping the core surfaces and functionality of the operating system.

The methods proposed in this related work contribute to key aspects and a higher predictive rate for malware detection. Certain research has focused on increasing accuracy, while others have focused on providing a larger dataset, some have been implemented by employing various feature sets, and many studies have combined all of these to improve detection rate efficiency. In [21] the authors offer a system for detecting Android malware apps to aid in the organization of the Android Market. The proposed framework aims to provide a machine learning-based malware detection system for Android to detect malware apps and improve phone users' safety and privacy. This system monitors different permission-based characteristics and events acquired from Android apps and examines these features employing machine learning classifiers to determine if the program is goodware or malicious. The paper uses two datasets with collectively 700 malware samples and 160 features. Both datasets achieved approximately 91% accuracy with Random Forest (RF) Algorithm. [22] Examines 5,560 malware samples, detecting 94 % of the malware with minimal false alarms, where the reasons supplied for each detection disclose key features of the identified malware. Another technique [23] exceeds both static and dynamic methods that rely on system calls in terms of resilience. Researchers demonstrated the consistency of the model in attaining maximum classification performance and better accuracy compared to two state-of-the-art peer methods that represent both static and dynamic methodologies over for nine years through three interrelated assessments with satisfactory malware samples from different sources. Model continuously achieved 97% F1- measure accuracy for identifying applications or categorizing malware. [24] The authors present a unique Android malware detection approach dubbed Permissionbased Malware Detection Systems

(PMDS) based on a study of 2950 samples of benign and malicious Android applications. In PMDS, requested permissions are viewed as behavioral markers, and a machine learning model is built on those indicators to detect new potentially dangerous behavior in unknown apps depending on the mix of rights they require. PMDS identifies more than 92–94% of all heretofore unknown malware, with a false positive rate of 1.52–3.93%. The authors of this article [25] solely use the machine learning ensemble learning method Random Forest supervised classifier on Android feature malware samples with 42 features respectively. Their objective was to assess Random Forest's accuracy in identifying Android application activity as harmful or benign. Dataset 1 is built on 1330 malicious apk samples and 407 benign ones seen by the author. This is based on the collection of feature vectors for each application. Based on an ensemble learning approach, Congyi proposes a concept in [26] for recognizing and distinguishing Android malware. To begin, a static analysis of the Android Manifest file in the APK is done to extract system characteristics such as permission calls, component calls, and intents. Then, to detect malicious apps, they deploy the XGBoost technique, which is an implementation of ensemble learning. Analyzing more than 6,000 Android apps on the Kaggle platform provided the initial data for this experiment. They tested both benign and malicious apps based on 3 feature sets for a testing set of 2,000 samples and used the remaining data to create a training set of 6,315 samples. Additional approaches include [27], an SVM-based malware detection technique for the Android platform that incorporates both dangerous permission combinations and susceptible API calls as elements in the SVM algorithm. The dataset includes 400 Android applications, which included 200 benign apps from the official Android market and 200 malicious apps from the Drebin dataset. [28] Determines whether the program is dangerous and, if so, categorizes it as part of a malware family. They obtain up to 99.82 % accuracy with zero false positives for malware detection at a fraction of the computation power of state-of-the-art methods but incorporate a minimal feature set. The results of [29] demonstrate that deep learning is adequate for classifying Android malware and that it is much more successful when additional training data is available. A permission-based strategy for identifying malware in Android applications is described in [30], which uses filter feature selection algorithms to pick features and implements machine learning algorithms such as Random Forest, SVM, and J48 to classify applications as malware or benign. This research [31] provides a feature selection using the Genetic algorithm (GA) approach for identifying Android malware. For identifying and analyzing Android malware, three alternative classifier techniques with distinct feature subsets were built and compared using GA.

III.CONCLUSION

In this research, we devised a framework that can detect malicious Android applications. The proposed technique takes into account various elements of machine learning and achieves a 96.24% in identifying malicious Android applications. We first define and pick functions to capture and analyze Android apps' behavior, leveraging reverse application engineering and

AndroGuard to extract features into binary vectors and then use python build modules and split shuffle functions to train the model with benign and malicious datasets. Our experimental findings show that our suggested model has a false positive rate of 0.3 with 96% accuracy in the given environment with an enhanced and larger feature and sample sets. The study also discovered that when dealing with classifications and high-dimensional data, ensemble and strong learner algorithms perform comparatively better. The suggested approach is restricted in terms of static analysis, lacks sustainability concerns, and fails to address a key multicollinearity barrier. In the future, we'll consider model resilience in terms of enhanced and dynamic features. The issue of dependent variables or high intercorrelation between machine algorithms before employing them is also a promising field.

REFERENCES

[1] A. O. Christiana, B. A. Gyunka, and A. Noah, “Android Malware Detection through Machine Learning Techniques: A Review,” Int. J. Online Biomed. Eng. IJOE, vol. 16, no. 02, p. 14, Feb. 2020, doi: 10.3991/ijoe.v16i02.11549. [2] D. Ghimire and J. Lee, “Geometric Feature-Based Facial Expression Recognition in Image Sequences Using Multi-Class AdaBoost and Support Vector Machines,” Sensors, vol. 13, no. 6, pp. 7714–7734, Jun. 2013, doi: 10.3390/s130607714. [3] R. Wang, “AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review,” Phys. Procedia, vol. 25, pp. 800–807, 2012, doi: 10.1016/j.phpro.2012.03.160. [4] J. Sun, H. Fujita, P. Chen, and H. Li, “Dynamic financial distress prediction with concept drift based on time weighting combined with Adaboost support vector machine ensemble,” Knowl.-Based Syst., vol. 120, pp. 4–14, Mar. 2017, doi: 10.1016/j.knosys.2016.12.019. [5] A. Garg and K. Tai, “Comparison of statistical and machine learning methods in modelling of data with multicollinearity,” Int. J. Model. Identif. Control, vol. 18, no. 4, p. 295, 2013, doi: 10.1504/IJMIC.2013.053535. [6] C. P. Obite, N. P. Olewuezi, G. U. Ugwuanyim, and D. C. Bartholomew, “Multicollinearity Effect in Regression Analysis: A Feed Forward Artificial Neural Network Approach,” Asian J. Probab. Stat., pp. 22–33, Jan. 2020, doi: 10.9734/ajpas/2020/v6i130151. [7] W. Wang et al., “Constructing Features for Detecting Android Malicious Applications: Issues, Taxonomy and Directions,” IEEE Access, vol. 7, pp. 67602–67631, 2019, doi: 10.1109/ACCESS.2019.2918139. [8] B. Rashidi, C. Fung, and E. Bertino, “Android malicious application detection using support vector machine and active learning,” in 2017 13th International Conference on Network and Service Management (CNSM), Tokyo, Nov. 2017, pp. 1–9. doi: 10.23919/CNSM.2017.8256035. [9] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, “Significant Permission Identification for Machine-Learning-Based Android Malware Detection,” IEEE Trans. Ind. Inform., vol. 14, no. 7, pp. 3216–3225, Jul. 2018, doi: 10.1109/TII.2017.2789219. [10] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and J. Blasco, “Dendroid: A text mining approach to analyzing and classifying code structures in

Android malware families," Expert Syst. Appl., vol. 41, no. 4, pp. 1104–1117, Mar. 2014, doi: 10.1016/j.eswa.2013.07.106.