MACHINE LEARNING TECHNIQUE FOR DETECTION OF MALWARE APPLICATION

[1] Ms. A. Sudhavali, M.Tech, Assistant Professor, Department of CSE, Eluru College of Engineering And Technology, Duggirala, Andhra Pradesh-534004.
[2] P. Sai, B.Tech, Department of CSE, Eluru College of Engineering And Technology, Duggirala, Andhra Pradesh-534004.
[3] T. Roja Mythili, B.Tech, Department of CSE, Eluru College of Engineering And Technology, Duggirala, Andhra Pradesh-534004.
[4] K. Jayanth, B.Tech, Department of CSE, Eluru College of Engineering And Technology, Duggirala, Andhra Pradesh-534004.
[5] P. John Wesly, B.Tech, Department of CSE, Eluru College of Engineering And Technology, Duggirala, Andhra Pradesh-534004.

Abstract: Malicious Application growth has been increasing dramatically along with increasing the diversity and complicity of their developing techniques. Machine learning techniques are the current methods to model patterns of static features and dynamic behaviors of Android malware. Whereas the accuracy rates of the deep learning classifiers increase with increasing the quality of the features, we relate between the apps' features and the features that are needed to deliver its category's functionality. Differently, our classification approach defines legitimate static features for benign apps under a specific category as opposite to identifying malicious patterns. We utilize the features of th e to p- ra te d apps in a specific category to train a malware detection classifier for that given category. Android apps stores organize apps into different categories, for instance, 26 categories on Google Play Store. Each category has its distinct functionalities which means the apps under a specific category are similar in their static and dynamic features. In general, benign apps under a certain category tend to share a common set of features. On the contrary, malicious apps tend to request abnormal features, less or more than what is common for the category that they belong to. This study proposes category- based machine learning classifiers to enhance the performance of classification models at detecting malicious apps under a certain category. The intensive deep learning experiments proved that category- based classifiers report a remarkable higher average performance compared to non- category based.

1. INTRODUCTION

MALWARE is defined as software designed to infiltrate or damage a computer system without the owner's informed consent. Malware is actually a generic definition for all kind of computer threats. A simple classification of malware consists of file infectors and stand-alone malware. Another way of classifying malware is based on their particular action: worms, backdoors, trojans, root kits, spyware, adware etc. Malware detection through standard, signature based methods [1] is getting more and more difficult since all current malware applications tend to have multiple polymorphic layers to avoid detection or to use side mechanisms to automatically update themselves to a newer version at short periods of time in order to avoid detection by any antivirus software. For an example of dynamical file analysis for malware detection, via emulation in a virtual environment, the interested reader can see [2]. Classical methods for the detection of metamorphic viruses are described in [3]. An overview on different machine learning methods that were proposed for malware detection is given in [4]. Here we give a few references to exemplify such methods. − In [5], boosted decision trees working on n-grams are found to produce better results than both the Naive Bayes classifier and Support Vector Machines. − [6] uses automatic extraction of association rules on Windows API execution sequences to distinguish between malware and clean program files. Also using association rules, but on honey tokens of known parameters, is [7]. − In [8] Hidden Markov Models are used to detect whether a given program file is (or is not) a variant of a previous program file. To reach a similar goal, [9] employs Profile Hidden Markov Models, which have been previously used with great success for sequence analysis in bioinformatics. − The capacity of neural networks to detect polymorphic malware is explored in [10]. In [11], Self-Organizing Maps are used to identify patterns of behavior for viruses in Windows executable files. In this paper, we present a framework for malware detection aiming to get as few false positives as possible, by using a simple and a simple multi-stage combination (cascade) of different versions of the perception algorithm [12]. Other automate classification algorithms [13] could also be used in this framework, but we do not explore here this alternative. The main steps performed through this framework are sketched as follows: 1. A set of features is computed for every binary file in the training or test datasets (presented in Section II), based on many possible ways of analyzing a malware. 2. A machine learning system based firstly on one-sided perceptions, and then on feature mapped one-sided perceptions and a kernelized one-sided perceptions (Section III), combined with feature selection based on the F1 and F2 scores, is trained on a medium-size dataset consisting of clean and malware files. Cross-validation is then performed in order to choose the

right values for parameters. Finally, tests are performed on another, non-related dataset. The obtained results (see Section IV) were very encouraging. 3. In the end (Section V) we will analyse different aspects involved in the scale-up of our framework to identifying malware files on very large training datasets.

2. LITERATURE SURVEY

2.1 "Android Malware Detection API Calls Analysis" Authors: Zhou et al. (2012)using Permission This study focused on analyzing the permissions and API calls used by Android apps to detect malicious behavior. Machine learning techniques were employed for classification. .

2.2 "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket" Authors: Arp et al. (2014): This introduced DREBIN, a machine learning-based system for detecting Android malware. It utilized features extracted from Android apps to classify them as either malicious or benign.

2.3 "combination of features from both static and dynamic analysis" Author:Waneta. (2017) used a combination of features from both static and dynamic analysis, then apply PCA to reduce the dimensionality of data and use SVM to perform the classification of applications into benign and malware classes.

2.4 "Indication of new malware" Authors: Li et al. (2018) In research that was conducted by Li et al. (2018), the authors indicated that new malicious Android applications are introduced into the mobile ecosystem every ten seconds.

2.5 "scalable malware detection method" Authors:Kyaw and Kham (2019) Kyaw and Kham (2019) reiterate that a scalable malware detection method that yields optimum results is the use of permission analysis through the Sapid approach. However, it has led to more challenges like cyber-attacks.

2.6 "signature-based systems used for malware detection have become inefficient in detecting advanced malware applications" Authors: Christiana et al., 2020 Among the challenges posed by android devices and the mobile ecosystem as a whole, malicious applications have undoubtedly taken the lead (Christiana et al., 2020). For instance, signature-based systems used for malware detection have become inefficient in detecting advanced malware applications (Christiana et al., 2020).

2.7 "Android operating system has been advancing in enhancing its robustness." Authors: Syrtis and Generativist (2021 As a result, machine learning techniques are now at the top in dealing with this challenge. In a study conducted by Syrtis and Generativist (2021), the authors start by appreciating how much the Android operating system has, over the years, been advancing in enhancing its robustness.

2.8 "Android applications are not approved by the associated organizations" Authors:(Singh et al., 2022) The robustness is associated with the advanced technologies, significant community support, and availability of tons of resources on the internet. (Singh et al., 2022). Consequently, some Android applications are not approved by the associated organizations and might be misused in collecting user data that might eventually be misused.

3. EXISTING SYSTEM

One existing work has used data processing and options generated from windows workable API calls. They achieved sensible leads to a really giant scale dataset with concerning 35,000 transportable workable files. Another activity foot printing methodology additionally provides a dynamic approach to discover self-propagating malware. All these existing ways have basically advanced the mechanical man malware detection; however, the misuse detection isn't reconciling to the novel mechanical man malware and continually needs frequent change of the signatures. Here lies the analysis gap. In exiting system, they implemented the classifiers like naive bayes and decision tree which gives the poor accuracy.

DISADVANTAGES

Misuse detection isn't reconciling accuracy is less. Mechanical man malware detection.

4. PROPOSED SYSTEM

In the proposed system we implement a better feature extraction technique and then we apply CNN Algorithm for feature extraction and classify Malicious Application detection which gives the better accuracy ratio when compare to existing system.

ADVANTAGES

Easy to identify and block malware. Accuracy is more.

Dynamic feature extraction using genetic algorithm.

SYSTEM ARCHITECTURE

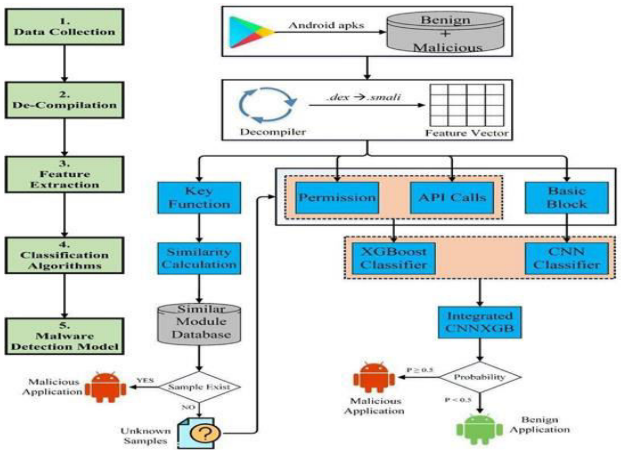

Fig1: System Architecture

5. UML DIAGRAMS

1. CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the

only UML diagrams, which can be mapped directly with object-oriented languages. It is also known as a structural diagram. Class diagram contains • Classes • Interfaces • Dependency, generalization and association.

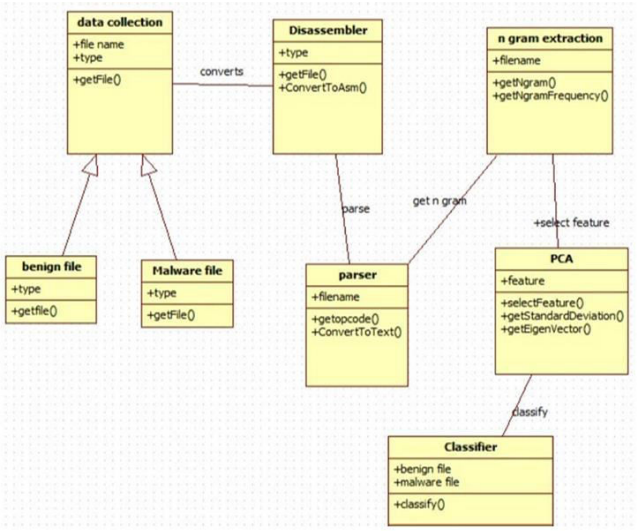

Fig 5.1 shows the class diagram of the project

2. **USECASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted

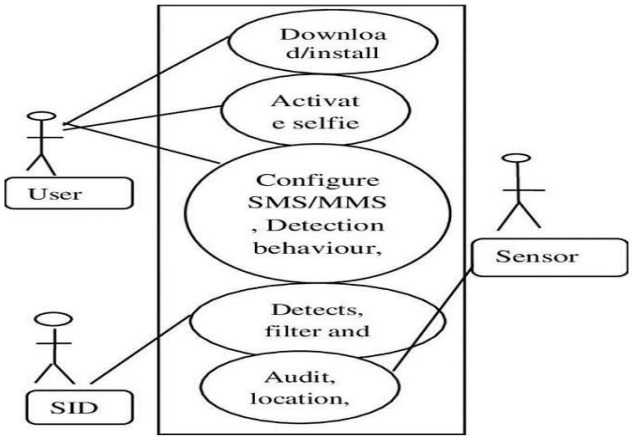

Fig 5.2 shows the Use case Diagram

3. SEQUENCE DIAGRAM:

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. Sequence diagrams are used to formalize the behavior of the system and to visualize the communication among objects. These are useful for identifying additional objects that participate in the use cases. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

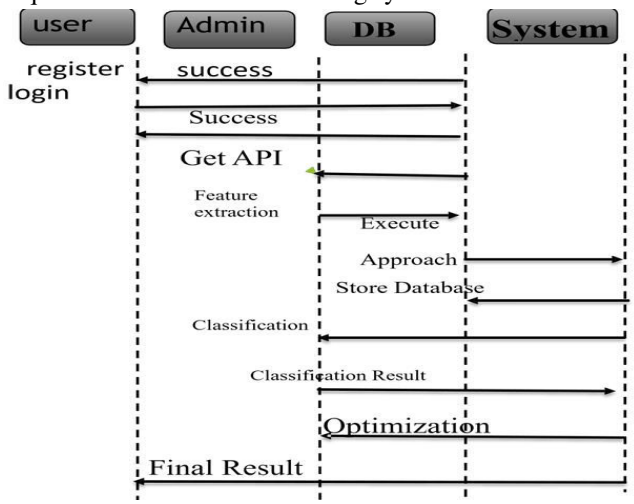

Fig 5.3 Shows the Sequence Diagram

6. RESULTS

6.1 Output Screens

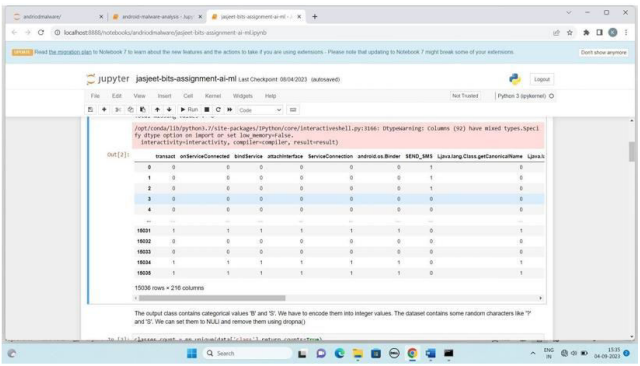

Fig 6.1 Dataset File

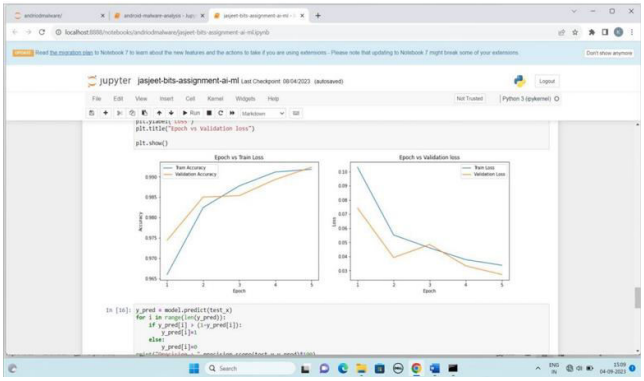

Fig 6.2 Performance Graph

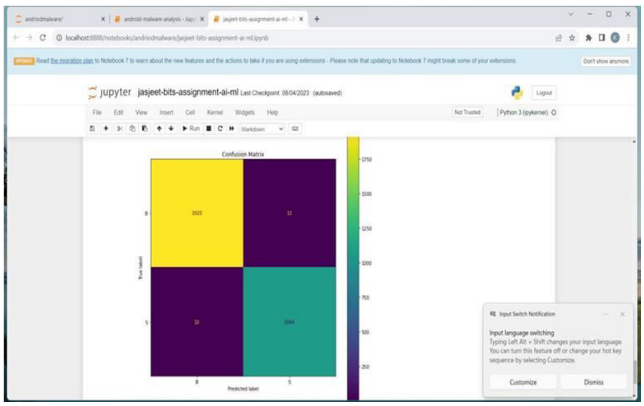

Fig 6.Confusion Matrix

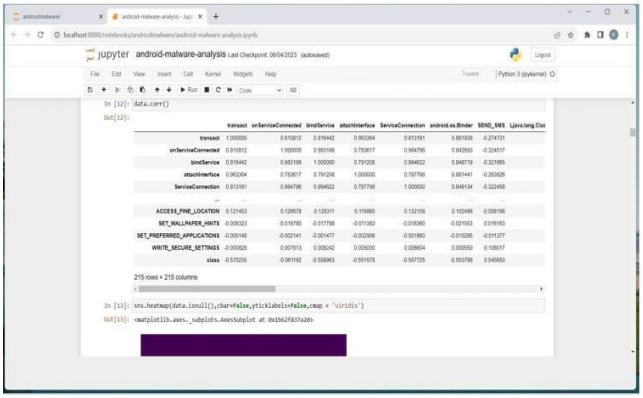

Fig 6.4 Performance Analysis

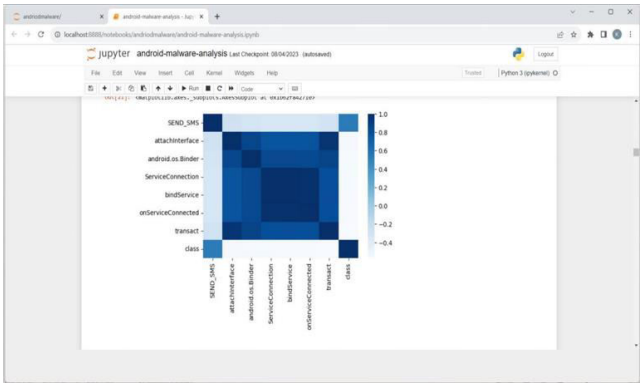

Fig 6.5 Accuracy Comparison Graph

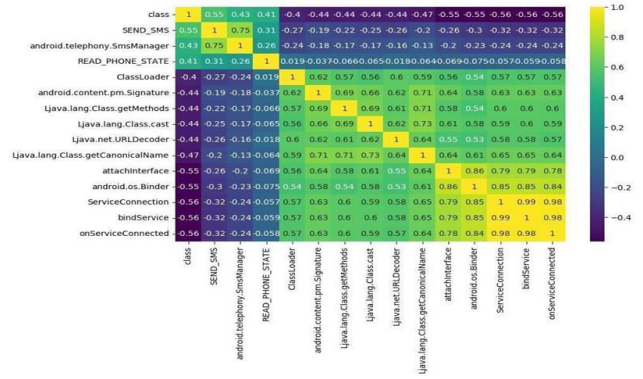

Fig 6.6 Malware Analysis.

7. CONCLUSION

In our study, we propose category-based deep learning classifiers to improve the performance of the classification models. In static analysis of Android malware, machine learning algorithms have been used to train classifiers with features of malicious apps to build models that capable of detecting malicious patterns. Differently, our classification approach defines legitimate static features for benign apps as opposite to identifying malicious patterns. We utilize the features of the top-rated apps in a specific category to define a profile of the common sets of features for that category. In other words, to detect whether or not the app possesses the characteristics of benign, we relate between the app's features and the features that are needed to deliver the category's functionality that the app belongs to. Android stores organize apps into different categories; 26 categories on the Google Play Store. For example: In each category, the apps deliver a similar functionality as a result the they tend to request a common set of features like same permissions, APIs, hardware components, broadcast receivers, intents filters, etc. On the contrary, malicious apps tend to have abnormal features, less or more than what is common for the category that they belong to. Malicious apps can be identified by comparing between the features they request to the features that are requested by benign apps in the same category. For example: malicious apps, compared to the benign apps in the same category, tend to request over-privileged permissions, listen specific events that broadcast by the Android system, or using unneeded APIs for the app's category functionality that can be used to lunch malicious behaviors

FUTURE SCOPE

Our future work will consider three aspects. First, including other static features such as: functions call in building the classification models to get a better understanding of the processes that apps may lunch in a way to increase the detection accuracy of the classifiers. Second, implementing the proposed solution on a large scale level by building profile models for other categories and sub categories. Third, testing the feasibility of integrating our solution with dynamic detection techniques by profiling dynamic features for each category; dynamic features like system calls, network connections, resources usage, and etc.

8. REFERENCES

1. Androguard-usage https: // code . google .com/p/androguard/wiki/Usage.Ac-cussed April 24, 2015.

2. Android statistics & facts Statista. http : // www.statista.com/topics/876/ android/. Accessed April 19, 2015.
3. Android and iOS continue to dominate the worldwide smart phone market with android shipments justshy of 800 million in 2013,accordingtodice.http://www idc.com / getdoc.jsp?containerId=prUS24676414.Accessed April 19, 2015.
4. Applicationfundamentals—androiddevelopers. 5.http://developer.android.com/guide/components/fundamentals.html. Accessed April 19, 2015.
6.redownload/installationhttps://redmine.honeynet.org/projects/are/ wiki. Accessed April 28, 2015. 7. Dynamic analysis tools for android fail to detect malware with heuristic evasio n techniques .http://thehackernews.com/2014/05/dynamic-analysis-tools for-android-fail.html. Accessed April 19, 2015.
8. ” Global smartphone sales exceed 1.2b units in 2014. ” gfk - we see the big picture. http://www.gfk.com/news-and-events/press-room/press releases/pages/global-smartphone-sales-exceed-12b-units-in-2014.aspx.
Accessed April 19, 2015.
9.Google:Wehavebillionmonthlyactiveandroidusershttp://www.businessinsider. com/google-we-have-1billion-monthly-active-android-users-2014-6. Accessed April 19, 2015.
10. Report: 97- Forbes. http://www.forbes.com/sites/gordonkelly/2014/03/24/
11. report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/.Accessed April 19, 2015.
12.Smartphoneosmarketshare,q42014http://www.idc.com/prodserv/ smartphone-os-market-share ’ s. Accessed April 19, 2015. Arp, D., Spreitzenbarth, M., Hu¨bner, M., Gascon, H., Rieck, K., and Siemens, C. (2014).