Malicious Application Detection in Android using Machine Learning

Pooja Thorat*1, Manesh Rathod*2, Santosh Shinde*3

**1,2 Student, Department of Information Technology Engineering,*
Sinhgad College Of Engineering Pune, Maharashtra, India.
**3Assistent Professor, Department of Information Technology Engineering,*
Sinhgad College Of Engineering Pune, Maharashtra, India.

Abstract

Due to the Android platform's recent surge in popularity, the number of malicious Android applications has increased quickly. We are more vulnerable to mobile viruses and fraudulent apps since our daily lives have become more dependent on the use of mobile applications like games, emails, and social networking sites. This paper examines the problem of identifying fraudulent apps on mobile internet, which is essential for safeguarding user privacy and private information. By transforming the Android internet dangerous application identification problem into a classification problem, we may utilize the SVM classifier to solve it. We conclude by testing the effectiveness of the suggested approach. Findings from experiments that the suggested can recognize.

Keywords: Malware, Data collection, feature extraction

INTRODUCTION

The simplicity of using smartphones has led to a rapid increase in the number of smartphone apps. Its effectiveness across a range of applications as well as the ongoing software and hardware development for smart devices. Global smartphone usage is projected to reach 4.3 billion users by 2023. Android is the most widely used mobile operating system. (OS) software. As of May 2021, it held a 72.2% market share. With a market share of 26.99%, Apple iOS is the second-largest vendor, while the remaining 0.81% is shared by Samsung, kaiOS, and other niche vendors. Google Play is the official app store for Android-powered mobile devices. It had more than 2.9 million published apps as of May 2021. They consist of over 2.5 million apps. However, if a smart phone is lost, there will be a significant threat to people's privacy. As a result, security concerns are becoming increasingly serious, making it essential to recognize risky mobile Internet applications.

Android has a market share that is influencing an increase in the volume of data being gathered from various consumers, with over one billion active users across all of their mobile devices. Due to these factors, cybercriminals have created malware in an effort to stop the problems that malware produces. Android employs a variety of security precautions, such as a unique user ID for each program, system permissions, and its method of distribution, Google Play. We use information obtained from Android phone data to locate hazardous applications. An Android phone's data is used as input. We classify the data set after preprocessing and segmenting it in order to discover malicious programs set and apply the SVM technique to detect malicious applications.

RELATED WORK

(1) Peng Tian and Xiaojun Huang, "A Malicious Application Detection Model to Remove the Influence of Interference API Sequence" The novel model for detecting malicious Android applications is suggested in this study. The model acquires the APP runtime's API call sequences and extracts features from them. These features exhibit the lowest levels of redundancy among themselves, have the highest correlation with the detection of malicious attributes, and it has been observed that the training of the detector can be hindered by API subsequences produced by legitimate behavior that may be present in malicious applications. To get rid of this interference and increase the detection accuracy, we integrate the GBDT algorithm with VSM and K-means. Experiments reveal that by using this technique, interference from the API

sequence may be successfully eliminated, resulting in greater detection accuracy.

2. Fei Chen, Yan Fu's paper "Dynamic Detection of Unknown Malicious Executables Based on API Interception" In this research, we suggest a novel method for the dynamic identification of malicious executables running on the Windows platform. Our method uses an API (Application Program Interface) interception mechanism to extract behavioural signatures from malicious executables, allowing the detection of previously undiscovered harmful executables. Obtaining the executable's sequence of API function calls, processing the sequence to produce a vector, and comparing the vector to the feature library created by security policies to determine whether the executable is malicious are the three main steps in the dynamic detection of unknown malicious executables. The experiment shows that this method works well at identifying unknown harmful executables.

3. "Study on the Application of Dalvik Injection Technique for the Detection of Malicious Programs in Android" by Yingbo Li, Jing Fang, and Cheng Liu. Malicious software that targets smartphones is continuously emerging as smart phones become more and more commonplace. Android is facing a significant security challenge because it now has the largest market share among phone operating systems. The analysis of the Dalvik injection technique's employment in the discovery of Android malware is the main subject of this article. It is possible to directly detect the applications on an Android phone by altering the system API (Application Program Interface) using the Dalvik injection approach. Determine whether the target program is malicious or not by looking through the list of sensitive APIs that malicious programs call. The research is done on static analysis methods. Also outlined were the tools that may be used to analyze Android code using static analysis methods. Fundamental analysis techniques include abstract representation, taint analysis, symbolic execution, program slicing, code instrumentation, and type/model checking. Although the most popular method for detecting privacy and security-related problems was correctly identified by this research, the usefulness of static analysis approaches for malware detection was not covered. In addition, it did not take into consideration current research that recommended fresh ways for malware identification and analysis. A thorough systematic assessment of static analysis methods that can be utilised to find Android malware was provided by the work published in [35]. Characteristic-based, opcode-based, program-graph-based, and symbolic execution-based techniques were recognized as the four approaches. Then, using the body of current literature, it assessed the performance of static analysis-based Android malware detection approaches based on those four methods. The article has suggested ML and statistical models as potential approaches for identifying Android malware.

SYSTEM ARCHITECTURE

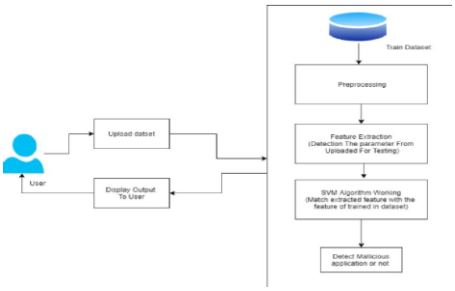


Fig: - System Architecture

Malware assaults are the most frequent event that might be categorized as a threat to Android. Numerous researchers have defined malware differently depending on the harm they cause. Any malicious program having a component of malicious code that aims to acquire unauthorized access and participate in unethical or illegal behaviour while breaching the three fundamental security principles of confidentiality, integrity, and availability is defined as malware in its broadest sense.

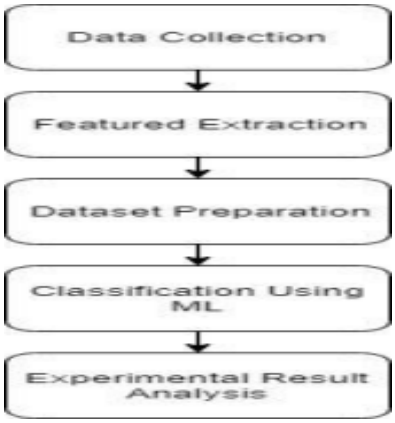
In first step the administrator must log in with a proper username and password. View and authorize users in step two. Here, the administrator may see the user's name, email address, and residential address as well as other contact information. In the third step view the Charts' Results. This involves looking at every product's search ratio, every keyword's search results, and every product's review ranking.

In the fourth step, E-commerce User- There are currently n users. Before doing any operations, the user should register. Once a user registers, the database will record their information.

In the fifth step, the end user, there are n users present. Before doing any operations, the user should register. After a user registers, the database will have access to their information.

METHODOLOGY

This section describes how Android applications are classified as harmful or benign based on permission analysis.



A) Data Collection

Numerous Android application samples—both dangerous and beneficial were downloaded. 1500 excellent Android apps came from the Google Play store and other trustworthy sources, whereas 2500 harmful ones were acquired from a range of websites, including VirusShare, zeltser, etc.

(B) Feature Extraction

Malware, permissions, and the many kinds of permissions that a program needs in order to operate were the first things we learned about. Android permissions are divided into many levels of security:

1. Normal Permissions are those that do not significantly compromise the user's privacy. The Android operating system directly grants these rights; user involvement is not required. like BLUETOOTH.

EXPERIMENTAL RESULT

This section presents the experimental findings made using the machine learning (ML) techniques listed below on the dataset created as previously described.

Table 1 Confusion Matrix

	Predicted Class		
Actual Class		Yes	No
	Yes	True Positive	False Negative
	No	False Positive	True Negative

CONCLUSION

In this post, we offer a practical method for malware detection on the Android platform. The APK file directory structure is painstakingly studied to generate the feature vector for Android malicious applications identification. The key argument of this study is that we view the classification problem of identifying dangerous mobile Internet applications. Experimental results show that the suggested strategy is effective.

REFERENCES

[1]. Number of Mobile Phone Users Worldwide from 2016 to 2023 (In Billions). Availabl online: https://www.statista.com/statistics/330695/number-of-smartphone-uses-worldwide/ (accessed on 19 May 2021).

[2]. Mobile Operating System Market Share Worldwide. Available online: https://gs.statcounter.com/os-market-share/mobile/worldwide/ (accessed on 19 May 2021).

[3]. Number of Android Applications on the Google Play Store. Available online: https://www.appbrain.com/stats/number-of-android-apps/ (accessed on 19 May 2021).

[4]. Gibert, D.; Mateu, C.; Planes, J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. J. Netw. Comput. Appl. 2020, 153, 102526. [Google Scholar] [CrossRef]

[5]. Khan, J.; Shahzad, S. Android Architecture and Related Security Risks. Asian J. Technol. Manag. Res. [ISSN: 2249-0892] 2015, 5, 14-18. Available online: http://www.ajtmr.com/papers/Vol5Issue2/Vol5Iss2_P4.pdf (accessed on 19 May 2021).

[6]. Platform Architecture. Available online: https://developer.android.com/guide/platform (accessed on 19 of May 2021).

[7]. Android Runtime (ART) and Dalvik. Available online: https://source.android.com/devices/tech/dalvik (accessed on 19 May 2021).

[8]. Cai, H.; Ryder, B‚G. Understanding Android application programming and security: A dynamic study. In Proceedings of the 2017 IE E International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 17–22 September 2017; pp. 364–375. [Google Scholar] [CrossRef]

[9]. Stephen Feldman, DillonStadther, Bing Wang, “Manilyzer: Automated Android Malware Detection through Manifest Analysis,” in 2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems.

[10]. William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick Mcdaniel, Anmol N. Sheth,” TaintDroid: An Information-Flow Tracking System for Real time Privacy Monitoring on Smartphones,” ACM Transactions on Computer Systems (TOCS) TOCS Homepage archive Volume 32 Issue 2, June 2014

[11]. G. Y. Wang, After Access: Inclusion, Development, and a More Mobile Internet, International Journal of Communication, 2017, 11:323-326

[12]. Q. Shi, X. Ding, J. Zuo and G. Zillante, Mobile Internet based construction supply chain management: A critical review, Automation in Construction, 2016, 72: 143-154

[13]. Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; Liu, H. A Review of Android Malware Detection Approaches Based on Machine Learning. IEEE Access 2020, 8, 124579–124607. [Google Scholar] [CrossRef]