International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

Android Malware Detection Using Machine Learning

[1]Chandragiri Santhoshi, [2]Gaddi Jahnavi, [3]Vartia Neha, [4]L.Thirupathi

[1,2,3,4]Stanley College of Engineering and Technology for Women, Hyderabad, Telangana.Corresponding author email: thiru1274@gmail.com

Abstract:

On the other hand, the Android platform's ease of use has aided malware development. The standard malware detection method, which relies on signatures, is unable to detect unknown applications. The study presents a lightweight machine learning-based approach for detecting malware on Android handsets. Because of its open-source nature and high performance, the Android smartphone has attracted a large number of users.In this system, we extract features using static and dynamic analysis, and then provide a new feature selection technique based on PCA and relief to reduce the project's feature dimensions. After that, a model will be developed using a variety of categorization algorithms. Experiments have proven that our method for detecting Android malware is effective. Because of its open-source nature and benefits of being free, Android has become increasingly significant in our daily lives. Malicious software, on the other hand, is fast increasing in number.On the other side, malicious software is rapidly expanding in number. As a result, one hot topic right now is how to accurately identify Android spyware.

Key Words: Android, Malware, Machine Learning, Decision Tree, APK, Static Analysis, Dynamic Analysis.

1. Introduction:

Smartphones are widely used throughout the world. Smartphones have so many features these days that they are almost as powerful as mini computers. Smartphones assist us in a variety of ways, including short message service, multimedia messaging service, email, video calling, GPS navigation, speech dictation, voice activated personal assistant, and now, by containing numerous applications and sensors, they assist us in keeping track of our health concerns. There are other operating systems available for smartphones, but Android is the most popular.

On September 23, 2008, Google and the Open Handset Alliance launched the Android platform, which quickly gained traction due to its user-friendliness and ease of creating and releasing applications in the Android Market. The ML Droid-framework, which uses machine learning to identify Android malware, rose to popularity in 2021.

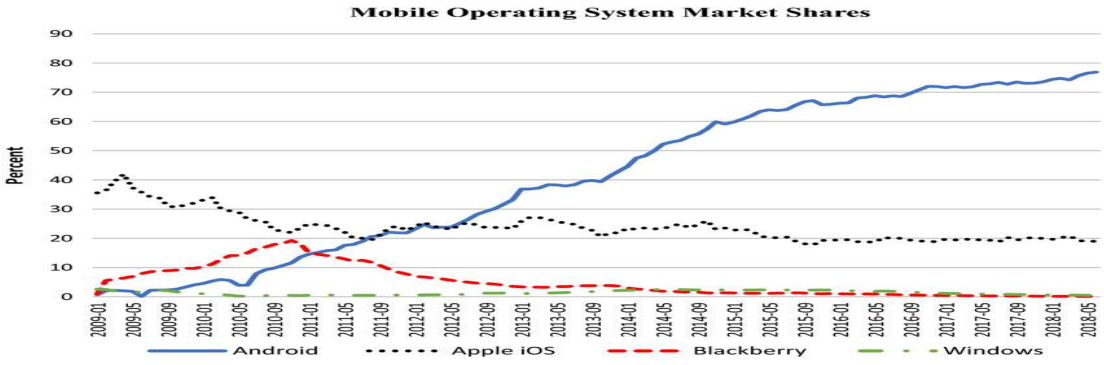

Fig1: World Wide Mobile Operating System Market Share in 2021

The bulk of malware is downloaded unknowingly by users through alternative app stores, and many of the programmes in these unauthorised stores are repackaged versions of apps available on the Google Play Store. In 2004, Cabir, the first mobile virus for the Symbian mobile operating system, was identified. The number of malwares in the mobile market has constantly increased since then.To date, a number of strategies have been utilised to identify malware on mobile platforms. Static procedures and dynamic techniques are the two types of techniques used to identify Android malware. Static approaches do not execute an application and instead examine different criteria such as signature verification and permission analysis to determine whether or not it is malicious.

Static approaches can be easily obscured on their own. Dynamic techniques include analysing an application's behaviour by running it in a limited environment, such as a sandbox. Dynamic analysis extracts some properties of an application to determine whether it is benign or malicious. API tracing and network monitoring are examples of these features; however, they are not all included in one solution. Malware detection has largely relied on machine learning approaches. Fig 1 shows the world-wide mobile operating system market share in the year 2021.

2. Literature Survey

In [1], To detect malware in Android apps, researchers used both static and dynamic analysis. With AI, they were able to combine static and dynamic examinations. They conducted a static analysis by isolating authorizations from the Android manifest.xml document and examining the difference in the number of consents mentioned by good and bad apps. They realised that the number of consents mentioned by a favourable and a harmful application is almost same. This method was tested on a variety of pleasant and unpleasant applications.

In[2], To demonstrate Android applications' harmful capabilities. As a result, they gathered the top 311 Android applications and checked them for events of specific authorization set in each application's arrangement document. Five of these applications were found to be performing dangerous functions, according to this check. Another five suggested potentially dangerous authorizations, but these might be disputed based on the use of the applications.

In [3], They demonstrated DREBIN, a lightweight malware detection solution for Android that enables for legitimate identification of dangerous apps on the phone. To create a shared vector space, DREBIN runs a comprehensive static analysis on the app's AndroidManifest.xml, removing many features (equipment segments, mentioned authorizations, App segments, and separated plans) and dismantling code (limited API calls, used consents, confined API calls, network addresses). When tested with 123,453 legitimate apps and 5,560 malware tests, DREBIN successfully identified 94 percent of malware with a false positive rate of 1%.

In [4], They offered a proactive way to detecting zero-day Android malware that did not rely on malware testing and results to identify potential security concerns posed by untrusted apps. They created Risk Ranker, a scalable platform that analyses apps to identify if they include potentially harmful actions. They performed static analysis on the Dalvik bytecode discovered in each application by isolating the information stream and control stream from the code path. In just four days, they processed 118,318 apps from multiple Android markets. During their study, they discovered 3281 malicious apps.

In [5], Introduced a lightweight static investigation-based framework for determining the location and family of Android malware. Droid Sieve assembles successful and strong features appropriate for computational learning based on a thorough examination of Android malware. When confronted with obscurity strategies like as reflection, encryption, and progressively stacked local code, their findings demonstrate that static inspection for Android can succeed in each case. While significant changes in malware characteristics remain an unresolved question, Droid Sieve remains resistant to cutting-edge obscurity techniques that can be used to quickly identify new and grammatically unique spyware.

In [6], APK Auditor, an authorisation-based Android malware identification framework that employs static research to define and order Android apps as benevolent or evil, is presented. APK Auditor is made up of three parts: a mark data set that store extracted application information and examination outcomes, an Android customer that customers utilise to submit application investigation requests, and a focal worker who communicates with both the mark data base and the mobile phone customer.To test the framework's performance, 8762 apps were used. The results demonstrate that APK Auditor can distinguish the majority of well-known malwares and highlights those with the greatest potential with an accuracy of about 88 percent and a specificity of 0.925.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

3.Objectives

3.1 With the ubiquity of malware on the internet, malware detection is critical because it serves as an early warningsystem for computer security against malware and cyber-attacks. It protects hackers from gaining access to the computer and information from being compromised.

3.2 Malware detection aids in identifying trends that can be used to treat and prevent infections in the future.

4.Proposed Architecture

To detect malicious programmes, both static and dynamic analysis will be used, after which characteristics will be gathered using both methodologies, and our machine learning classifiers will be trained and verified. Static analysis will come first, then dynamic analysis, and then machine learning., As shown in Fig 2.

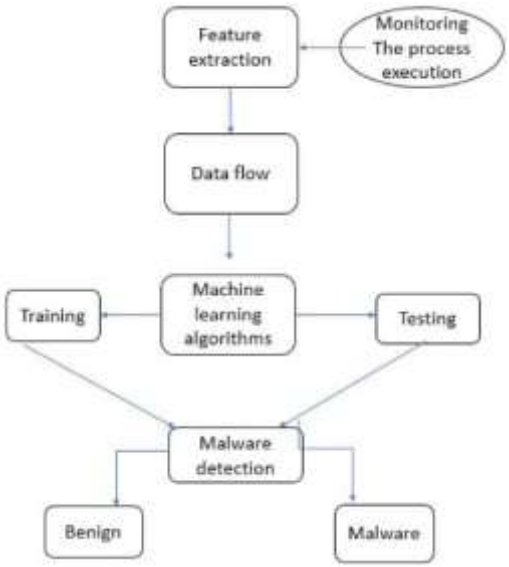

Fig 2: Proposed Architecture for Android Malware Detection

4.1 Static Analysis:

At this point, a program's permissions will be extracted. During the installation process, when a user downloads an app from any app store, he is presented with a list of privileges from which to choose.It is up to him whether he accepts or rejects those permissions. If the user agrees to those rights, the programme will be installed; if the user refuses, the application will not be installed. The majority of users accept all permissions without even looking at the list of permissions. Furthermore, if a user finds a permission dubious, he has no other alternative except to accept or decline the entire list of permissions; the user can only accept or decline the entire set of permissions.

In static analysis, we figure out following things:

i. The total number of permissions used by each programme.
ii. An application's request for permissions. Permissions can be collected from the package without executing the programme. We can determine the number of permissions sought by both benign and

malicious programmes using static analysis, as well as whether the number of permissions requested varies. Both benign and malicious applications require a similar number of permissions.

iii. Next, we'll examine a list of permissions that an application requests to determine if there's a distinction between permissions sought by a benign programme and permissions asked by a dangerous programme. We'll figure out what extra permissions rogue programmes will ask for. The permission lists used by malicious and benign programmes are nearly identical. Both use permissions that are essentially equivalent, yet some harmful applications only use one permission.

4.2 Dynamic Analysis:

Tracing system calls will be used to do Dynamic Analysis. When a programme runs, it uses a variety of system calls to carry out its tasks. There are a few things that will distinguish between benign and malicious programmes.

4.2.1 Both benign and dangerous applications employ a large number of system calls.

4.2.2 A list of system calls used by malicious and benign programmes is provided below. Both benign and malicious apps employ the Strace (System Tracer) programmer to collect system calls. Using Strace to trace system calls, we may acquire a list of all the system calls used by a specific programmer. After that, all unnecessary system calls are filtered out, leaving only the ones that are required to be recorded. Following that, a comparison of those system calls will be performed based on the above considerations to determine which system calls benign programmes use and which system calls malignant apps use.

5.Methodologies

A collection of malicious and non-malicious Android application package (APK) files collected from a publicly available dataset. The static features are then extracted using a python script written in the Jupyter notebook environment. Two of the characteristics are API calls and permissions. To facilitate the training process, these features are created and stored as a data frame in Comma Separated Values (CSV) files. Using a decision tree classifier, test and evaluate the performance of the suggested model.

Decision Tree: A decision tree is a decision-making aid that uses a tree-like model of decisions and their potential outcomes, such as chance event outcomes, resource costs, and utility (as depicted in Fig 3). It's one way to demonstrate an algorithm that is entirely made up of conditional control statements.

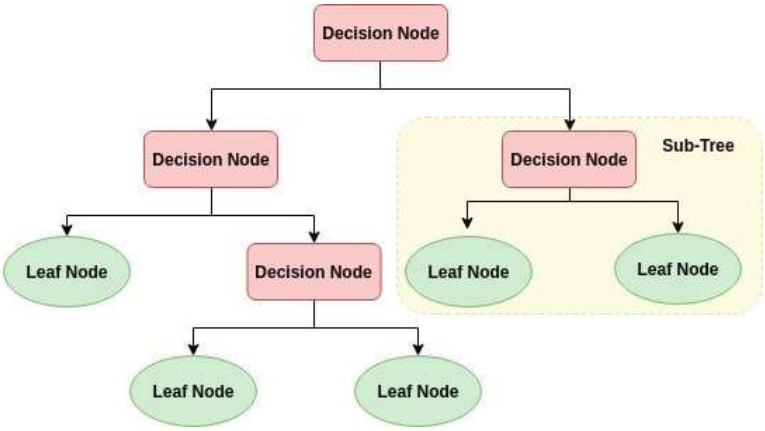

Fig 3: Representation of Decision tree classifier

Another approach to detecting malicious code in Android apps is to use a different methodology. We were able to create and construct an app that can analyse existing and newly installed Android apps to see if they are harmful or not. The system was developed using Android Studio, a Java-based Android SDK, and XML. The Object-Oriented Analysis and Design Methodology (OOADM), which used the Unified Modelling Language (UML) to model the system, was used to analyse, design, and develop it.Malware Detection System clearly demonstrates that the system's introduction is precise, with promising results such as low computational expenditure and high results. This is a useful Java-based programme. After selecting the application to review after it has been transmitted, you remove information from it and send data to the channel. As a result of the responses, if the application is malicious, it will uninstall. This module creates a new section that hasn't appeared in earlier work. After a client has checked an application, a report is generated that includes the application's idea - liberal or vengeful, malware collection, potential harm to the device/archives, and a removal choice.

6.Results and discussions

We took the dataset from Kaggle which is available publicly to everyone. This dataset is downloaded with the extension of .csv. The result comes in the format of HTML page and we select a row of values and identify whether the malware is present or not for that selected row and the same is shown below in the Fig 4 and 5.

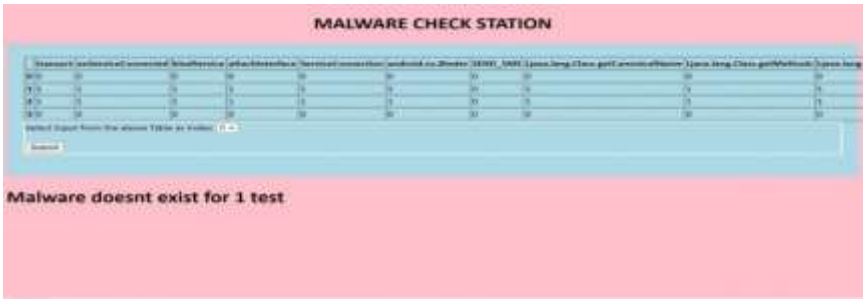

Fig 4: Malware doesn’t exist for the Zeroth row of values of dataset.

Fig 5: Malware exists for the third row of values of dataset.

1. Conclusion

Using machine learning, we construct an Android malware detection system based on many algorithms that can detect unknown Android applications, in contrast to traditional detection approaches. To extract varied data, we apply static and dynamic analytic methodologies. This is something we can utilise to keep malware from being exploited in the first place. This strategy can be used to resolve issues on workstations running Windows and Apple operating systems.

References

2. [1] Malware Repository, http://contagiominidump.blogspot.com
3. [2] Thomas Zefferer, Peter Teufl, David Derler, Klaus Potzmader Alexander Oprisnik, Hubert Gasparitz and Andrea Hoeller "Power Consumption-based Application Classification and malware Detection on Android Using Machine-Learning Techniques" in FUTURE COMPUTING 2013
4. [3] Hahnsang Kim, Joshua Smith, Kang G. Shin "Detecting energy greedy anomalies and mobile malware variants" in MobiSys'08
5. [4] Aubrey-Derrick Schmidt, Rainer Bye, Hans-Gunther Schmidt, Jan Clausen, Osman Kirazy, Kamer Ali Y¨uksely, Seyit Ahmet Camtepe, and Sahin Albayrak "Static analysis of executables for collaborative malware detection on android" in Communications, 2009. ICC '09. IEEE International Conference
6. [5] Leonid Batyuk, Markus Herpich, Seyit Ahmet Camtepe, Karsten Raddatz, Aubrey-Derrick Schmidt, and Sahin Albayrak "Using Static Analysis for Automatic Assessment and Mitigation of Unwanted and Malicious Activities Within Android Applications" in Malicious and Unwanted Software (MALWARE), 2011 6th International Conference
7. [6] Borja Sanz, Igor Santos, Carlos Laorden, Xabier UgartePedrero, Pablo Garcia Bringas, Gonzalo Álvarez. "PUMA: Permission Usage to Detect Malware in Android", in International Joint Conference CISIS'12-ICEUTE´12-SOCO´12 Special Sessions.
8. [7] Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: "TaintDroid: An InformationFlow Tracking System for Realtime Privacy Monitoring on Smartphones" in: Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI) (Oct 2010). [8] Burguera, I., Zurutuza, U., &Nadjm-Tehrani, S. (2011). "Crowdroid: Behavior-based malware detection system for Android" in 2011 ACM CCS Workshops on Security and Privacy in Smartphones and Mobile Devices (SPSM'11), 17-21 October 2011, Chicago, Illinois, USA.
9. [9] Grace, M., Zhou, Y., Zhang, Q., Zou, S., & Jiang, X. (2012). "RiskRanker: scalable and accurate zero-day Android malware detection." in the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys'12), Low Wood Bay, Lake District, United Kingdom
10. [10] Portokalidis, G., Homburg, P., Anagnostakis, K., and Bos, H.: "Paranoid Android: Versatile protection for smartphones" in ACSAC'10, Dec. 2010.
11. [11] Su, X., Chuah, M., Tan, G."Smartphone dual defense protection framework: Detecting malicious applications in android markets" in: Mobile Ad-hoc and Sensor Networks (MSN), 2012 Eighth International Conference on, pp. 153-160 (2012).