ANDROID MALWARE DETECTION and its SECURITY

Vikas Kumar
Department of Computer Science,
Amity School of Engineering and Technology
Uttar Pradesh, India

Assist. Prof. Pradeep Kumar Kushwaha
Department of Computer Science,
Amity School of Engineering and Technology
Uttar Pradesh, India

ABSTRACT

The popularity of mobile apps is growing thanks to advancements in technology. This article specifically discusses the Android mobile platform and its layered approach to app development, as well as the security concerns associated with it. Android is an open-source operating system for mobile phones that includes middleware, a user interface, and app software. However, the Android Market's lack of strict security checks means that it is vulnerable to security threats. To counter this, the article proposes a layered approach for developing secure Android apps and the use of an Android Application Sandbox that can identify potentially harmful programs through static and dynamic analysis.

Keywords -- Mobile-phones, Android OS, Malware, Algorithms, Operating System..

1. INTRODUCTION

As of the current moment, Android is the most widely utilized operating system among end-users. With over 2.5 billion active devices each month android [1] and a growing preference for mobile Internet usage, Android dominates digital services access globally across various devices, from phones to vehicles and specialized tech, serving diverse purposes such as communication, entertainment, finance, and health. As security and privacy become critical concerns, Android must offer reliable safeguards for users and developers.

Even though the security of Android has a complex design, and new updates are released to address new security threats, users often don't have the latest version installed on their devices. Additionally, securing the app distribution system may not be enough to protect the ecosystem since users from all over the world often access third-party app stores that lack a thorough app vetting process. These factors contribute to a system with numerous potential security vulnerabilities.

Current efforts to address Android malware focus on developing new and advanced methods for detecting and classifying malware, typically using machine learning models. However, these academic advances are rarely reflected in the practices of anti-malware vendors, who still rely primarily on signature-based methods that can be easily bypassed with simple code modifications. It is imperative to find more effective ways to identify similarities between various malware samples

2. ANDROID OS MODEL

2.1 Ecosystem Context

To properly understand certain design decisions, it's important to consider the larger Android ecosystem, which is not isolated. A successful ecosystem involves mutual trust among all parties, and it's necessary for the platform to create safe environments where users, developers, and the operating system can establish mutually advantageous (or beneficial terms). If an agreement cannot be reached, then disallowing the action is the most trustworthy operation (default-deny). This concept serves as the foundation for the security model of the Android operating system.

As an operating system focused on end-users, Android aims to be flexible and useful to typical users while also being attractive to developers. To ensure user safety and privacy, user interfaces and workflows need to prioritize safety, and explicit intent is mandatory for any actions that have the potential to jeopardize security or privacy. The OS cannot offload security or privacy choices to individuals who are not specialists in the field, who lack the skills and experience to make them [2].

The Android's ecosystem is an immense and diverse, with various Original Equipment Manufacturers (OEMs) Manufacturing various types of Android devices in large quantities, numbering in the tens of thousands [3]. Some OEMs lack technical expertise and rely on others to develop hardware and firmware, and Devices that are created using the Android Open Source Project (AOSP) can be developed without the need for permission or registration. However, modifying the APIs and other interfaces of these devices can have a notable impact on the overall device ecosystem, and it may take a while before they can be fully adopted by most users.

Developers have the flexibility to write applications in various programming languages, as long as they interact with the Android framework through the clearly defined Java language APIs, which govern the process workflow. ("The Android Platform Security Model - ACM Digital Library") At present, Android doesn't have support for other programming language (except Java) APIs for controlling the basic process lifecycle. However, this flexibility comes with a drawback - security mechanisms cannot depend on compile-time checks or any other assumptions regarding the build environment. Instead, Android security must be based on runtime protections around the app boundary. (“The Android Platform Security Model - ACM Digital Library”).

2.2 THREAT-MODEL

The threat models employed for mobile devices are distinct from those typically utilized for desktop or server operating systems, primarily due to two major factors. Firstly, mobile devices are highly susceptible to being lost or stolen. Furthermore, as an inherent part of their usual usage, they frequently connect to untrusted networks. Additionally, Due to the fact that mobile devices are frequently used in close proximity to their users, they are susceptible to being subjected to a greater amount of privacy-sensitive information compared to other device categories.. A layered threat model for mobile devices was previously presented in a separate study. and we adopt this model to discuss the Android security model in this article [4]. The categories of threats are ranked according to their level of capability, with less numbers indicating more limited and greater numbers indicating more advanced and capable adversarial scenarios.

Specifically, we assume that potential adversaries have the ability to physically access Android devices, posing a threat to various mobile and wearable devices, as well as other forms of Android devices like smart home appliances, vehicles, televisions, and more. This implies that in our assessment of potential threats, we consider Android devices to be within the reach of adversaries or in close proximity to them. This includes scenarios such as theft or loss, as well as instances where multiple users may use the same device and could be innocuous but possibly inquisitive. As a result, we identify particular risks that arise due to physical or close access to these devices.

3. ANDROID SECURITY MODEL

The main purpose of the Android OS is to protect and secure user information, manage system resources efficiently, and maintain a separation between different applications. To accomplish this, several security features have been implemented, such as a strong security system at the OS level via the utilization of the Linux kernel, compulsory confinement of applications, secure communication between processes, endorsement of applications, and the ability for users to specify permissions are implemented as measures to ensure security. In Figure 1, various components and considerations of the Android software stack are displayed, with each layer operating under the assumption that the underlying layer is adequately secured. The Linux kernel is the foundation of the Android security model, providing a user-based permissions model, process isolation, secure IPC, and the ability to remove kernel components. This kernel has been in existence for a long time, has undergone continual enhancement, and is trusted by many industry experts.

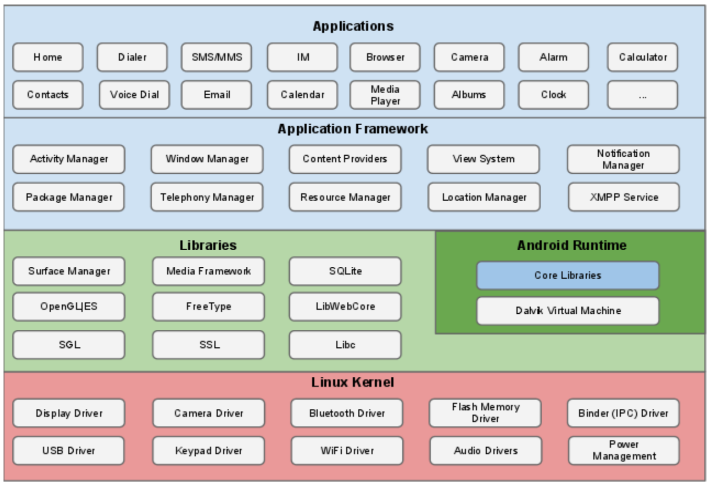


Figure 1: Android software stack

1. The principle of multi-party consent requires the agreement of all primary parties before any action can be taken.. Any single party has the authority to reject or block the action. Having control over data does not necessarily imply ownership. [5].
2. Access to the ecosystem is open, and neither users nor developers need to undergo central vetting or registration processes, as both are considered integral parts of the open system. Explicit support is provided for generic interaction between applications..
3. The Android specification includes the security model as an essential requirement for compatibility. This means that security is considered an integral aspect of the Android operating system. and this requirement is upheld through various test suites, such as Compatibility (CTS), Vendor, and others.
4. A factory reset is a process that brings the device back to a secure state by erasing all the data stored in the writable partitions and restoring it to a state where only the protected partitions are relied upon for integrity.
5. In modern terms, Android apps are not considered as fully authorized representatives for the actions performed by users, since applications function as security principals. In other words, although users use Android apps to perform various tasks, the apps themselves do not have complete authorization to act on behalf of the users[7], [8]. The text also discusses the specific details of each rule and provides examples to explain how the Android security model works.

4. ANDROID-MALWARE

The increasing popularity of Android has made it a major focus for cyber attackers. This is because of its open-source nature and the possibility of reverse engineering the Java code utilized in app development , malicious codes can easily be embedded, leading to a rise in attacks [9]. As per Networks' report in 2015, incidents of mobile malware witnessed a significant surge of 155% in 2011 and a further 614% from March 2012 to March 2013, with 92% of it being targeted at Android. Upgrading to the latest OS can eliminate around

77% of these threats. In the third quarter of 2018, at least 5,000 devices were infected by a single threat that spread using a deceptive voice messaging application that acts as an intermediary, user data is sent to a remote server. The number of new mobile malware collected in 2013 was 2.47 million, signifying a 197% increase from 2012 [10].

The nature of Android malware has progressed from a basic Trojan that sends SMS messages to more advanced codes capable of infecting other applications, encrypt user data, obtain root privileges, install other malicious apps without user knowledge, and load a payload from a remote server [11], [12]. A report by Castillo (2011) provides a comprehensive analysis of Android malware from the past, present and future possibilities.

5. MALWARE-DETECTION

There are generally two methods that can be used to broadly categorize the detection of Android malware:

a. The Signature-based detection which identifies malware based on its identity. However, this method can be easily bypassed through byte code level transformation attacks [10].

b. A machine learning-based detection method that employs a heuristic approach to extract features from the application's behavior [10], such as permissions requests and APIs calls, and applies machine learning algorithms to determine metrics of measurement like accuracy, precision, and false-positive rates.

Numerous techniques have been proposed for detecting Android malware, including using permissions requested by applications, specific application programming interfaces, the fundamental aspects of the applications, such as the underlying code, sandboxing, discretionary access control, component encapsulation, and application signing. [13], [14].

This research work specifically concentrates on utilizing manifest permission, command signature, API calls signature, and intents to detect Android malware. Permissions are rights that developers state in their applications so that they can interact with the system components or modules of other applications [15].

There are four types of permissions in Android: Normal permissions, Signature permissions, System permissions and Dangerous permissions. Signature and System permissions are reserved for firmware-developed software or pre-installed applications. Normal permissions are automatically approved, while Dangerous permissions require users' approval. Researchers have analyzed The Android permission specification includes certain difficult-to-detect permissions, such as activity hijacking, broadcast theft, service hijacking, malicious service launch and malicious activity launch.

6. PAST WORK

Numerous studies and investigations have been conducted pertaining to the detection of malicious software. For instance, Li, Shang, Deng & He (2017) introduced a method that uses the naïve Bayes algorithm to detect Android malware. Their detection model depended on novel malware permissions and training-permissions to improve accuracy, and it yielded a detection rate of 97.59% for non-malicious apps [16][20].

In 2019, Ali introduced a modern approach that combines intelligent techniques in a hybrid model. The model employs a Support-Vector-Machine (SVM) and optimizes it using a Genetic Algorithm (GA) and Particle-Swarm-Optimization (PSO) to enhance the accuracy of the classifier. This method achieved a 95.60% accuracy with the use of GA [17].

Dehghantanha (2018), Dabbagh, and conducted a static analysis of Android applications, examining the presence and frequency of keywords in the manifest file of the applications. The dataset was used to produce better malware detection results, and the highest accuracy of 83% was achieved using the KNN and SVM classification algorithm [18][2.

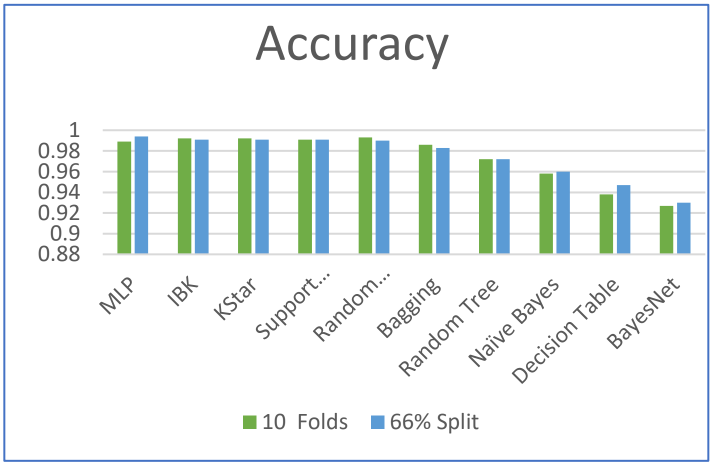
Finally, Sung (2018), Gudla and Shohel Rana optimized and assessed several algorithms by utilizing a classifier that relies on static analysis to identify malware. They achieved an accuracy of 94.33% with the random forest algorithm applied to the dataset [19].

7. METHODOLOGY

- Studied the method calls and Intent ICCs to understand the dynamic features of Android apps. Our analysis captured control flows at a coarse-grained level but not data flows. These traces reveal important dynamic characteristics and security features of Android apps. Our empirical study includes benchmark apps, dynamic analysis inputs, calculated metrics, and study process details.

- To compare a query APK with a dataset, its methods are extracted and divided into basic blocks. Strands are generated from each method and sorted into buckets based on their method. This is done for every sample in the dataset, and the strands are saved for future runs. The query APK strands are compared with all dataset method strands using the Strand Similarity Measure. The Local Evidence Score determines the significance of strand matching, and the Global Evidence Score sums up all LES values to generate a similarity score between methods. More details on each step are discuss.

8. ANALYSIS OF RESULT



COMPARISON OF THE ACCURACY

The utilized dataset comprised of both benign and malware applications. After testing 10 algorithms using a 10-fold cross-validation and a 66% split in the WEKA environment. The results, which are presented in Table-1, include accuracy, false positives, precision, recall, f-measure, ROC, and RMSE. While all algorithms were found to be effective with the dataset, the multilayer perceptron (MLP) was the most successful and displayed the highest accuracy of 99.4% under a 66% split.

References	Techniques	Accuracy	FP-Rate
Current-Paper	**MLP**	**99.40%**	**0.60%**
Yuan et aL (2014)	Deep learning	96.50%	
Zhang et al. (2014)	DroidSIFT	93%	5.15%
Yerima. Sezer,Mcwilliams, & Muttik (2013)	Bayesian-Classifier	92.10%	6.30%
Sato et al. (2013)	Analysis Manifest Files	90.00%	
Gascon et al., (2013)	Function Call graphs	89%	
Abela et al. (2013)	AMDA	78%	
Shang, Li, Deng, & He (2017)	Improved Naive Bayes	97.59%	8.25%
Ali (2019)	SVM with GA	95.60%	6.80%
Kakavand, Dabbagh, & Deh hantanha (2018)	KNN (IBK)	83%	
Shohel Rana, Gudla, & Sun (2018)	Random forest	94.33%	

Table-1 **SHOWING HOW THE PERFORMANCE METRIC OF THE MLP COMPARES WITH THAT OF OTHER LITERATURE SOURCES**

Furthermore, according to the results in Table I, MLP showed the lowest false positive rate of 0.006 and achieved the best recall, the RMSE and f-measure. On the other hand, Random Forest performed the best in terms of accuracy, precision, recall, f-measure, and ROC under the 10-fold validation. However, when considering the overall performance, MLP outperformed other algorithms under the 66% split. The results were also visually presented in the table. When compared to results from other sources, MLP demonstrated superior performance when tested and trained with the 66% split.

9. CONCLUSION

The advancement of technology has led to an increase in the complexity of malware applications. As a result, researchers have developed various models and techniques to improve the detection of these malware. This particular research targeted to assess the effectiveness of classification algorithms in detecting Android malware. The results of the analysis conducted in the WEKA environment indicated that the multi-layer perceptron (MLP) algorithm outperformed other algorithms in terms of precision, accuracy, recall, and f-measure.

Future studies should replicate the use of the MLP algorithm on other android application datasets to ensure that its high performance is not solely due to the particular dataset used in this research. It is crucial to test the algorithm on various datasets to avoid any potential dataset biases. In addition, the current dataset used in this research only has four categories of attributes, and future studies could incorporate more categories to determine if the algorithm remains highly accurate. Finally, MLP should be considered as the classification and detection algorithm for developing anti-malware solutions in future research.

10. REFERENCES

[1] "Android Security 2017 Year in Review 2."

[2] Anne Adams and Martina Angela Sasse, "USERS ARE NOT THE ENEMY".

[3] Sundar Pichai, "Android Has Created More Choice, Not Less. Retrieved from," 2018. https://blog.google/around-the-globe/google-europe/android-has-created-more-choice-not-less/ (accessed Feb. 21, 2023).

[4] R. Mayrhofer, "An architecture for secure mobile devices," *Security and Communication Networks*, vol. 8, no. 10, pp. 1958–1970, Jul. 2015, doi: 10.1002/sec.1028.

[5] ANDREW S. TANENBAUM and HERBERTBOS, “MODERN OPERATING SYSTEMS FOURTH EDITION.”

[6] AOSP. [n.d.]., “Android Compatibility Definition Document. Retrieved from.” https://source.android.com/docs/compatibility/cdd (accessed Feb. 21, 2023).

[7] H. Carter, N. Scaife, P. Traynor, and K.R.B. Butler, “CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data,” in *Proceedings – International Conference on Distributed Computing Systems*, Institute of Electrical and Electronics Engineers Inc., Aug. 2016, pp. 303–312. doi: 10.1109/ICDCS.2016.46.

[8] William Robertson, Amin Kharraz, Davide Balzarotti, Engin Kirda and Leyla Bilge. *Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 9148. in Lecture Notes in Computer Science, vol. 9148. Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-20550-2.

[9] R. Sato, D. Chiba, and S. Goto, “Detecting Android Malware by Analyzing Manifest Files,” *Proceedings of the Asia-Pacific Advanced Network*, vol. 36, no. 0, p. 23, Dec. 2013, doi: 10.7125/apan.36.4.

[10] M. Zhang, Y. Duan, H. Yin, and Z. Zhao, “Semantics-aware Android malware classification using weighted contextual API dependency graphs,” in *Proceedings of the ACM Conference on Computer and Communications Security*, Association for Computing Machinery, Nov. 2014, pp. 1105–1116. doi: 10.1145/2660267.2660359.

[11] S. Verma and S. K. Muttoo, “An android malware detection framework-based on permissions and intents,” *Def Sci J*, vol. 66, no. 6, pp. 618–623, Nov. 2016, doi: 10.14429/dsj.66.10803.

[12] J. Li *et al.*, “Networked human motion capture system based on quaternion navigation,” in *BodyNets International Conference on Body Area Networks*, 2017. doi: 10.1145/0000000.0000000.

[13] L. Davi, A. Dmitrienko, A.-R. Sadeghi, and M. Winandy, “Privilege Escalation Attacks on Android.”

[14] H. Gascon, F. Yamaguchi, D. Arp, and K. Rieck, “Structural detection of Android malware using embedded call graphs,” in *Proceedings of the ACM Conference on Computer and Communications Security*, 2013, pp. 45–54. doi: 10.1145/2517312.2517315.

[15] Institute of Electrical and Electronics Engineers and Nile University, *2019 15th International Conference on Electronics, Computer and Computation (ICECCO).*

[16] F. Shang, Y. Li, X. Deng, and D. He, “Android malware detection method based on naive bayes and permission correlation algorithm,” *Cluster Comput*, vol. 21, no. 1, pp. 955–966, Jun. 2017, doi: 10.1007/s10586-017-0981-6.

[17] W. Ali, “Hybrid Intelligent Android Malware Detection Using Evolving Support Vector Machine Based on Genetic Algorithm and Particle Swarm Optimization,” 2019. [Online]. Available: https://www.researchgate.net/publication/336777808

[18] Institute of Electrical and Electronics Engineers and Nile University, *2019 15th International Conference on Electronics, Computer and Computation (ICECCO).*

[19] M. Shohel Rana, C. Gudla, and A. H. Sung, “Evaluating machine learning models for android malware detection - A comparison study,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Dec. 2018, pp. 17–21. doi: 10.1145/3301326.3301390.