TAP-WAVE-RUB ANDROID APPLICATION (TWRDROID): A LIGHTWEIGHT PERMISSION ENFORCEMENT APPROACH TO EMERGING SMARTPHONE MALWARE.

K.Vijayalakshmi,
Mobile and Pervasive Computing,
Anna University Regional Campus,
Coimbatore, India.

J.Jayavel,
Teaching Fellow,
Anna University Regional Campus,
Coimbatore, India.

Abstract—Smartphones are undoubtedly becoming ubiquitous. Malware has been a serious issue for smartphones and continuing advancing. Traditional defense mechanisms to malware, are not suitable for smartphones due to their resource intensive constraints. This made the malware to spread enormously all over the smartphone users. In this paper, we present the TWR (Tap-Wave-Rub) enhanced android application permission model which incorporates the denied access for the smartphones malware. And also the developing android application TWRDROID will review the permissions of apps and users can find whether it is malicious or not. It can remove the malicious applications. This paper is designed to enhance the security features of the android especially in the permission-based access control of the android application. This Android application will make the users who are prompted to deny or grant individual permissions to an application which needed for the first time. The proposed approach will be very effective for malware detection/prevention with low False Negative Rate and False Positive Rate, while imposing little to no additional burden on the smartphone users.

Index Terms— Android security application, Android permission mechanism, Malware, Smartphone security, Malware detection/prevention

I.INTRODUCTION

A smartphone is an advanced mobile operating system phones which combines features of a PC operating system with other features useful for mobile use. Smartphone malware is easily spread through an insecure app store. Often malicious software is hidden in pirated versions of legitimate apps, which are then distributed through third-party application stores. Malicious software risk also comes from what is known as an "update attack", where a legitimate android application is later changed to include a malware component, which smartphone users then install when they are notified that the app has been updated. Android is a free, open source OS built on Linux for mobiledevices, including the Linux kernel, the Intermediate layer, the Application Framework layer and the Application layer in figure 1.

The Android security is based mainly on permission models. A permission is a controlling access to a part of the code or to data on the device. The biggest flaw is that the user cannot decide to grant individual permissions, while denying others. Many smartphone users, though an app might request a suspicious permission among many legitimate permissions, will still confirm the installation. Because of the above said problems, researchers have been involved to determine techniques that uses individual permissions and the combination of permissions to detect and characterize malware. The limitation is imposed to protect critical data and code that might be misused to damage the user experience.

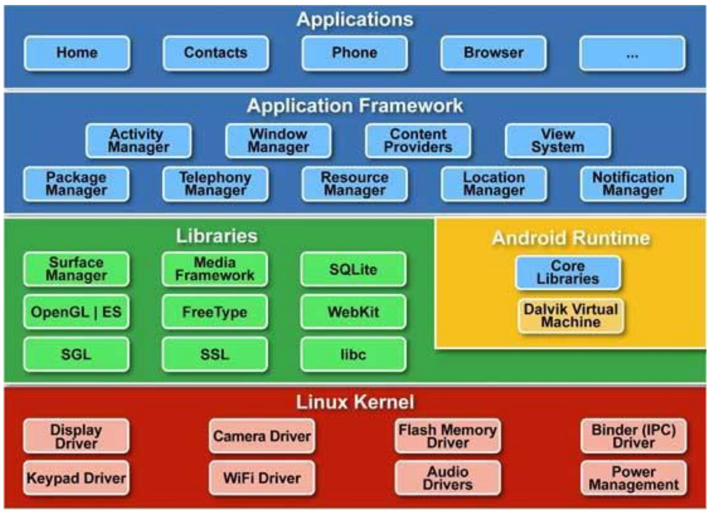

Figure 1: Android Architecture.

II.RELATED WORK

The most closely related concept to ours is the one proposed in [1].It proposes the TWR enhanced permission model in the android platform. Any android application needs to get an user’s acceptance for the requested permissions to access the privacy-related resources. The idea of the TWR system model is to add another layer of permission check before the original android permission check. It also utilizes the concept that whether there are hardware interruptions to differentiate software initiated activity and human initiated activity. To efficiently detect malware from apps available on official and third-party sources, many efforts have been contributed to studying the nature of smartphone environments and their applications in the past ten years. Google tests apps for possible malicious behaviour through a service called Bouncer [2]. Bouncer examines android apps submitted to the Android Market automatically by execution inside a virtual Android platform in Google's cloud infrastructure. Even though malware download numbers reduced since the installation of Bouncer, this service does not provide security against modern attack approaches [3]. Enck et al. [4] suggested a policy-based system called Kirin to detect malware at install time based on undesirable combination of permissions. Diverse concepts evaluate the detection of malware with permissions using machine learning on Android [5-9]. They all realize that a permission-based mechanism can be used as a quick filter to identify malicious android applications. Zhou and Jiang [10] characterize Android applications (both normal and malicious software applications) with individual permissions imposing on the number of occurrences of permissions in those groups. The Android environment incorporates the permission system model to restrict applications inorder to secure the sensitive resources of the users [11]. Thus, the permission system was designed to protect users from applications which has invasive behaviours, but its effectiveness highly depends on the users understanding of permission approval. The developer is responsible for determining appropriately which permissions an application requires. According to these papers [12, 13] and [14], many users do not understand what each permission means and blindly granting permission to them, allowing the application to access sensitive information of the user.

III.TECHNICAL FOUNDATION

Android is a privilege-separated OS, in which each apps runs with a distinct system identity. Further security features of android are a "permission" mechanism that enforces restrictions on the specific operations that a particular process can perform, and single-URI permissions for granting ad hoc access to specific resources. A basic Android application has no permissions associated with it by default, meaning it cannot do anything that would affect the user experience or any data on the device. To make use of protected features of the device, the developer must include one or more <uses-permission> tags in app manifest file of any android application. Table 1 gives some examples of permissions. System permissions are classified into several protection levels. The two most important protection levels are *normal* and *dangerous* permissions

- *Normal* permissions implies areas where android app needs to access data or resources outside the app's sandbox, but where there's very little risk to the user's privacy or the operation of other apps.
- *Dangerous* permissions implies areas where the app wants data or resources that involve the user's private information, or could potentially affect the user's stored data or the operation of other apps.

A central design of the Android security architecture is that no android application, by default, has permission to perform any operations that would adversely affect the operating system, or the user's privacy-related information. This includes reading or writing the user's private data (such as SMS or contacts), accessing the internet, keeping the device awake, etc.

Because each Android application works in a process sandbox, apps must openly share resources and data. They do this by declaring the permissions they need for additional capabilities which are not provided by the basic sandbox. Usually android applications declare the permissions they require, and the Android operating system prompts the user for consent.

Manifest Permission Strings Description	
Call Phone	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed.
Modify Phone State	Allows modification of the telephony state such as power on
Write SMS	Allows an application to write SMS messages.
Read Contacts	Allows an application to read the users contacts data.

Table 1: Examples of Permissions.

IV. PROPOSED WORK

In Android 6.0 Marshmallow, the permissions system was changed to allow the user to control an android apps permissions individually, to block applications if desired from having access to the device's contacts, calendar, phone, sensors, SMS, location, microphone and camera. Full permission control is only possible with root access to the device.

Below 6.0 versions in Android system, the application requests permission when it installs an application. Users can't install the application if they do not accept the application permission. Therefore, users accept the permission without checking the risk of the permission. However, it is not wise to press the accept button carelessly, because it may result in a serious security threat such as a malicious application What we found from the android security applications that have been developed up to now is that most applications have very simple permissions. To mitigate this problem, we are going to develop an android application for managing permission, which is named as TWRDROID. This TWRDROID android application which will review the permissions of apps and users can find whether it is malicious or not. It can remove the malicious applications. Compared with those applications, the newly developing has functions such as permissions as well as the classification by the level of the risk. In this work we utilized Java Micro Edition, Android Software Development Kit, Eclipse the Integrated Development Environment, and Java Programing Language. The environment of the operation of TWRDROID is Android 4.0(API 14) and above, and it doesn't require any permission for usage.

a) Review of TWR Enhanced Permission Model [1]

Permission mechanisms have become usual on smartphone operating systems to provide access control to sensitive services for installed third party android application. The Android environment has the most extensive permission system model and become a market leader. Thus, we review the design of TWR system on the Android platform. The idea of the TWR system model is to add another layer of permission check before the original APC (Android permission check). Let begin with the assumption that the adversary is not able to maliciously alter the kernel control flow. Intercepted permission requests in the android are handled by the five components in the TWR's architecture: TWR PermissionChecker, TWR GestureManager, TWR GestureExtractor, TWR TemplateCreator, and TWR GestureDatabase. The architecture of TWR is depicted in Figure 3.The TWR PermissionChecker stands in front of the original APC. When an application begins a request to access a sensitive service, the request is intercepted by TWR PermissionChecker. This component interacts with TWR GestureManager to check whether the requested service is protected by a certain gesture. If not, the request is forwarded to the APC as usual. Otherwise, TWR GestureManager interacts with the TWR GestureExtractor to begin collecting gesture data (tapping, waving, and rubbing in this paper). Later, the captured data is sent to the TWR GestureManager to proceed further. For user-dependent gesture recognition, we prefer to create a gesture template which is used as a reference in the original recognition stage. The user can interact with the TWR TemplateCreator application to register a new gesture template, to update and delete exiting gesture templates. TWR TemplateCreator is an Android application which allows interaction between TWR and the Android user. When the user creates, deletes, or modifies the gesture data, it needs to retrieve and store the gesture template to TWR GestureDatabase through TWR GestureManager. Since the gesture is user dependent, it compares the similarity between the newly captured data with the gesture template stored in the TWR Gesture-Database.

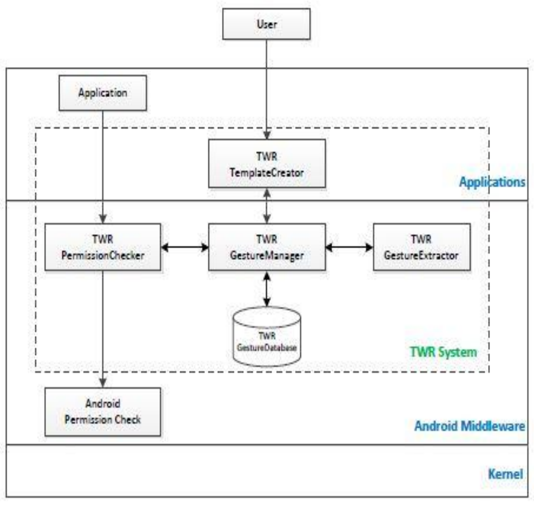

Figure 3: TWR Architecture.

V.CONCLUSION

In this paper, we are developing an android application named TWRDROID for managing the permission of the Android applications. The permission system will change to allow the user to control an application's permissions individually to block apps if desired from having access to the device's contacts, calendar, phone, SMS, location, microphone and camera. And also, we reviewed the TWR (Tap-Wave-Rub) enhanced permission model, a lightweight permission enforcement approach for smartphone malware prevention and detection. Our future effort will be focused on realizing this approach in practice and further evaluate it with a wide range of smartphones and smartphone users. Our results will suggest the proposed approach will be very effective for malware prevention, with quite low FPR and FNR, while imposing little to no additional burden on the users.

VI.REFERENCES

[1] Tap-Wave-Rub: Lightweight Human Interaction Approach to Curb Emerging Smartphone Malware Babins Shrestha, Di Ma, , Yan Zhu, Haoyu Li, and Nitesh Saxena, in IEEE,2015.

[2] H. Lockheimer, "Android and security," Google Mobile Blog, Feb, vol. 2, 2012.

[3] X. Jiang, "An evaluation of the application verification service in android 4.2, January 2014."

[4] W. Enck, M. Ongtang, and P. McDaniel, “Mitigating Android software misuse before it happens,” 2008.

[5] D. Arp, M. Spreitzenbarth, M. H¨ubner, H. Gascon, K. Rieck, and C. Siemens, “DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket,” 2014.

[6] C.-Y. Huang, Y.-T. Tsai, and C.-H. Hsu, “Performance Evaluation on Permission-Based Detection for Android Malware,” in Advances in Intelligent Systems and Applications-Volume 2, pp. 111–120, Springer, 2013.

[7] X. Liu and J. Liu, “A Two-Layered Permission-Based Android Malware Detection Scheme,” in Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on, pp. 142–148, IEEE, 2014.

[8] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. A´ lvarez, “Puma: Permission usage to detect malware in android,” in International Joint Conference CISIS12-ICEUTE´ 12-SOCO´ 12 Special Sessions, pp. 289–298, Springer, 2013.

[9] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, J. Nieves, P. G. Bringas, and G. A´ lvarezMaran˜o´n, “MAMA: Manifest Analysis for Malware Detection in Android,” Cybernetics and Systems, vol. 44, no. 6-7, pp. 469–488, 2013.

[10] Y. Zhou and X. Jiang, “Dissecting android malware: Characterization and evolution,” in Security and Privacy (SP), 2012 IEEE Symposium on, pp. 95–109, IEEE, 2012.

[11] M. Frank, B. Dong, A. P. Felt, and D. Song, “Mining Permission Request Patterns from Android and Facebook Applications,” in ICDM, pp. 870–875, 2012.

[12] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, “Android permissions: User attention, comprehension, and behavior,” in Proceedings of the Eighth Symposium on Usable Privacy and Security, p. 3, ACM, 2012.

[13] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall, “A conundrum of permissions: installing applications on an android smartphone,” in Financial Cryptography and Data Security, pp. 68–79, Springer, 2012.

[14] F. Tchakount´e, P. Dayang, J. M. Nlong, and N. Check, “Understanding of the Behaviour of Android Smartphone Users in Cameroon: Application of the Security,”