International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 09
August 2017

Effective Risk Communication for Android Apps

[1]Ms.E. Sai Priya ; [2]Mrs.N. Swapna Goud ; [3]Mr.G.Vishnu Murthy
[1]M.Tech Student, Department of Computer Science and Engineering, Anurag Group of Institutions, Telangana, India.
[2]Assistant Professor, Department of Computer Science and Engineering, Anurag Group of Institutions, Telangana, India.
[3]Professor and Head of the Department, Department of Computer Science and Engineering, Anurag Group of Institutions, Telangana, India.
[1]Mail id: priya.china55@gmail.com, [2]Mail id: swapnagoudcse@cvsr.ac.in,
[3]Mail id: hodcse@cvsr.ac.in

Abstract:

The android platform adopts permissions to protect sensitive resources from untrusted apps. However, after permissions are granted by users at install time, apps could use these permissions (sensitive resources) with no further restrictions. Thus, recent years have witnessed the explosion of undesirable behaviors in Android apps. An important part in the defense is the accurate analysis of Android apps. However, traditional syscall-based analysis techniques are not well-suited for Android, because they could not capture critical interactions between the application and the Android system. This paper presents VetDroid, a dynamic analysis platform for generally analyzing sensitive behaviors in Android apps from a novel permission use perspective. VetDroid proposes a systematic permission use analysis technique to effectively construct permission use behaviors, i.e., how applications use permissions to access (sensitive) system resources, and how these acquired permission-sensitive resources are further utilized by the application. With permission use behaviors, security analysts can easily examine the internal sensitive behaviors of an app. Using real-world Android malware, we show that VetDroid can clearly reconstruct fine-grained malicious behaviors to ease malware analysis. We further apply VetDroid to 1249 top free apps in Google Play. VetDroid can assist in finding more information leaks than TaintDroid, a state-of-theart technique. In addition, we show how we can use VetDroid to analyze fine-grained causes of information leaks that TaintDroid cannot reveal. Finally, we show that VetDroid can help to identify subtle vulnerabilities in some (top free) applications otherwise hard to detect. .

Keywords

Android Security, Permission Use Analysis, Vetting Undesirable Behaviors, Android Behavior Representation.

1. Introduction

Smartphone platform are becoming more and more popular these days. To protect sensitive resources in the smart phones, permission-based isolation mechanism is used by modern smart phone systems to prevent untrusted apps from unauthorized accesses. In Android, an app needs to explicitly request a set of permissions when it is installed. After permissions are granted to an app, there is no way to inspect and restrict how these permissions are used by the app to utilize sensitive resources. Unsurprisingly, Android has attracted a huge number of attacks. According to McAfee threat report of Q3 2012, Android remains the largest target for mobile malware and the number almost doubled in Q4 2012. While these malware apps are clear examples containing undesirable behaviors, unfortunately even in supposedly benign apps, there could also be many hidden undesirable behaviors such as privacy invasion.

An important part in the fight against these undesirable behaviors is the analysis of sensitive behaviors in Android apps. Traditional analysis techniques reconstruct program behaviors from collected program execution traces. A rich literature exists that focuses on solutions to construct effective behavior representations. All these research efforts have mostly used system calls to depict software behaviors because system calls capture the intrinsic characteristics of the interactions between an application and the underlying system. Previous studies differ from each other only in how to structure the set of system calls made by the applications is not readily applicable due to the following unique features of Android.

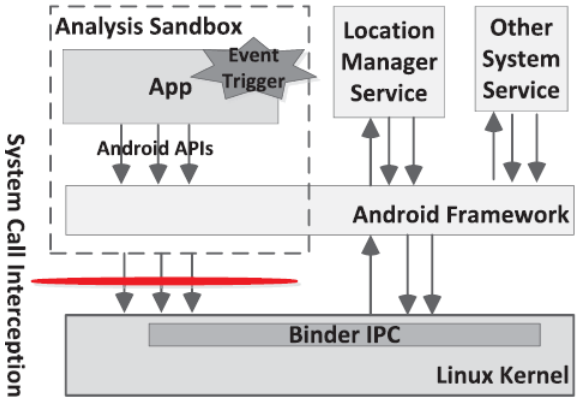

Fig. 1. Limitations of syscall-based solutions on Android platform.

A. Android Framework Managed Resources

As depicted in Figure 1, Android is an application framework on top of Linux kernel where applications do not directly use system calls to access system resources. Instead, most system resources in Android are managed and protected by the Android framework, and the application-system interactions occur at a higher semantic level (such as accessing contacts, call history) than system calls at the Linux Kernel level. Indeed, Android provides specific APIs for applications to access system resources and regulates the access rules. Using system calls to learn the interaction behaviors between applications and Android will lose a semantic view of accesses to Android resources, degrading the quality and precision of the reconstructed behaviors.

B. Binder Inter-Process Communication (IPC)

In Android, system services are provided in separated processes, with a convenient IPC mechanism (Binder) to facilitate the communication among system services and applications. Binder IPC is heavily used in Android and recommended in the design of applications. Figure 1 demonstrates the problems brought by the wide use of IPC to traditional syscall-level behavior reconstruction. First, traditional solutions would only intercept a lot of system calls used to interact with the Binder driver, hiding the real actions performed by the application. Second, the use of IPC in Android apps breaks the execution flow of an app into chains among multiple processes, making the evasion of traditional syscall-based behavior monitoring easier.

C. Event Triggers

Android employs an event trigger mechanism to notify interested applications when certain (hardware) events occur. In this model, for example, if an application wants to be notified when the phone's location changes, it just needs to register a callback for such an event. When Android sniffs a location change event from the location sensors, it notifies all the interested applications of the latest location by invoking their registered callbacks. Although the event notification is proceeded via Binder IPC (sycall), this asynchronous resource access model via system delivery is quite different from the synchronous application-request access model in three aspects. First, selecting privileged event notifications in a syscalls requires ad-hoc Binder IPC dissecting. Second, intercepting event notification is far away from identifying the callbacks because application may have its specific logic in dispatching events to specific callbacks. Thirdly, could find that application registered callbacks are application code, so their executions cannot be captured with syscalls. As a result, traditional behavior reconstruction methods will lose such important behavior characteristics. The above analysis indicates that a general method to analyze sensitive behaviors of Android apps is highly desired. Since Android does not use system calls as the main mechanism to isolate applications, system calls do not appear to be a good vehicle for representing behaviors.

Apps are developed in such way that to protect sensitive resources in the smart phones. Permission - based isolation mechanism is used by modern smart phone systems to prevent untrusted apps from unauthorized accesses. The main challenge to protect sensitive data. Permission Use Analysis as a new and complementary aspect in analyzing Android apps.

2. Literature Survey

Security Analysis can be described as a systematic analysis technique using permission. This paper elaborates a dynamic analysis platform for generally analyzing sensitive behaviors in Android apps called 'Security Analysis' using permission base analysis. It actually shows how applications use permissions to access sensitive resources, and how these acquired permission sensitive resources are further utilized by the application. Internal behaviours of apps can be easily tested using permission behaviours. Security Analysis can also remake the fine-grained behaviours. Security Analysis can track all potential sensitive behaviours inside the apps. Security Analysis overcomes the problems of previous already done researches.

Android-based Smartphone users can get free applications from Android Application marketing platform. But, certification of application is not done by organizations and they may contain malware applications that can steal privacy information for users. Permissions are adapted by android platform in order to protect sensitive resources from untrusted

apps. After permissions are granted by users at install time, applications can use these permission with no further restrictions. Recent years have witnessed the expandation of undesirable behaviors in Android apps. Analysis of android apps must be done correctly. As traditional system call based analysis techniques could not capture critical interactions between the application and the Android system, are not well suited for android. Internal behaviors of apps can be easily examined by security analysts using permission behaviors. Using real-world Android malware, security analysis can clearly reconstruct fine-grained malicious behaviors to ease malware analysis. Security Analysis can assist in finding more information leaks than Taint Droid, a state-of-threat technique.

Smartphone can be described as a mobile device equipped with enhanced computing capability and connectivity, such as iPhone by Apple, Windows Phone by Microsoft, etc. In the past few years, the global telephony industries have increased the sales of Smartphones. A Smartphone is usually sold with an in-built mobile operating system (OS) together with a number of pre-installed ‘‘applications’’ packaged by the device manufacturer. An application, the software running on smart phones, enhances the Smartphone’s functionality and supports the interaction with end users to accomplish their tasks. web browser, alarm clock, address book, media player and Calendar are the common applications provided by the device manufacturers, but one important application exists on every Smartphone—the ‘‘application store’’, which allows end users to access online application markets to browse and download additional applications of their choice. Smartphone application distribution was highly dependent on third party sources, where individual application developers were free to upload their products. There is still a large group of end users who prefer visiting third-party application markets, due to a huge number of low-price applications being available. But not all the applications from markets are ‘‘safe’’. The software that is specially designed to harm a device, its OS or other software is called ‘‘Malware’’, which stands for ‘malicious software’. The increasing sales of smart phones has increased the growth of mobile malware.

After an application is installed, a set of application programming interfaces (APIs) are called during the runtime. Each API call is associated with a particular permission. When an API call is made, the Android OS checks whether or not its associated permission has been approved by the user. Only a match result will proceed to execute the certain API call. In this way, the required permissions are able to protect the user’s privacy-relevant resource from the unauthorized operations. It cannot fully stop the malware developers who can declare any required permissions for their applications. Smartphones platforms are becoming more and more popular these days. To protect sensitive resources in the smart phones, permission-based isolation mechanism is used by modern Smartphone systems to prevent entrusted apps from unauthorized accesses. In Android, an app needs to explicitly request a set of permissions when it is installed. , After permissions are granted to an app, there is no way to inspect and restrict how these permissions are used by the app to utilize sensitive resources. Unsurprisingly, Android has attracted a huge number of attacks. While these malware apps are clear examples containing undesirable behaviors, unfortunately even in supposedly benign apps, there could also be many hidden undesirable behaviors such as privacy invasion. An important part in the fight against these undesirable behaviors is the analysis of sensitive behaviors in Android apps. Security Analysis can be used to analyze fine-grained causes of information leaks that Taint Droid cannot reveal. All the researches efforts for these have mostly used system calls to depict software behaviors because system calls capture the intrinsic characteristics of the interactions between an application and the underlying system.

3. System Analysis

A. Existing System

Traditional techniques for analyzing malware permission related analysis techniques.

a) Malware Analysis

Plenty of studies have focused on analyzing malware at the level of system call. In sequenced system calls with arguments were translated into actions that capture the sample’s behaviors, such as changes to file system, modifying registries. Crowdroid used system call vectors to represent the signature of malicious behaviors. The temporal pattern of system calls was proposed to depict the application behavior for Symbian platform. The limitation of reconstructing behaviors using linear system calls with a large-scale study. They reconstructed resource access behaviors by considering read/write system calls to identify malicious intents with the observation that most benign programs access their own files and registries. Dependency graphs of system calls were firstly proposed in to represent behaviors. It captures the intrinsic application-system interactions and seems to be a good solution for behavior representation. In researchers reconstructed dependencies among system calls by matching the types of their

arguments and return values. Comparettital employed dynamic taint analysis to track the dependencies among system calls.

Syscall-based techniques are not well-suited for the Android platform due to the inability of monitoring Android-specific behaviors. DroidScope seems to notice these problems by seamlessly reconstructing the semantics from system calls and Java. It only refines existing work, leaving the root problems of Android's special permission mechanism and programming model untouched. Although they were reported to detect known and unknown malware samples, the do not analyze the fine-grained internal behaviors of malware samples, which is the focus of Security Analysis

b) Permission Analysis

The effectiveness of the time-of-use and install-time permission grant mechanism. This work was extended in to provide guidelines for platform designers in determining the most appropriate permission-granting mechanism for a given permission. Permission-based security rules were used to design a lightweight certification framework that could mitigate malware at install time. Android's permission system by introducing runtime constraints on the granted permissions. Mobile IFC introduced context-aware policies for permission enforcement. In this permission model, permissions are granted depending on the device state, such as the GPS location or time of the day. This mechanism brings a new kind of flexibility and interesting security applications. To help end users understand application behaviors at install time, AppProfiler devised a two-step translation technique which maps API calls to high-level behavior profiles. While Security Analysis also tries to provide better behavior understanding, it is a tool provided for different users (security analysts) and it uses a different new technique/perspective (permission use behavior) to precisely capture application-system interactions and sensitive behaviors inside an app. novelly leveraged Natural Language Processing (NLP) techniques to assess the risks of application by measuring whether the developers have explicitly explained the reasons for requiring permission-sensitive resources in its functional descriptions. Performed an empirical analysis on the expressiveness of Android's permission sets and discussed some potential improvements for Android's permission proposed the first solution to systematically detect over privileged permissions in Android apps and one third of the applications in this study were found to be over privileged. Probabilistic models of permission request patterns or permission request sets were also used to indicate the risk of new applications. To extract permission specifications for Android, used API fuzz testing while adopted static analysis on Android source code.

Copper Droid was an analysis tool to reconstruct Android-specific behaviors with syscall-level introspection. It might be more suited for large-scale automated analysis, while Security Analysis to help a human analyst to understand much better internal behaviors of the analyzed malware. Our Security Analysis differs from all existing work in that it provides the first systematic frame.

B. Proposed System

Security Analysis:

Design a dynamic analysis system called Security Analysis to automatically perform permission use analysis on Android apps. It is non-trivial to construct an effective permission use analysis technique. Security Analysis overcomes several key challenges to completely identify all permission sensitive behaviors with accurate permission use information during the runtime. Analysis is performed in two phases: first, Security Analysis identifies all sensitive interactions between the Android system and apps with accurate permission use information by intercepting all invocations to Android APIs and sniffing exact permission check information from Android's permission enforcement system; second, based on the identified sensitive interactions, Security Analysis tracks all potential sensitive behaviors inside the apps, by recognizing the exact delivery point in the application for each permission-sensitive resource and locating all the use points of these resources with permission-based tainting analysis. Security Analysis also features a driver to enlarge the scope of the dynamic analysis to cover more application behaviors and a behavior profiler to generate behavior graphs with highlighted sensitive behaviors for analysts to examine.

To evaluate the effectiveness of permission use analysis and to analyze Security Analysis real-world Android malware. The results show that the permission use behaviors reconstructed by Security Analysis can significantly ease the malware analysis. Apply Security Analysis to more than one thousand top free apps in Google Play Store. Security Analysis finds more information leaks than the state-of-the-art leak detection system TaintDroid and shows its capability to analyze the fine-grained incentives of information leaks among the apps. Security Analysis even detects subtle Account Hijack Vulnerability in a top free Android app. The analysis overhead caused by Security Analysis is reasonably low for an offline analysis tool.

This paper makes the following major contributions.

•Analyze the limitations of existing syscall-based behavior analysis methods when applied to Android platform and propose permission use analysis as a new perspective to analyze Android apps.

•Present a systematic framework to reconstruct permission use behaviors. Our automated solution is able to completely identify all possible permission use points with accurate permission information.

• Implement a prototype system, Security Analysis, and evaluate its effectiveness in analyzing real-world Android apps. Security Analysis not only greatly eases the analysis of malware behaviors, but also assists in identifying fine grained causes for information leakages and even subtle vulnerabilities in benign Android apps otherwise hard to detect.

Vetting Market Apps

Next,use Security Analysis to vet 1,249 top (benign) apps crawled from Google Play official store. These apps are top free apps crawled from 32 different categories such as games, education, entertainment, finance, social, sports, tools. Also use multiple emulators to parallelize the process of reconstructing permission use behaviors for these apps. There are several interesting findings.

Finding 1 (Security Analysis Can Assist in Finding More Information Leaks Than TaintDroid): Based on the reconstructed permission use behaviors, implement a simple permission-based filter that selects permission use graphs with at least one permission to read system resource and one permission to exfiltrate data to a remote party. The selected graphs are further classified with regard to E-PUPs. Manually check these classified behaviors and confirm four kinds of information leaks, as listed in Table IV.

Finding 2 (Security Analysis Can Inspect the Fine-Grained Causes of Information Leakage): Permission Use Analysis captures the internal logic of permission usages inside an app, enables us to analyze the fine-grained procedure of information leakage. Analyze the permission use behaviors of several information leaks reported by Security Analysis to investigate the contexts of reading and leaking sensitive information. Mainly focus on Phone Number and Location leakage cases because they are relatively interesting.

Finding 3 (Security Analysis Can Help Detect Subtle Application Vulnerabilities): Since SMS service is unique and quite important for smart phones, analyze 33 apps that request both RECEIVE_SMS and SEND_SMS permissions by running these apps in Security Analysis By carefully examining the permission use behaviors, find that the Viber application is vulnerable to Account Hijack attack.

Use Security Analysis to reconstruct the permission use behavior of the activation process. As Viber intercepts incoming SMS messages in ActivationSmsReceiver, and extracts the activation code from the message body using a regular expression. Once an activation code is matched, the activation process is proceeded in the Registration Activity. ActivationCodeReceived() function. By carefully examining the whole permission use behavior, Viber does not check the origin of an activation SMS. An attacker could pass the activation by intercepting the activation SMS from the victim and sending it to the attacker's Viber client, causing the victim's account hijacked. It is not hard to steal an SMS from a victim, especially when the Account Hijack attack on the victim could lead to a reasonable profit. SMS stealing could be possibly implemented by malware such as SMSReplicator. To further confirm this vulnerability, perform an experiment to hijack the Viber account of avolunteer in our group. By stealthily replacing an app in his smartphone into our repackaged version (the activation SMS from Viber server is forwarded by our repackaged app to the attacker's device. After binding the volunteer's phone number to the attacker's device, free calls and messages are successfully initiated to his friends on behalf of his identity.

4. Design and Implementation

A. Modules

1. Security Analysis model.
2. Permissions use Analysis module.
3. Trigger module.

Module Description:

Security Analysis model:

A risk signal two relevant measures are the warning rate which defines how often a user receives warnings generated by the risk signal and the detection rate which defines what percentage of malicious apps will trigger the signal.

Permissions use Analysis module:

Risk signals based only on apps from the Android market are more robust as they are not tuned to detect malicious apps in our particular data set, and aim only at detecting apps that request too much permission. Furthermore, it may be desirable for the signals to use only critical

Trigger module:

Different apps have different functionalities, and thus may require different permissions; it thus makes sense to take into account the intended functionality of an app when deriving a risk signal based on permissions.

5. Conclusion

The results from four user studies validated our hypothesis that when risk ranking is presented in a user-friendly fashion, e.g., translated into categorical values and presented early in the selection process, it will lead users to select apps with lower risk. The majority of participants preferred to have such a risk metric in Google Play Store. A risk metric would cause positive changes in the app ecosystem. When users prefer lower-risk apps, developers will have incentives to better follow the least-privilege principle and request only necessary permissions. It is also possible that the introduction of this risk score will cause more users to pay for low risk apps. Creates an incentive for developers to create lower risk apps that do not contain invasive ad networks and in general over-request permissions. Our studies are not the last word on the question of how to best present risk information. For example, we have also not examined how the risk score interacts with other factors to affect a user's choice, such as user ratings in the natural setting and whether an app is free or not. Also of interest is how users behave when choosing among a list of search results (as opposed to choosing between two options). These topics are important ones for future research.

6. References

[1] W. Enck et al., "TaintDroid: An information flow tracking system for real-time privacy monitoring on smartphones," in Proc. 9th USENIX Conf. OSDI, pp. 1–6, 2010.

[2] IDC: Android Market Share Reached 75% Worldwide in Q3 2012. [Online]. Available: http://techcrunch.com/2012/11/02/idc-android-Market-share-reached-75-worldwide-in-q3-2012/, accessed May 7, 2013.

[3] D. Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to Android," in Proc. 17th ACM CCS, 2010, pp. 73–84.

[4] Mcafee Threats Report: Third Quarter 2012. [Online]. Available: http://www.mcafee.com/ ca/ resources / reports / rp - quarterly - threat - q3-2012.pdf, accessed May 7, 2013.

[5] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," in Proc. 5th Int. Conf. DIMVA, Jul. 2008, pp. 108–125.