International Journal of Scientific and Technical Advancements
ISSN: 2454-1532

Android Security Threats and Proposed Solutions: An Overview

Akshay Bhardwaj[1], A. J. Singh[2]
[1, 2]Department of Computer Science, Himachal Pradesh University, Himachal Pradesh, India
Email address: akshay117@gmail.com, aj.hpucs@gmail.com

Abstract— Android is most by and large utilized stage for cell telephones today which brags of a moved security model having MAC and sandboxing. These parts gift masters and clients to restrain the execution of an application to the amuse allotted. The misuse of vulnerabilities of the endeavor is bound inside the favorable position farthest reaches of an applications sandbox. Advantage animating assaults have made puzzling as the use of android structures have broadened. Distinctive sorts of portions have given some kind of rest to the originators however the security highlight managing by the engineers has not helped much. In this paper we talk about the stray bits of the favorable position expanding strike and the assorted systems used to counter and keep this issue.

Keywords— Threats;solutions; android; security; overview.

I. INTRODUCTION

Since the first introduction in 2008, Android has gained a tremendous number of users over the last few years. Smartphones are the fastest growing technology market segment. According to Gartner [1], a technology research and advisory firm, 1.1 billion devices running on Google's Android OS were shipped in 2014 alone, marking its 80 percent mobile market share. Attributing to this fast-pace increment is the proliferation of Android apps, which provides an ever-growing application ecosystem. Officially reported by Android Google Play Store, the number of apps in the store has reached over 1.6 million in early 2015, surpassing its major competitor Apple Apps Store [2]. Mobile applications are essential to the smartphone experience. Mobile applications are getting increasingly sophisticated, robust, life-engaging, and privacy-intrusive. The market offers a wide variety of applications ranging from entertainment, productivity, health care, to online dating, home security, and business management [3]. Users depend more and more on mobile devices and applications.

As mobile applications are gaining increasing popularity among users, the privacy and security of smartphone users become a concern. Due to the large user base, smart devices are used to store sensitive personal information more frequently than laptops and desktops. As a consequence, a malicious third-party app can not only steal private information, such as the contact list, text messages, and location from its user, but can also cause financial loss of the users by making secretive premium-rate phone calls and text messages [4]. At the same time, the rapid growth of the number of applications on Android markets makes it hard for app market places, such as Google App Store for example, to thoroughly verify if an app is legitimate or malicious. As a result, mobile users are left to decide for themselves whether an app is safe to use. In addition, unlike iOS, Android device owners do not have to root or "jailbreak" their devices to install apps from "unknown sources". This gives Android users broad capability to install pirated, corrupted or banned apps from Google Play simply by changing a systems setting. This provides further incentive for the users to install third-party applications, but exposes their privacy to significant security risks [5].

The exponentially increasing number of Android applications, the unofficial apps developers, and the existing security vulnerabilities in Android OS encourage malware developers to take advantage of such vulnerable OS and apps and steal the private user information to inadvertently harms the apps markets and the developer reputation [6]. Moreover, since Android OS is an open source platform, it allows the installation of third-party market apps, stirring up dozens of regional and international app-stores such as PandaApp [7] and GetJar [8]. Android malware can gain control of device, steal private information from users, consume excessive battery, use telephony services to steal money from users' bank accounts, and even turn the device into a botnet zombie.
There are a large variety of Android vulnerabilities and they can occur in any layers of Android OS stack, such as application layer or framework layer. Vulnerabilities also appear in benign apps through the accidental inclusion of coding mistakes or design flaws. As we described before, the flawed Android OS provides a fertile ground for attackers. There are a variety of security issues on Android phones, such as unauthorized access from one app to the others (information leakage), permission escalation, repackaging apps to inject malicious code, colluding, and Denial of Service (DoS) attacks.

Realizing these shortcomings in the current Android architecture, much efforts have been put towards addressing these problems [9]. In addition to Android OS's various security measures such as sandboxing and Android permission model, many security and privacy solutions were proposed to cope with the existing security Android OS vulnerabilities, including many resource management systems such as [10, 11,

Akshay Bhardwaj and A. J. Singh, "Android security threats and proposed solutions: An overview," *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 103-106, 2016.

12] and security solutions using different approaches and techniques [13, 14, 15].

II. Android Security Issues and Threats

Android security is built upon a permission-based mechanism which regulates the access of third-party Android applications to critical resources on an Android device. Such permission-based mechanism is widely criticized for its coarse-grained control of application permissions and the inefficient permission management, by developers, marketers, and end-users. For example, users can either accept all permission requests from an app to install it, or not to install the app. This type of permission management is proved to be undesirable for the devices security. In this section, we discuss the main security issues of the Android, which leads to user information leakage and puts the user's privacy in jeopardy [21], [22], [23].

2.1 Information leakage

In current Android architecture design, apps are restricted from accessing resources or other apps unless it is authorized by the users. Users have to grant all resource access requests before installing and using an app. Information leakage occurs when users grant resources without any restriction from OS. However, Android's permission control mechanism has been proven ineffective to protect user's privacy and resource from malicious apps. Studies showed that more than 70% of smart phone apps request to collect data irrelevant to the main function of the app [24][25]. With more than 1.4 million available apps in Google Play, and a great number of apps from miscellaneous third-party markets, a significant number of malicious apps have been exposed to Android users for installation. However, when installing a new app, only a small portion of users pay attention to the resource being requested, since they tend to rush through prompted permission request screens to get to use the application. Only a small portion (3%) of users are cautious and make correct answers to permission granting questions. In addition, the current Android permission warnings do not help most users make correct security decisions [26]. The "blaming the users" approach has become a failure to protect Android users. As pointed out in [26, 27], the reasons for the ineffectiveness of the current permission control system include: (1) inexperienced users do not realize resource requests are irrelevant and will compromise their privacy, (2) users have the urge to use the app and may be obliged to exchange their privacy for using the app.

2.2 Privilege escalation

Privilege escalation or permission escalation attacks were leveraged by exploiting publicly available Android kernel vulnerabilities to gain elevated access to resources that are normally protected from an application or user. This type of attack can result in unauthorized actions from applications with more privileges than intended, which causes many sensitive information leakage. Android exported components can be exploited to gain access to the critical permissions [28].

2.3 Repackaging Apps

Repackaging is one of the most important and common security issues of the Android OS. Repackaging is the process of disassembling/decompiling of .apk files using reverse-engineering techniques and adding (injecting) malicious code into the main source code. Repackaging techniques that can be used on the Android platform allow malicious code to be disguised as a normal app. It is difficult to distinguish between a repackaged malicious code and a normal app because the repackaged app usually appears to function in the same way as the legitimate one. The repackaging steps are as follows [29, 30]:

Unpacking: unpacking APK files using available tools such as apktool, which is a tool based on reverse-engineering.

Decompiling: decompiling the Java source code using JAD and extracting the source code of Java classes.

Code injection: injecting code and adding resources into the main source code using Java developing environments.

Repacking: rebuilding the files using apktool and signing the generated files using jarsigner.

Geimini and KungFu are examples of trojans which are based on APK repackaging. These Trojans can be bundled into many valid Android apps.

2.4 *Denial of Service (DoS) attack*

The increasing number of smartphone users and prevalence of mobile devices (phones, tablets) which are connected to the Internet can be a platform for growth of DoS attacks. Since the majority of smartphones are not equipped with the same protections (i.e. anti-virus programs) as PCs, malicious apps find it as a proper platform for DoS attacks. Overusing limited CPU, memory, network bandwidth and battery power are the main goals of DoS attacks [31].

2.5 Colluding

Colluding threat is a client-side attack. In this attack, users install a set of apps developed by the same developer and same certificate and grant different types of permissions including sensitive and non sensitive. After installing apps, these apps can take advantage of a shared UID and get access to all their permissions and resources [32].

III. Proposed Solutions

Considering the security issues that we described in section 4, so far there have been proposed many studies towards the principles and practices to manage resource usage [10-12], [33-37] and security solutions to address these vulnerabilities. The existing security and privacy solutions are classified into three categories. We explain the categories in more details in related sections. Since proposed works that we cover in this survey, use different tools and techniques to implement their solutions.

Before going into further details, in this section we described the main categories and all existing applied techniques.

3.1 Existing techniques and mechanisms

3.1.1 Static Analysis

Those works that use static analysis approach [38][30] are based on the application's structure and code [39]. In this section we describe the main techniques of static analysis.

Akshay Bhardwaj and A. J. Singh, "Android security threats and proposed solutions: An overview," *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 103-106, 2016.

- Application Signature: As we described before, any Android application has a unique signature. Signature-based solutions check the contents or patterns of an application against a dictionary of malware signatures. If they find a matching, they can take action. This method is somewhat limited by the fact that it can only identify a limited amount of emerging threats, e.g. generic, or extremely broad, signatures.
- Permission Analysis: This mechanism works based on granted permissions to the applications. They assess the risk of the granted permissions and the sensitivity of the resources. Depending on the risk level, they analyze and detect the malicious apps.
- Control Flow Analysis In this type of static analysis techniques, it needs to extract apps' Control Flow Graph (CFG) and look for existing possible resource misusing and vulnerabilities within the application's code. Based on the discovered threats and vulnerabilities, they make a decision on maliciousness or vulnerability of the application.

3.1.2 Dynamic Analysis

Existing works that use dynamic analysis [40-46] mainly work based on application behavior analysis during the runtime process. There are three main parameters that can be considered as application's behavior and activities: system calls, battery consumption, and network usage [46].

3.1.3 Crowdsourcing

Crowdsourcing is the process of obtaining needed services, ideas, or content by soliciting contributions from a large group of people [486. In Android OS security scope, it is defined as a process in which we collect data from users or devices toward improving the security of devices and privacy preserving. For example, it can be collecting the system call log of a device or users' reviews on an app.

3.1.4 Policy-based

Using this technique, solutions require users to define several policies on the prepared services in order to customize the service. For example, in smartphone security area, the policies can be the level of permission granting restriction to applications.

3.1.5 Recommendation-based

In this approach, they help inexperienced users through providing recommendations on challenging app security and privacy decisions such as granting permissions to apps and restricting apps' resource accesses.

REFERENCES

[1] Gartner, "Gartner: 1.1 billion android smartphones, tablets expected to ship in 2014," Online; accessed at June 5, 2015, http://tinyurl.com/n8t3h9y.

[2] Victor, "Android's google play beats app store with over 1 million apps, now officially largest," Online; accessed at May 12, 2013, http://www.phonearena.com/news/Androids-Google-Play-beats-App-Store-with-over-1-million-apps-now-officially-largest id45680.

[3] "Number of available applications in the google play store," 2015, http://www.statista.com/.

[4] W. Rothman, "Smart phone malware: The six worst offenders," Online; accessed at April 17, 2015. http://www.nbcnews.com/tech/mobile/smart-phone-malware-six-worst-offenders-f125248.

[5] "Bit9 report: Pausing google play: More than 100,000 android apps may pose security risks," 2012, https://www.bit9.com/files/1/Pausing-Google-Play-October2012.pdf.

[6] "Pandaapp," Online; accessed at August 7, 2015, http://www.pandaapp.com.

[7] "Pandaapp," Online; accessed at August 7, 2015, http://www.getjar.mobi.

[8] W. Enck, "Defending users against smartphone apps: Techniques and future directions," in Proc. of the 7th International Conference on Information Systems Security (ICISS'11), Kolkata, India, LNCS, S. Jajodia and C. Mazumdar, Eds., vol. 7093.Springer Berlin Heidelberg, December 2011, pp. 49–70. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-25560-1 3

[9] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in Proc. of the 18th annual international conference on Mobile Computing and Networking (CMCN'18), Istanbul, Turkey. ACM, August 2012, pp. 317–328. [Online]. Available:http://doi.acm.org/10.1145/2348543.2348583

[10] O. R. E. Pereira and J. J. P. C. Rodrigues, "Survey and analysis of current mobile learning applications and technologies," ACM Computing Surveys, vol. 46, no. 2, pp. 27:1–27:35, Dec. 2013. [Online]. Available:http://doi.acm.org/10.1145/2543581.2543594

[11] B. Liu, S. Nath, R. Govindan, and J. Liu, "DECAF: Detecting and characterizing ad fraud in mobile apps," in Proc. of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI'14), Seattle, WA, USA. USENIX Association, April 2014, pp. 57–70. [Online]. Available:

[12] http://dl.acm.org/citation.cfm?id=2616448.2616455

[13] E. Fernandes, B. Crispo, and M. Conti, "Fm 99.9, radio virus: Exploiting fm radio broadcasts for malware deployment," Information Forensics and Security, IEEE Transactions on, vol. 8, no. 6, pp. 1027–1037, June 2013.

[14] R. Fedler, J. Sch¨utte, and M. Kulicke, "On the effectiveness of malware protection on android," June 2013.

[15] C. Jarabek, D. Barrera, and J. Aycock, "Thinav: Truly lightweight mobile cloud-based anti-malware," in Proc. of the 28th Annual Computer Security Applications Conference (ACSAC'12), Orlando, Florida, USA.
ACM, December 2012, pp. 209–218. [Online]. Available: http://doi.acm.org/10.1145/2420950.2420983

[16] S. Brahler, "Analysis of the android architecture," 2010. [Online]. Available: https://os.itec.kit.edu/ downloads/sa 2010 braehler-stefan android-architecture.pdf

[17] Android, "Google android documents, android framework architecture," Online; accessed at May 28, 2015, https://source.android.com/devices/. [Online]. Available: https://source.android.com/devices/

[18] N. Elenkov, Android Security Internals: An In-Depth Guide to Android's Security Architecture, 1st ed. San Francisco, CA, USA: No Starch Press, 2014.

[19] Android, "Google android documents, android application architecture,"Online; accessed atMay 28, 2015, http://developer.android.com/guide/components/fundamentals.html. [Online]. Available: http://developer.android.com/guide/components/fundamentals.html

[20] J. Annuzzi, L. Darcey, and S. Conder, Introduction to Android Application Development: Android Essentials, ser. Developer's Library. Addison Wesley, 2014. [Online]. Available: https://books.google.com/books?id=c1kXAgAAQBAJ

[21] [21] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. Gaur, M. Conti, and R. Muttukrishnan,"Android security: A survey of issues, malware penetration and defenses," 2015. [Online]. Available:http://dx.doi.org/10.1109/COMST.2014.2386139

[22] C. Efstratiou and I. Leontiadis, "What is the price of free," Online; accessed at April 17, 2012, http://www.cam.ac.uk/research/news/what-is-the-price-of-free.

[23] S. Gunasekera, Android Apps Security, 1st ed. Berkely, CA, USA: Apress, 2012.

[24] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in Proc. of the 8th Symposium on Usable Privacy and Security

(SOUPS'12), Pittsburgh, PA, USA. ACM, July 2012, pp. 3:1–3:14. [Online]. Available: http://doi.acm.org/10.1145/2335356.2335360

[25] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in Proc. of the 18th ACM conference on Computer and communications security (CCS'11), Chicago, IL, USA.ACM, October 2011, pp. 627–638.

[26] L. Davi, A. Dmitrienko, A.-R. Sadeghi, and M. Winandy, "Privilege escalation attacks on android," in Proc. of the 13th Information Security Conferenec (ISC'11), Boca Raton, Florida, USA, LNCS, M.- Burmester, G. Tsudik, S. Magliveras, and I. Ilic, Eds., vol. 6531.Springer Berlin Heidelberg, October 2011, pp.346–360. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-18178-8 30

[27] H. Huang, S. Zhu, P. Liu, and D. Wu, "A framework for evaluating mobile app repackaging detection algorithms," in Proc. of the 6th International Conference on Trust and Trustworthy Computing (TRUST'13), London, UK, LNCS, M. Huth, N. Asokan, S. ˇCapkun, I. Flechais, and L. Coles-Kemp, Eds., vol. 7904. Springer Berlin Heidelberg, June 2013, pp. 169–186. [Online]. Available:http://dx.doi.org/10.1007/978-3-642-38908-5 13

[28] W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in Proc. of the 2nd ACM Conference on Data and Application Security and Privacy (CODASPY'12), San Antonio, Texas, USA.ACM, March 2012, pp. 317–326. [Online]. Available: http://doi.acm.org/10.1145/2133601.2133640

[29] [29] E. Kovacs, "Wi-fi direct flaw exposes android devices to dos attacks," Online; accessed at July 8, 2015. http://www.securityweek.com/wi-fi-direct-flaw-exposes-android-devices-dos-attacks.

[30] C. Marforio, A. Francillon, and S. Capkun, Application Collusion Attack on the Permission-Based Security Model and Its Implications for Modern Smartphone Systems.Department of Computer Science, ETH Zurich, 2010. [Online].Available:https://books.google.com/books?id=nvszMwEACAAJ

[31] J. Crussell, R. Stevens, and H. Chen, "MAdFraud: Investigating ad fraud in android applications," in Proc. of the 12th annual international conference on Mobile systems, applications, and services (MobiSys'14), Bretton Woods, NH, USA. ACM, June 2014, pp. 123–134. [Online]. Available: http://doi.acm.org/10.1145/2594368.2594391

[32] A. Gember, C. Dragga, and A. Akella, "ECOS: Leveraging software-defined networks to support mobile application offloading," in Proc. of the 8th ACM/IEEE symposium on Architectures for networking and communications systems (ANCS'12), University of Texas in Austin, TX, USA, October 2012, pp. 199–210.[Online]. Available: http://doi.acm.org/10.1145/2396556.2396598

[33] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang, "Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing," in Proc. of the 12th ACM Conference on Ubiquitous Computing (UbiComp'12), Pittsburgh, PA, USA.ACM, September 2012, pp. 501–510. [Online]. Available: http://doi.acm.org/10.1145/2370216.2370290

[34] D. Barrera, J. Clark, D. McCarney, and P. C. van Oorschot, "Understanding and improving app installation security mechanisms through empirical analysis of android," in Proc. of the 2nd ACM workshop on Security and privacy in smartphones and mobile devices (SPSM'12), Raleigh, NC, USA.ACM, October 2012, pp. 81–92. [Online]. Available: http://doi.acm.org/10.1145/2381934.2381949

[35] A. Pathak, A. Jindal, Y. C. Hu, and S. P. Midkiff, "What is keeping my phone awake?: Characterizing and detecting no-sleep energy bugs in smartphone apps," in Proc. of the 10th international conference on Mobile systems, applications, and services (MobiSys'12), Low Wood Bay, Lake District, UK. ACM, June 2012, pp. 267–280. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307661

[36] P. Faruki, V. Ganmoor, V. Laxmi, M. S. Gaur, and A. Bharmal, "Androsimilar: Robust statistical feature signature for android malware detection," in Proc. of the 6th International Conference on Security of Information and Networks (SIN'13), Aksaray, Turkey.ACM, November 2013, pp. 152–159. [Online]. Available: http://doi.acm.org/10.1145/2523514.2523539

[37] A. Gosain and G. Sharma, "A survey of dynamic program analysis techniques and tools," in Proc. of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA'14), Odisha, India, S. C. Satapathy, B. N. Biswal, S. K. Udgata, and J. Mandal, Eds., vol. 327.Springer International Publishing, November 2015, pp. 113–122. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11933-5 13

[38] B. Rashidi, C. Fung, and T. Vu, "Recdroid: A resource access permission control portal and recommendation service for smartphone users," in Proc. of the 2014 ACM MobiCom Workshop on Security and Privacy in Mobile Environments (SPME'14), Maui, Hawaii, USA. ACM, September 2014, pp. 13–18. [Online]. Available: http://doi.acm.org/10.1145/2646584.2646586

[39] A. R. Beresford, A. Rice, N. Skehin, and R. Sohan, "Mockdroid: Trading privacy for application functionality on smartphones," in Proc. of the 12th Workshop on Mobile Computing Systems and Applications (HotMobile'11), Phoenix, Arizona, USA. ACM, March 2011, pp. 49–54. [Online]. Available: http://doi.acm.org/10.1145/2184489.2184500

[40] M. Backes, S. Gerling, C. Hammer, M. Maffei, and P. von Styp-Rekowsky, "Appguard: Enforcing user requirements on android apps," in Proc. of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'13), Rome, Italy. Springer-Verlag, March 2013, pp. 543–548. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36742-7 39

[41] G. Russello, A. B. Jimenez, H. Naderi, and W. van der Mark, "Firedroid:Hardening security in almost-stock android," in Proc. of the 29th Annual Computer Security Applications Conference (ACSAC'13), New Orleans, Louisiana, USA. ACM, December 2013, pp. 319–328. [Online]. Available: http://doi.acm.org/10.1145/2523649.2523678

[42] [42] Y. Jing, G.-J. Ahn, Z. Zhao, and H. Hu, "Riskmon: Continuous and automated risk assessment of mobile applications," in Proc. of the 4th ACM Conference on Data and Application Security and Privacy (CODASPY'14), San Antonio, Texas, USA. ACM, March 2014, pp. 99–110. [Online]. Available: http://doi.acm.org/10.1145/2557547.2557549

[43] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-based malware detection system for android," in Proc. of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM'11), Chicago, Illinois, USA.ACM, October 2011, pp. 15–26. [Online]. Available: http://doi.acm.org/10.1145/2046614.2046619

[44] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in Proc. of the 16th ACM Conference on Computer and Communications Security (CCS'09), Chicago, Illinois, USA. ACM, November 2009, pp. 235–245. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653691

[45] T. Ball, "The concept of dynamic analysis," ACM SIGSOFT Software Engineering Notes (SEN), vol. 24, no. 6, pp. 216–234, Oct. 1999. [Online]. Available: http://doi.acm.org/10.1145/318774.318944

[46] WikiPedia, "Wikipedia, crowdsourcing definition," Online; accessed at May 29, 2015, http://en.wikipedia.org/wiki/Crowdsourcing. [Online]. Available:http://en.wikipedia.org/wiki/Crowdsourcing