Analysis of the Impact of Permissions on the Vulnerability of Mobile Applications

Gouayon Koala(✉), Didier Bassolé, Aminata Zerbo/Sabané, Tegawendé F. Bissyandé, and Oumarou Sié

Laboratoire de Mathématiques et d'Informatique, Université Joseph Ki-Zerbo, Ouagadougou, Burkina Faso
gouayonkoala1@gmail.com, dbassole@gmail.com, aminata.sabane@gmail.com, tegawende.bissyande@fasolabs.org, oumarou.sie@gmail.com
http://www.univ-ouaga.bf

Abstract. In this paper, we explored the potential risks of authorizations unexplained by benign apps in order to maintain the confidentiality and availability of personal data. More precisely, we focused on the mechanisms for managing risk permissions under Android to limit the impact of these permissions on vulnerability vectors. We analyzed a sample of forty (40) apps developed in Burkina Faso and identified abuses of dangerous authorizations in several apps in relation to their functional needs. We also discovered combinations of dangerous permissions because it exposes the confidentiality of the data. This analysis allowed us to establish a link between permissions and vulnerabilities, as a source of risk of data security. These risks facilitate exploits of privileges that should be reduced. We have therefore proposed the need to coordinate resolution mechanisms to the administrators, developers, users to better guide the required permissions by benign apps on Android.

Keywords: Permission abuse · Vulnerability · Privilege exploit · Security

1 Introduction

The availability of many apps and features for smartphones and tablets has attracted a large number of users, which in turn has generated interest for growing number of malware authors [22]. The latter benefit from access to sensitive resources to conduct their business despite existing detection and protection efforts [14]. To this end, we are studying the security of a sample of forty (40) apps developed in Burkina Faso, with an emphasis on analyzing the risks associated with permissions.

We have particularly worked on the Android platform which dominates the mobile OS market with more than 81% users [13]. This popularity of Android makes it the most widespread mobile operating system in the world and therefore

R. Zitouni et al. (Eds.): AFRICOMM 2019, LNICST 311, pp. 3–14, 2020.
https://doi.org/10.1007/978-3-030-41593-8_1

the preferred target of attacks. Although the existing studies have proposed several mobile data protection alternatives, they have not focused on permission management, which remains an important issue. We propose to analyze the potential risks of permissions in order to propose mechanisms to ensure the security of data stored on Android.

The rest of this document is organized as follows: the Sect. 2 presents the context and motivations of this study. In the Sect. 3, we present some existing related work before explaining in the Sect. 4 our analytical methodology adopted to achieve our objectives. The Sect. 5 is devoted to the presentation of results followed by proposals to improve the protection of user data. In the Sect. 6, we conclude by reviewing our analyses with our contributions and also present future work.

2 Context

The use of smartphones and tablets with increasing technological characteristics, combined with the multitude of diversified apps, has led to a recent explosion in their market [23]. In 2018 alone, more than 1.55 billion smartphones sold worldwide accounted for 85.7% billion of total sales, compared to less than 260 million computers sold, according to Gartner's report [26,27].

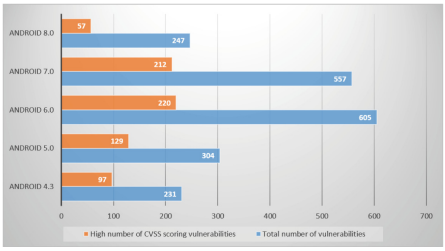
This use of mobile phones leads to a continuous growth of sensitive data on these devices. In 2015, more than 16.2% of the files downloaded through file sharing services contained sensitive data [8]. This sensitive data includes conference call numbers, passwords, credit card or bank account numbers, alarm codes and secure corporate offices, names and phone numbers, multimedia content, etc. [11] and reveals a personal nature. With users having so much personal data on their devices, the confidentiality and availability of this data becomes an imperative [5,9,10]. Mobile OS are becoming more and more like computer operating systems. As a result, the data protection challenges in these mobile devices are becoming similar to the challenges of computer platforms.

Android dominates the market and therefore attracts authors of malware and researchers [12,14]. This market domination makes it a privileged target for pirates. Friedman et al. [7] presented a taxonomy of threats such as malware, phishing and social engineering, direct attack by hackers, communication data (interception and spoofing), loss and theft of removable storage devices or media, malicious insider actions and user policy violations.

3 Related Works

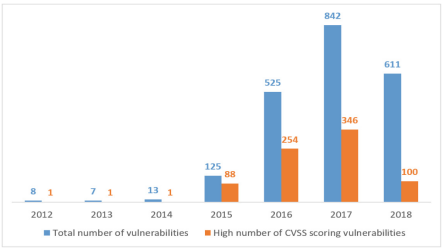
The work of Shewale et al. [11] and Jimenez et al. [12] have revealed that, despite Android's patch efforts, the number of vulnerabilities continues to grow. These vulnerabilities cause denial of service attacks (KillingInTheNameOf [19]), privilege escalations (GingerBreak [11]), code execution and non-authenticated access (Master Key), sniffing with clear text sending of connection information and the middle man with monitoring user activity on the Android browser.

Figures 1 and 2 show the evolution of the vulnerabilities extracted from the Common Vulnerabilities Exposures (CVE) site [25]. Vulnerabilities with a score between 9.3 and 10 are critical because they expose more data.



x-axis: number of vulnerabilities
y-axis: Android version

Fig. 1. Vulnerabilities by version



x-axis: years
y-axis: number of vulnerabilities

Fig. 2. Vulnerabilities per year

Among the attacks against Android, malware represents the largest part and is the most recurrent [13–15,24]. Thanh [2] has grouped these malware into families with malicious activities and areas where searches have been conducted. He selected the recognizable features of ordinary users and concluded that 99% was designed for Android. The most common malicious objects on smartphones Android are computer viruses, worms, adware, spyware, ransomware, botnets, rootkits, Trojans and bugs [8,19,21]. Also, more than 86% of malware is piggy-backed apps [3]. Unfortunately, the mechanisms for detecting and blocking these attacks remain insufficient. Several tools based on static or dynamic analysis [1–3,20] have been developed for securing mobile data. FlowDroid [16], developed by Arzt et al. allows to determine the flows for sensitive Android sources. Avdienko et al. have developed the MudFlow tool [17], which uses several classifiers, trained on the data flow of benign apps, to automatically identify apps with suspicious features. Chin et al. [18], designed *ComDroid*, for vulnerability detection in app communication systems. However, these tools cannot verify the existence of attacks or fixes for vulnerabilities found in the app design. Some characteristics are resistant to obsfucation. Thus, Sawadogo [4] proposed an extraction technique based on graph partitioning. This involves using the INFOMAP algorithm to extract community-related characteristics in order to use supervised learning to detect associated risks. The work of He et al. [1] is helping to detect and reduce malware threats. In [6], Gilbert et al. [6] proposed *AppInspector*, a cloud-based approach to monitoring suspicious behaviour of certain resources. This approach makes it possible to identify the risks related to the confidentiality of data stored in mobile applications.

4 Analytical Methodology

4.1 Characteristics of the Analyzed Apps

Studies in some countries have identified security breaches of data stored via mobile apps [2]. Thus, through this study, we wish to put a particular emphasis on the degree of data exposure with the use of apps developed in Burkina Faso with the habits of developers.

Our sample includes a total of forty (40) apps developed in Burkina Faso. This sample includes at least one app in each app category that Google Play offers. All these apps have been downloaded from Google Play. This approach allows us to have apps that have undergone Google's verification tests before being published on its site. We were interested in the number of users of these apps as a criterion. On the one hand, we have chosen the apps that are most downloaded from the Google Play website. On the other hand, we were interested in the different categories of activities (see Fig. 3). We have apps in all sectors of activity such as tourism, news, health, education, financial transactions, justice, culture, religion, job search, history, geolocation services and entertainment.

Fig. 3. Analyzed apps

The objective of this analysis is to evaluate the risks of attacks related to data confidentiality, availability and integrity. This consists in verifying the management of access authorizations, code vulnerability, intellectual property protection (the piggybacking) of mobile apps developed in Burkina Faso. Our analysis therefore focused on the functionalities of the app, the resource requirements of these features, the necessary permissions for these features, the required permission abuses, the communication between apps and the integration of advertising libraries considered dangerous.

4.2 Analysis Tools

For the analysis of our apps, we used the static approach. Our technique is based on analyses based on comparisons. The choice of this approach to analyze our apps is linked to the following advantages:

- it allows to inspect the source code of an app, because the semantic information, methods and structure of an app are contained in its file manifest;
- is an approach that is not influenced by the continuous evolution of Android environments;
- it allows to analyze the characteristics of a normal app and that of its app piggybacked;
- it allows to check the interactions of an app with the system;
- reveals potential security threats and privacy breaches by examining of code;
- some malicious apps do not activate immediately when performing dynamic detection analysis.

The current method used by static analysis is reverse engineering. Before you can inspect the source code of an Android app, you must decompress its apk. We used the Apktool decompression tool (version Apktool 2.2.2) (Fig. 4).

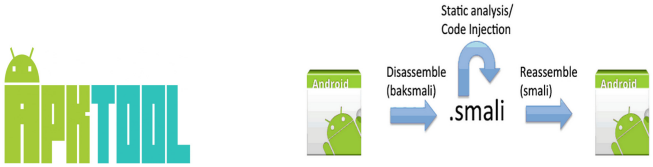


Fig. 4. Tools and procedure for decompiling/recompiling an apk.

Apktool is used to disassemble (or reassemble) the file *class.dex* in the apk and get the bytecode of the file *.dex*. It is used with the tools smali and baksmali (smali-2.2.5 and baksmali-2.2.5). smali allows you to have the files in a more human-readable format and also to compile the file if you have made changes. baksmali is used to decompile the files *class.dex*.

5 Results of the Analysis

One of the disadvantages of the Android security model is the permissions management [17]. Android has approximately one hundred and thirty (130) permissions, including permissions that are at risk with respect to their access to sensitive and personal information. Some permissions are new and more exploited by malware authors [14].

5.1 Impacts of Permissions

Developers of malware exploit the weaknesses of permissions management at several levels. Figure 5 presents cases of privileges granted, by inheriting permissions from dangerous libraries (Scenario 1) or by reusing the code (Scenario 2). Libraries being integrated into the host apps that use them, they essentially form a symbiotic relationship. A library can effectively take advantage of and naturally inherit all the authorizations that a user can grant to an app, thus compromising the confidentiality, availability and integrity of sensitive data. For the inheritance of permissions by piggybacked apps, the pirate may, after modification of the code of the benign app, add dangerous permissions or libraries to the initial permissions to compromise the confidentiality and availability of the data and grant themselves additional privileges.

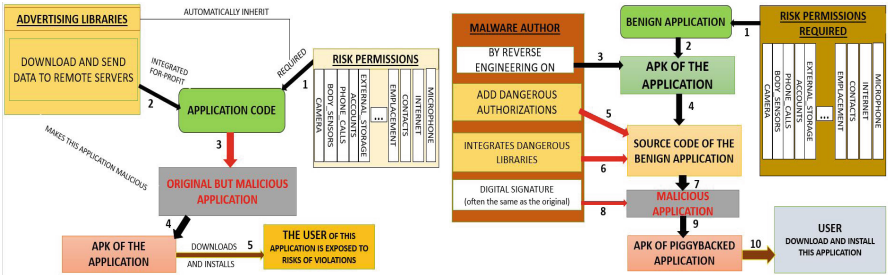


Fig. 5. Inheritance of dangerous permissions

The consequences of these scenarios are the exploitation of privileges granted through dangerous permissions, the probable collection of private information, the risk of confidentiality violations and data availability, an app identical to the original that the user will use unsuspectingly and that will expose his data. The Fig. 6 (Scenario 3) presents the exploitation of data residue privileges after data recovery from an uninstalled app. These are data residue attacks. When a

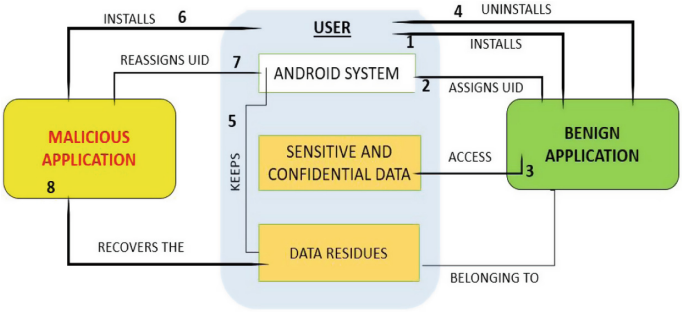


Fig. 6. Scenario 3

user installs an app, the system assigns an identifier called a User ID (UID). It is with this ID that the app can access the resources it needs. If this app has privileges to access sensitive data (such as access to accounts), this data is stored in several components accessible with the UID. If this app is uninstalled, this data is stored in several forms as data residue. If the user installs a new app of the same type, the system can assign the UID of the uninstalled app to the user and the new app therefore inherits all the privileges of the uninstalled app.

These three scenarios show that the permissions granted are vectors of exposure of sensitive data and constitute a weakness that makes these resources vulnerable.

5.2 Results

Our analysis allowed us to determine the most requested resources across apps requiring the same authorization. To determine the permissions granted in each app, we represented the set of app permissions request data as a binary matrix with N as the number of apps, and T as the total number of permissions, $x \in \{0,1\}^{N*T}$. The entry $x_{it} = 1$ means that the i app requests permission t. The line $x_{i*} \in \{0,1\}^{T}$ represents all the required permissions of the app i. And for each app analyzed, we have the required dangerous authorizations, all the permissions granted and the permissions necessary for the app's functional needs. Figure 7 associates to each permission the number of apps having this authorization while Fig. 8 associates to each app its number of permissions.

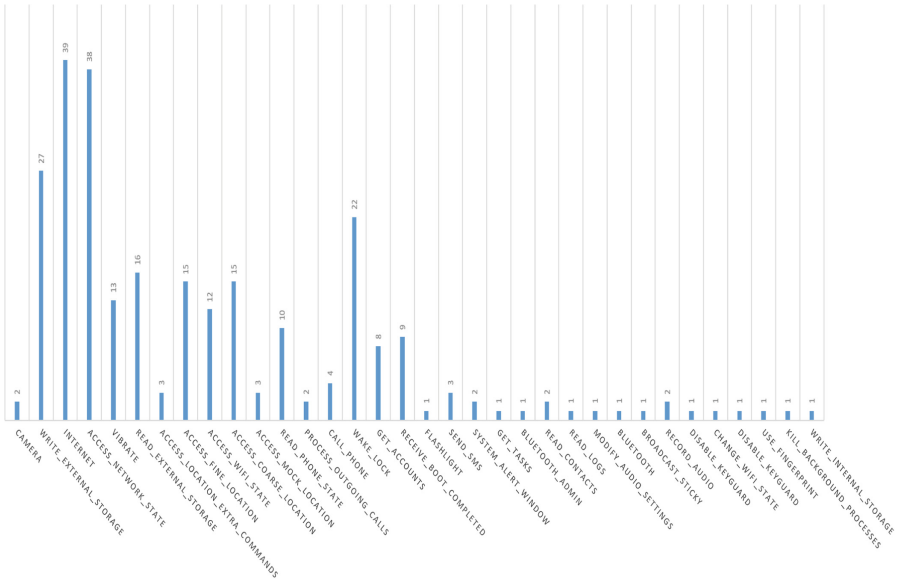


Fig. 7. Number of apps per permission

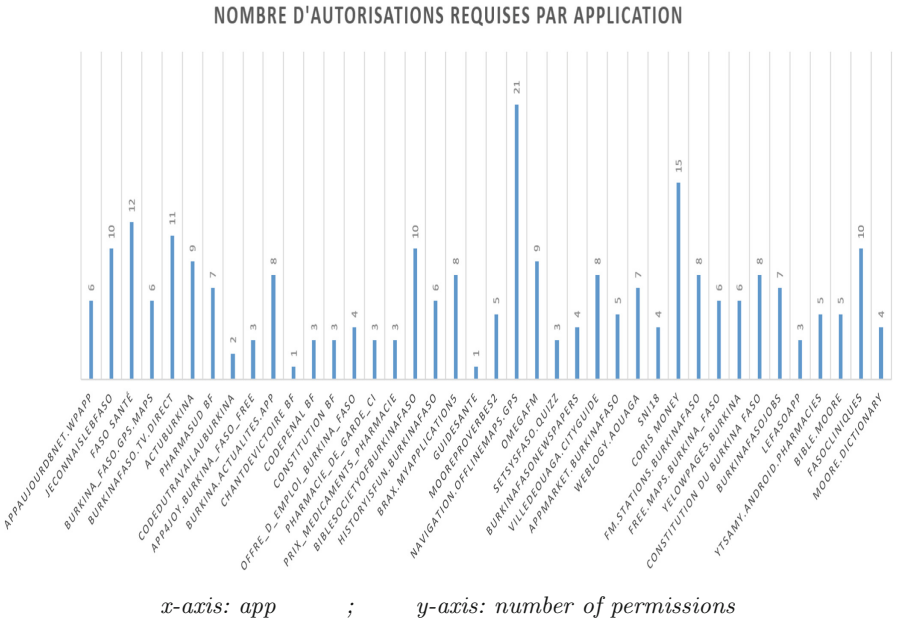


Fig. 8. Number of permissions per app

5.3 Discussion of the Results

Among the forty (40) apps analyzed, only ten (10), or 25%, have normal (risk-free) authorizations. These authorizations do not provide access to sensitive information, although some of these apps have access to the Internet. Moreover, there is only one app that does not provide Internet access. The other thirty (30) apps (i.e. 75%) provide access to various sensitive resources. Each of these apps requires at least one risky permission and thus grants access to the corresponding sensitive data. We found that 17.5% of the apps have only one dangerous permission including access to the SD card (READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE). Access to the SD card for writing is 67.5% (27 apps) that grant authorization against 40% (16 apps) for reading access to the SD card. 20% of apps grant access to accounts (GET_ACCOUNTS or GET_ACCOUNTS_PRIVILEGED). Only half of these apps (the 10%) require these permissions for their functionality. The other half therefore does not need these authorizations due to their functionalities. 7.5% of the apps allow abusive access to the messaging system (sending, receiving or reading SMS messages).

The results of this analysis show that twenty-seven (27) apps (or 67.5%) only require the authorizations necessary for their operation, compared to thirteen (13) apps (or 32.5%) that have authorization abuses that can compromise data confidentiality, integrity and even availability. Also, seven (7) apps (or 17.5%) have integrated high-risk advertising libraries.

Permission Management
Authorization management has evolved with fixes to the latest versions of Android. Since Android 6.0 (Marshmallow), permissions are required to install the app. Users ability to revoke the individual authorizations of an app via the app settings interface. However, with Android 5.1 (Lollipop) and earlier versions, the app permissions are all required for installation. Users have no choice, they must accept all the permissions of the app or refuse them all. Authorizations giving access to sensitive resources expose these resources to malware threats. The need to find alternatives is essential for the protection of sensitive resources. Also, in most of the analyzed apps, there is no expression of hardware aspects to use technical capabilities.

Permissions-Risk Link for Using an App
Two permissions can each be harmless but granted together, the risk of exposing confidentiality can increase considerably. For example, permissions INTERNET and READ_SMS, may seem harmless each, but combined we have an app that can read SMS and send content to a third party. There are several combinations of permissions that carry significant risks but are not presented as such to the user. In our analysis, the permission INTERNET_ACCESS was used by thirty-nine (39) apps of which thirty-eight (38) also have permission ACCESS_NETWORK_STATE. These two common permissions take control of the device's hardware and help the user connect to the Internet and maintain a constant connection to the network properties of the devices. Also, the permission ACCESS_FINE_LOCATION has been used by fifteen (15) apps as well as the permission ACCESS_COARSE_LOCATION. These two permissions are used simultaneously by twelve (12) apps. These two permissions provide access to user information and allow the user's location to be determined, whether it is an approximate or precise location. Therefore, the combination of permissions that control the device hardware and permissions that access user information makes the app more vulnerable to the risks of malicious attacks.

5.4 Proposals for Solutions

After analyzing apps and identifying potential threats to mobile data, we propose various preventive measures to prevent malware from infecting smartphones. The recommendations we make are intended to reduce the risks discovered from the scenarios cited and the related work.

As the main service provider, the administrator is the regulator of apps and services on mobile devices. It must therefore add to its security policy:

- extend the permission management policy to verify the need for permission by each app to limit malicious or unknown actions;
- block potentially dangerous combinations of permissions. In our case, having the permissions "INTERNET_ACCESS or ACCESS_NETWORK_STATE" and "ACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION" in the same app increases the risk of data leakage;

- improve its User-ID (UID) allocation system to prevent any reuse. Thus, he could associate the app life cycle with his UID;
- have only one signature per app to avoid its reuse by piggybacked app's developers;
- ensure that apps that offer the same features use the same permissions or permissions groups;
- have appropriate mechanisms in place to approve third-party libraries used in app development;
- have a file *manifest* containing the necessary permission group for each app category based on functionality and avoid modifying this file that developers will exploit.

The developer is primarily responsible for granting permissions through the declaration of the resources to be accessed. To reduce the risk of privacy breaches, it should respect certain principles:

- inquire about or comply with the good practices provided by Android in order to make apps less sensitive to security issues and reduce security vulnerabilities that can be exploited by malware;
- use approved libraries;
- ask for permissions related to the app's features;
- add comments on the app and its features with permissions.

The user, by accepting these permissions, will have access to different functions of his device. However, he may not understand the potential danger associated with each permission or how combinations of permissions can expose his data to attacks. The choice of authorizations for the installation is up to them in addition to being the first to be concerned by the consequences of the attacks. We offer a number of precautions for the user:

- the user must check the risks for each combination of permissions to avoid making his data vulnerable;
- it must be careful in approving dangerous permissions during the installation of apps;
- the system simply tells the user which permission groups the app needs, not individual permissions, hence the need to review the permissions granted to prevent vulnerabilities.
- the user must stop apps that access sensitive resources and also have access to the Internet to limit the leakage of their data to remote sites;
- it must disable networking features such as WiFi or Bluetooth if they are not used, to prevent smartphones from being infected by malware;
- make the updates taking into account the possible added permissions.

6 Conclusion and Future Work

In this paper, we focused on the management of permissions granted in Android; an aspect that has not been developed by this existing work. This orientation

has led to the discovery of permission abuses and the establishment of three privilege exploitation scenarios. We have discovered cases of abuse of permissions exposing confidential data. In addition, there are threats of violations through the integration, in certain apps, of ad libraries that inherit permissions from host apps. Our results can be used to reduce the threat vectors of malicious apps through improved vulnerability management policy. By focusing our approach on permission abuse and the integration of dangerous advertising libraries, we have been able to highlight that the high level of use of dangerous permissions is a risk indicator. We have therefore established a link between the permissions granted and the risk of attacks. We also identified the habits of Android app developers in Burkina Faso. For data protection, we have proposed measures for the user, developer and administrator.

Following the various security issues related to the use of identified mobile services and apps, proposals have been made to improve mobile data protection. Thus, in future work, we will address some studies that we have not been able to deepen. It is about:

- analyze a sample of apps developed in Africa to broadly study the habits of developers in Africa and the risks faced by users of these apps;
- develop a tool to warn and guide the user or administrator when an app requires a dangerous authorization or a combination of authorizations that will expose the user's data. The warning will occur during the publication of the app at the administrator's and during the installation at the user's;
- put more emphasis on data residue attacks.

References

1. He, D., Chan, S., Guizani, M.: Mobile application security: malware threats and defenses. IEEE Wirel. Commun. **22**, 138–144 (2015)
2. Thanh, H.L.: Analysis of malware families on android mobiles: detection characteristics recognizable by ordinary phone users and how to fix it. J. Inf. Secur. **4**, 213–224 (2013)
3. Wang, Y., Alshboul, Y.: Mobile security testing approaches and challenges. In: Conference Paper, February 2015
4. Sawadogo, S.: Partitionnement de Graphes: Application à l'identification de malwares, master 2, mai 2015
5. Mishra, R.: Mobile application security: building security into the development process (2015)
6. Gilbert, P., Chun, B.-G.: Vision: automated security validation of mobile apps at app markets (2011)
7. Friedman, J., Hoffman, D.V.: Protecting data on mobile devices: a taxonomy of security threats to mobile computing and review of applicable defenses. Inf. Knowl. Syst. Manag. **7**, 159–180 (2008)
8. Rezaie, S.: Mobile security education with android labs. Ph.D. thesis, The Faculty of California Polytechnic State University, March 2018
9. Zonouz, S., Houmansadr, A., Berthier, R., Borisov, N., Sanders, W.: Secloud: a cloud-based comprehensive and lightweight security solution for smartphones. Comput. Secur. **37**, 215–227 (2013)

10. Lindorfer, M., Neugschwandtner, M., Platzer, C.: MARVIN: efficient and comprehensive mobile app classification through static and dynamic analysis. In: 2015 IEEE 39th Annual Computer Software and Applications Conference, vol. 2, pp. 422–433 (2015)
11. Shewale, H., Patil, S., Deshmukh, V., Singh, P.: Analysis of android vulnerabilities and modern exploitation techniques, March 2014
12. Jimenez, M., Papadakis, M., Bissyandé, T.F., Klein, J.: Profiling android vulnerabilities (2014)
13. Mobile Threats Report, Juniper Networks Third Annual, March 2012 through March 2013
14. Li, L., et al.: Understanding android app piggybacking: a systematic study of malicious code grafting (2016)
15. Li, L., et al.: On locating malicious code in piggybacked android apps. October 2017
16. Arzt, S., et al.: FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. In: Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2014, New York, pp. 259–269 (2014)
17. Avdiienko, V., et al.: Mining apps for abnormal usage of sensitive data. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, May 2015, vol. 1, pp. 426–436 (2015)
18. Chin, E., Felt, A.P., Greenwood, K., Wagner, D.: Analyzing inter-application communication in Android. In: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services - MobiSys 2011, pp. 239–252. ACM (2011)
19. Ratsisahanana, R.A.: Caractérisation et détection de malware Android basées sur les flux d'information. Autre, Supélec (2014)
20. Calvet, J.: Analyse Dynamique de Logiciels Malveillants. Cryptographie et sécurité [cs.CR]. Université de Lorraine (2013)
21. Sang, F.L.: Protection des systèmes informatiques contre les attaques par entrées-sorties. Cryptographie et sécurité [cs.CR]. INSA de Toulouse, pp. 9–10 (2012)
22. Grace, M., Zhou, W., Sadeghi, A-R., Jiang, X.: Unsafe exposure analysis of mobile in-app advertisements (2012)
23. Dinh, H.T., Lee, C., Niyato, D., Wang, P.: A survey of mobile cloud computing: architecture, applications, and approaches, October 2011
24. Symantec, 19 August 2013. https://www.symantec.com/security-center/writeup/2013-081914-5637-99. Accessed 18 Dec 2018
25. Vulnerabilities of Android. https://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224. Accessed 18 Jan 2019
26. Gartner: Preliminary Worldwide PC Vendor Unit Shipment Estimates for 2018, January 2019. https://www.gartner.com/en/newsroom/press-releases/2019-01-10-gartner-says-worldwide-pc-shipments-declined-4-3-perc. Accessed 22 Apr 2019
27. Gartner: Worldwide Smartphone Sales to End Users by Vendor in 2018, February 2019. https://www.gartner.com/en/newsroom/press-releases/2019-02-21-gartner-says-global-smartphone-sales-stalled-in-the-fourth-quart. Accessed 28 Apr 2019