SindhUniv. Res. Jour. (Sci. Ser.) Vol. 52 (01) 49-56 (2020)

http://doi.org/10.26692/sujo/2020.03.08

SINDH UNIVERSITYRESEARCH JOURNAL (SCIENCE SERIES)

Monitoring And Controlling Access to Privacy Sensitive Resources of Android System

S. JAN, A. RAUF*, R. SAEED*, N. M. FANCY*, F. Q. KHAN**, G. AHMAD[++***], K. AZAM****

National Center for Cyber Security-UETP, Dept: of CS&IT, University of Engineering & Technology, Peshawar, Pakistan.

Received 24th September 2019 and Revised 16th January 2020

Abstract: There has been a significant increase in the use of Android platform in the last decades mainly because of the features that it offers, i.e., open-source architecture, a wide range of Application Programming Interface (API). For controlling access to resources and data, the android platform offers contains a permission management system. However, recently several third-party applications are found abusing the android permission system. Such applications misuse the granted permissions without user knowledge. Some of the researchers have proposed permission managers which revoke the third-party application permissions to stop the misuse. Although such security applications allow users to revoke the app permissions, however, it is difficult for the users to differentiate between a genuine resource access and a malicious access. In this research work, a security application has been developedthat presents useful monitoring information to usersto help them in deciding on which applications should be restricted from using the phone resources and data. It monitors applications' and systems' activities (e.g., process importance, screen on/off information) and calculates the corresponding risk to notify user about the resource access. If further enables the user to revoke the granted permissions to an android application by considering such resource access reports.

Keywords: Security, privacy, android applications, resources

1. INTRODUCTION

Today the smartphones are carrying much more data with them as compared to a traditional phone. Too many applications are available through which users can easily extend their phone features. Numerous amounts of data availability have gained the attention of hackers. Recently it is reported that google play store contains some of the applications which appear as a legitimate application but are encountered spying on user data. These applications once installed on the phone can to use the phone resources and sending user data to remote location without your consent. The spying on user datais accomplished by abusing the android permission system.

A. Android Security

To understand the importance of security for android devices, we discuss some famous security incidents and the permission systems of android.

Security Incidents of Android Devices:The number of spy applications for mobile phone operating systems generally and android framework specifically are increasing. In 2016, some applications were identified which were recording the data of users from the usage of the resources in the phone and further and sending them to remote server by misusing the android permission system. Trend Micro in July 2016, discovered an updated version of a very popular game GO Pokémon(Micro 2017) which contained a trojan inside it and available at third-party app stores. The trojan could access the SMS, calls, phonebook, camera, audio recorder, Gmail account and could even turn on or off the Wi-Fi of a mobile phone.

Similarly, in Aug 2017, two applications SonicSpy(Micro 2017) and FakeToken(Micro 2017)wereidentified at google play store deploying the same attack of recording personal information and sending it to remote servers. SonicSpy was integrated into messaging applications and has affected more than 4,000 mobile systems before they get detected. A “FakeToken” was inserted into buy-a-ride application that would keep recording the user’s financial credentials when they pay for a ride in the application. Later on, it was identified as mobile ransomware. In Sep 2017, another application as GO Keyboard(Micro 2017) was reported which could record a user information and transmit it to a server in China. It could even execute the code from a remote server on your mobile device.

Android Permission System: In android OS, the permission management system controls the access to resources and data. An application explicitly declares the permission for the resources they want to use, in the app manifest file. The application can request the permissions at install time or at runtime.

[++]Corresponding Authoremail: gulzar@uetpeshawar.edu.pk, sadeeqjan@uetpeshawar.edu.pk
* Department of Computer Science, University of Peshawar, Peshawar, Pakistan.
** Dept: of IT, Faculty of Computing & IT, King Abdulaziz University, Jeddah, Saudi Arabia
*** Dept: of Electrical Engineering, University of Engineering & Technology Peshawar, Pakistan.
**** Dept: of Mechanical Engineering, University of Engineering & Technology Peshawar, Pakistan.

In Android version 5.1.1 (SDK 22) and previous versions, users are asked about the permissions when installing an application. They can either accept all permissions or reject them, however, in the latter case, the application installation will fail. There is no option to customize the permissions, i.e., accept some while reject others.

In Android version 6.0 (SDK 23) and above, the new concept of runtime permissions is introduced which resolves these issues, however not completely. Users are asked about granting a permission at runtime which can either be allowed or denied. However, the user is held responsible in this case and when they have limited knowledge of permissions, they might allow all of them to get the application installed quickly. Users do not know if the application is genuinely using the resource or not. Therefore, it is difficult for him to decide whether to allow or deny the permission. There is a need to provide users more useful information through which he can differentiate between genuine or malicious resource access.

B. Problem Analysis and Motivation

For the proof of concept, we have conducted an experiment by using penetration testing tools. The tools we have used are Kali Linux penetration testing distribution (Tedi Heriyanto and Lee Allen 2014)., Metasploit framework (Kennedy et al. 2011) and some real android devices. A spy application is generated using payloads already present in Metasploit framework. This spy application is then packaged with a genuine android application with the help of Apktool. The final packaged spy application is installed on two android real devices with different versions.

It is observed on the first device that the spy application work as genuine application and the user will not be aware of spyware working at the backend. The spyware is able to collect data and send to a server on request even when the phone screen is off and the phone is locked.On the second device, it is observed that the permission list is displayed when the application starts using the resource once which might put user to think why it is asking to use camera when I have not initiated it. However, if the user once allows it will not ask again for the permission and will start using the resource. Hackers may set the interface to ask permission for once.

With this observation, we have concluded that most of the spy applications act as genuine. They ask permissions genuinely and afterward misuse them. They can even operate when the screen is off and the phone is locked. They are usually initiated by the internal commands and not by user interaction.

2. LITERATURE REVIEW

Previously a lot of research is carried out to detect and prevent phone resources and data from spy applications. For handling permission abuse, permission managers havebeen introduced by using four different techniques Apk modification, Customized ROM, Hooking technique and AppOps API (Tools 2014).The solution based on Apk modification normally work in a fashion to disassemble android applications, perform the required changes by the users, reassembles and finally reinstalls the updated version. Customized ROM based solutions update some of the core functionalities of android operating system to achieve desired results.Another popular method is called Hooking where the running code's is modified to interrupt the normal execution of a process. Google also introduced the AppOpsAPI in Android 4.3. It has an application at the user interface level enabling the users for revoking application permissions for various resources in the phone dynamically.

Advanced permission manager(Tools S. 2016)works by modifying .apk files to update the permission list already defined inside the application. Appguard application(Backes et al. 2013) comes with predefined security policies. The apk files are extracted to "classes.dex" by the process of decompilation and rewrites of some of the code. In(Do et al. 2014) an ideal permission removal system is presented. The system decompiles the apk file and extracts java code file from it rather than dex code files like in(Backes et al. 2013). This is because removing permission dependent code from java files is quite easy then from dex files. A privacy-enforcing framework is presented in(Neisse et al. 2016) that decompiles the android apk and extracts the byte code. The byte code and the concrete security policy set by the user arepassed to the instrumentation system, which generates the instrumented byte code.

CyanogenMod(Kondik 2016) is a user-defined ROM consisting of several functions along with the permission control system through which the users can grant/deny various permissions when required. The APEX(Nauman et al. 2010)framework enables users for granting permissions and imposing constraints on various resources based on the defined/enforced policies. Another extended version of Android is MockDroid(Beresford et al. 2011) that can present the resources to users with several options, e.g., availability, fake value. In addition, AppFence is also introduced which aims to impose privacy controls by providing "shadow data" in place of real data to android applications. AppFence works by substituting the data with virtual "shadow data" as per user request. For example, when an application requests for a contact lists, it may receive an empty list. Another application called Permission Tracker(Kern & Sametinger 2012)gives an overview to the users about the granted permissions, observe the resources for various

access requests and grant/deny such requests. Middleware for permission managers(Wang et al. 2015)is based on user-defined ROM providing user interfaces for controlling permissions and controlling the potential violations of permission usage. Ensuring Privacy System(Constantinescu 2015)utilizes the framework for helping in hooking or extending/replacing various functions. It can work by revoking the permissions as well as enabling users in providing fake data to other unwanted applications regarding permissions. DelDroid(Hammad et al. 2017)system works by detecting applications that have already received unwanted permissions and revoke them by applying the lease privilege principles.Similarly, another application called Identidroid(Shebaro et al. 2014)enables users to control access to their data/resources as well as helps in anonymizing it with fake values.

LBE (Lets Be Elite) system is based on root-level controls(Tools 2015) to monitor applications and notifies users about any access to potentially sensitive resources. Using this application, run-time access to resources can be controlled by users. Real-time monitoring system(Li et al.2015)works by injecting proxy server into android code for monitoring permissions and other potential violations of privacy. Patronus(Wang et al. 2015)is used for the detection of potentially harmful codes (e.g., viruses) by analyzing transaction data. Similarly, DeepDroid(Wang et al. 2015) is a security application that works by defining how various resources should be used. Their system is implemtned via instrumentation of memory of sensitive processes.

FireDroid(Russello et al. 2013) is another application for enforcing security policies in android. In this application, ptrace utility is used for monitoring processes for security purposes. The system works by intercepting zygote process and init.rc file.

PrivacyMod(Silva et al. 2015) and AppOps Starter (Tools 2014)are extender versions of the Google API AppOps for monitoring/controlling confidential data for privacy reasons. The systems offers run-time notifications, logging permissions and enabling users for grading/denying such permissions. AppOps API based solution(Silva et al. 2015, Tools 2014)is similar in nature and dependent on the extension of API or to hook it as it has been deprecated for 3rd-party apps.

3. MATERIALS AND METHODS

The security application presented in this research consists of two system modules: Monitoring System (MS) and Controlling System (CS). The MS monitor the selected privacy sensitive resources shown in Table 1. The security application monitors to check for the user consent about the resource whether in foreground or background. It creates report for selected privacy sensitive resources and display it to user.

TABLE 1 MONITORED ANDROID RESOURCES

RESOURCES	PERMISSION
Camera	CAMERA
Location	ACCESS_COARSE_LOCATION ACCESS_FINE_LOCATION
Microphone	RECORD_AUDIO
Telephony	CALL_PHONE PROCESS_OUTGOING_CALL READ_SMS RECEIVE_SMS SEND_SMS WRITE_SMS
Contacts	READ_CONTACTS

For selecting the privacy sensitive resources, we have categorized the resources into three groups that are hardware resources, software resources and data shown in Table 2. Then the resources in each category are prioritize on the basis of the risk they impose if assigned to a third-party application. The resources which are assigned the 1st priority is considered as privacy sensitive resources. At the end few resources from each category are selected for monitoring. The Appendix A: Potentially risky permissions from(Nauman et al. 2010) is also considered for selecting the resources for monitoring.

Architecture of the Proposed Tool:As depicted in Figure 1, the architecture of our proposed tool consists of the following two components:

- The Monitoring System (MS)
- The Controlling System (CS)

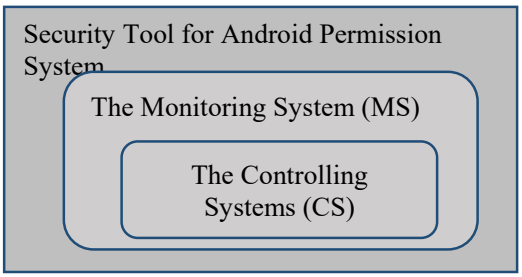

Figure 1 Architecture of the Proposed Tool

TABLE 2 PRIORITIES OF RESOURCES

Resources	Priority 1	Priority 2
Hardware Resources	Camera	NFC
	Gps	Speaker
	Microphone	Battery
	Memory	Vibrator
	Network	Flashlight
	Bluetooth	
	Keypad	
Software Resources	SMS Service	Maps
	Call Service	Calendar

	Mailing applications	Notes
	Social media	Widget
Data Resources	Contacts	Phone state
	Message log	Account info
	Call log	Network state
	Files	Synchronization info

A. The Monitoring System (MS)

The Monitoring System is a system module of security application that decides about the apps using the resources in a legitimate way and those which are using it in an illegal way. For differentiating between these two types of applications, we monitor security relevant tasks of the applications. Such information can be used by users to identify which apps are harmful and misusing their resources so that they can be uninstalled or stopped.

The MS module generates reports containing information about resources that can be presented to users. The data included in such reports depict the status of applications when they are using various types of resources.

Application Name:The application name is mentioned in the report which use various types of resources. For instance, a user may select the speaker resource to get information about speaker usage. A list of applications will appear which have used the speaker resource.

Date and Time of Access:The report also include the date and time at which a resource e.g., speaker was used by a particular application.

Duration:Duration for which the resource was used by a particular application is also an important piece of information included in these reports. The duration is recorded by using Dumpsys tool. Dumpsys tool is used to record the AppOps service. The facility of searching the record is also available to extract the duration for which the resource was used by a particular application.

Date and Time of Release:Resources are utilized by applications for certain duration of time and then they are released. Our reports also include the date and time at which a particular resource is released by an application. The time of release is also recorded by dumping the AppOps service file using Dumpsys tool.

Screen Information:Another important feature that is recorded in our reports is about the screen on/off, i.e., it records if the mobile screen was switched off or On when a resource was accessed. It further stores the locking status of the device when the resource was accessed. There are four possibilities as listed in Table 3. Such information is stored via APIs (Power Manger/Keyguard Manger).

TABLE 3. SCREEN ON/OFF INFORMATION

Screen	On	Off
Lock	Screen on and lock	Screen off and lock
Unlock	Screen on and unlock	Screen off and unlock

Window Information:Recording about window is also an important information. This basically represents the contents of on the screen when the resource was accessed. Such information helps to identify how user was interacting with the application and the platform. It includes the information about the focused window at the time of resource access. It also helps to identify whether the resource was accessed in the background or foreground, e.g., whether the requesting application is on focused window or not. The tool Dumpsys is used for recording this type of information.

Process Information:The last piece of information that we record in the reports is related to the executing processes. The class "API RunningAppProcessInfo" is used to collect the importance level of various processes as listed in Table 4 with specific meanings. The importance level of each process is assigned to foreground/background. In case of the foreground, the process is running and user is interacting with it. On the other hand, for background, the process is executing however the user is not interacting with it. Such information can be used to describe the behaviour of the application.

TABLE 4. RUNNING PROCESSES AND THEIR IMPORTANCE

Code	Importance Based on User Interaction	Importance	Description
100	Foreground	Foreground	The process is on top of the screen through which a user can interact.
125	Foreground	Foreground Service	The process is not on top of the screen but is running and user know about it.
200	Foreground	Visible	The process is visible to the user but not as the immediate window. It may be behind the current user interface.
325	Foreground	Top Sleeping	Process is on top of the screen, but the device is asleep therefore is not visible to user.
300	Background	Service	The process runs as a background service and the user is not aware of it.
400	Background	Background	The process is cached and does not have any active running component.

As listed in Table 4, the initial 4 scenarios are related to foreground. The reason is that they are all

in the knowledge of the user. On the other hand, if we look at the last 2 cases, they users may not be aware of them and therefore they are classified as background. For recording such type of Process information, we use the class RunningAppProcessInfo API. The class can be used to collect all type of relevant information about a process.

When we combine all these features, we get a good overview of the application's behavior when it is using a resource. This information is then used by the report generator for making report of all modules. Such information is further passed to other classes (e.g., interface) to present it to user when the application is opened.

The proposed work presents a policy based on risk calculation for notifying user about resource access.However, it is important to note that when the notifications on each resource access also affects the stability of the system. In this research work, we present a policy on how to notify events whilemaintaining the stability. Our proposed policy clearly distinguishes when it is required to notify the user and when it is not required when a resource is accessed. In addition, the policy also defines when there is no need to notify users regarding the events. A value is calculated based on the risk level and monitoring results for resource accesses. Further, a value called threshold is also defined.

For every feature, we have wo possibilities, i.e., the normal and the abnormal. These possibilities are listed in Table 5. To calculate the risk, a value of 1 is added for every abnormal outcome. In case of a normal output, the user notification is suppressed.

- The value of 1 is added to risk level when the process importance is unknown (in background)
- In case of the screen off event, 1 is added again to the risk level.
- In case of the phone lockage, the value of 1 is added to risk level.
- In case of non-focused window, the value of 1 is added to the risk level.

These cases give rise to total sixteen scenarios. In Table 6, 7 of such cases are listed as the other cases are not possible. This is because when the phone's screen is off or it is locked, then the focused windows cannot be there for the application. We have defined the threshold to 2 which means the user will be notified if the risk level >= 2.

B. The Controlling System (CS)

Through the process of controlling, we take actions when the applications behave in abnormal ways. Users of a mobile applications should have the ability to control how their personal data/resources are used by applications. Such actions can consist of Allowing or Disallowing access to resources by users. Our developed application presents a spinner widget button to users where they can opt for allowing or disallowing for using their resources or whether to present fake data to the application. The default value is set to "Allow". Incase of the user selection of "Disallow" to a specific resource, the particular application can no longer use that resources.

TABLE 5. MONITORING FEATURES BEHAVIOUR

Features	**Outputs**	
	Abnormal	***Normal***
Process	Background/Unknown	Foreground
Screen	Off	On
Phone	Lock	Unlock
Window	Not Focused	Focused

TABLE 6. POLICY BASED RISK CALCULATION

No	**Cases**	**Risk Calculation**	**Risk level**	**Notification**
Case 1	Process(B/U), Screen(off), Phone(L), Window(NF)	Process(B/U) +1 Screen(off) + 1 Phone(L) +1 Window(NF) +1	4	Notify
Case2	Process(B/U), Screen(off), Phone(UL), Window(NF)	Process(B/U) + 1 Screen(off) + 1 Window(NF) + 1	3	Notify
Case 3	Process(B/U), Screen(on), Phone(L), Window(NF)	Process(B/U) +1 Phone(L) +1 Window(NF) +1	3	Notify
Case 4	Process(B/U), Screen(on), Phone(UL), Window(NF)	Process(B/U) +1 Window(NF) +1	2	Notify
Case 5	Process(B/U), Screen(on), Phone(UL), Window(F)	Process(B/U) +1	1	Do not Notify
Case 6	Process(F), Screen(on), Phone(UL), Window(NF)	Window(NF) +1	1	Do not Notify
Case 7	Process(F), Screen(on), Phone(UL), Window(F)	Null	0	Do not Notify

4. RESULTS & DISCUSSIONS

Easy to use Interface:In this research work, we have developed an easy to use and efficient interface for the android application. There exist many security solutions for android, however, their use is limited because of the complexities involved and the requirement of customized ROM. Our

developed application does not require any customized ROM and can be installed by any user by just downloading it and making a few clicks without any pre-requisites.

Data and Resource Monitoring:Our developed application contains a monitoring module which observes all requests made from various applications to resources. The information about start/stop time of resource usage is also recorded.

Access Information Record:All information about the accesses to resources are recorded. Such information contains foreground/background context and user consent/interaction with the application etc. Such information can be used for deciding the nature/behavior of an application i.e., its genuineness as resource misuse is often done in (Micro 2016, Micro 2017, Micro 2017, Micro 2017).

Balance between Usability and Event Notifications:Access to resources are controlled in real-time by our developed application. However, notifying the users about each resource access and asking them for allowing/disallowing these accesses often annoys the users and affects the usability. To address this issue, we have developed a policy for event notification, i.e., the user will not be notified if he/she has explicitly made interactions with the application. The user will be asked only when the application request to use a resource without the user interaction.

Comprehensive Reports:Our developed application offers comprehensive reports about the use of each resource. It consists of the list of applications which have used that particular resource, the date/time of resource access, start/end time of usage etc.

Access Controls of Resources/Data:The main aim of our developed application is to block applications for using resources without the user consent or give the application fake data. It provides full control of resources to users, i.e., in case a user selects to block a particular application from using a resource, that particular application can no longer use the resource. The developed application gives options to user for blocking the application completely or allowing with fake data for a particular resource request. For instance, if the microphone resource is blocked for some application e.g., MS word, the microphone can no longer be used by MS word anymore. On the other hand, if we allow the resource with fake data, the application will be presented with the fake data and in this way the abnormal termination of the application can be avoided.

5. CONCLUSION AND FUTURE WORK

Mobile applications often misuse our confidential data and resources without our explicit consent. There exist applications which look genuine at first sight however, they may contain malicious code for sending our sensitive information to hackers or misusing the phone's resources. It is often a challenge for the users to identify whether an application is using the resources as per its need or maliciously. A security monitoring application that can monitors each resource access in real-time and notifying the user about it will adversely affect the usability.

This research has presented a security application thatmonitors how applications are using selected privacy sensitive resources.When resources are used by an application, our monitoring security application records the features and based on the pre-defined policy, it makes a decision if the user should be notified for this event or not. The use is not notified for each resource access and therefore keeping intact the system usability. The application generates comprehensive reports about resource usage. The user is notified for any type of potentially malicious access to resources or data. With the help of the reports, users are in a better position to decide about malicious and legitimate access to resources. The controlling module of the security application helps the user in revoking application permissions when required.

Our developed security application provides resource-based reports to users. The application can be extended to application-based interface. Application-based reportswill include resources accessed by that particular application and their details. In future, recording more features for all resources can be added that will differentiate the genuine application form malicious application more precisely. The security application presented in this research can be updated for the newer android version.

Another possible future work is to extend the developed security application to monitor more resources of the android system. It can also be enhanced by setting a variable risk threshold value based on the android user. The variable risk threshold value will help to decrease the false negative and false positive rate of the application.

6. ACKNOWLEDGMENT

This research is supported by the Higher Education Commission (HEC), Pakistan through its initiative of National Center for Cyber Security for the affiliated Security Testing-Innovative Secured Systems Lab (ISSL) established at University of Engineering & Technology (UET) Peshawar, Grant No: 2(1078)/HEC/M&E/2018/707.

REFERENCES

Backes, M. (2013)"Appguard--enforcing user requirements on android apps",s.l., s.n., pp. 543-548.

Beresford, A. R., Rice, A., Skehin, N. & Sohan, R., (2011)"*Mockdroid: trading privacy for application functionality on smartphones"*,s.l., s.n., pp. 49-54.

Constantinescu, A. S., (2015)"Ensuring privacy in the android os by hooking methods in its api". *Journal of Mobile, Embedded and Distributed Systems,* Volume 7, pp. 107-112.

Do, Q., Martini, B. & Choo, K.-K. R., (2014)"Enhancing user privacy on android mobile devices via permissions removal".s.l., s.n., pp. 5070-5079.

Hammad, M., Bagheri, H. & Malek, S., (2017)"Determination and enforcement of least-privilege architecture in android".s.l., s.n., pp. 59-68.

Hornyack, P. et al., (2011)"These aren't the droids you're looking for: retrofitting android to protect data from imperious applications".s.l., s.n., pp. 639-652.

Kern, M. & Sametinger, J., (2012)"Permission tracking in android". s.l., s.n., pp. 148-155.

Kondik, S., (2016)"*Welcome to Cynogenmod"*. [Online] Available at: http://www.cyanogenmod.org/ [Accessed 18 March 2019].

Li, S. (2015)"Real-time monitoring of privacy abuses and intrusion detection in android system".s.l., 379-390.

Micro, T., (2016) "Malicious Version of Popular Mobile Game Pokemon Go App Spotted." [Online] Available at: https://www.trendmicro.com/vinfo/no/security/news/mobile-safety/malicious-version-of-popular-mobile-game-pokemon-go-app-spotted
[Accessed 18 March 2018].

Micro, T., (2017)"FakeToken Android Banking Trojan Returns as a Ride-sharing App", Trend Micro Inc. [Online]
Available at: https://www.trendmicro.com/vinfo/no/security/news/mobile-safety/faketoken-android-banking-trojan-returns-as-a-ride-sharing-app
[Accessed 2019 March 2019].

Micro, T., (2017)"GO Keyboard Apps Collect and Send User Data to Remote Servers". [Online] Available at: https://www.trendmicro.com/vinfo/us/security/news/mobile-safety/go-keyboard-apps-collect-send-user-data-remote-servers

Micro, T., (2017)"SonicSpy Android Spyware Found in Google Play". [Online] Available at: https://www.trendmicro.com/vinfo/no/security/news/mobile-safety/sonicspy-android-spyware-found-in-google-play
[Accessed 18 March 2019].

Nauman, M., Khan, S. & Zhang, X., (2010)"Apex: extending android permission model and enforcement with user-defined runtime constraints". s.l., s.n.,.

328-332.

Neisse, R., Steri, G., Geneiatakis, D. & Fovino, I. N., (2016)"A privacy enforcing framework for Android applications". computers & security, Volume 62, pp. 257-277.

Russello, G., Jimenez, A. B., Naderi, H. & Mark, W., (2013) "Firedroid: Hardening security in almost-stock android". s.l., s.n., pp. 319-328.

Shebaro, B., Oluwatimi, O., Midi, D. & Bertino, E., (2014)"Identidroid: Android can finally wear its anonymous suit". Transactions on Data Privacy.

Silva, P., Amorim, V. J. P., Ribeiro, F. N. & Muzetti, I., (2015)"Privacymod: Controlling and monitoring abuse of privacy-related data by android applications".s.l., s.n., pp. 42-47.

Sun, M., Zheng, M., Lui, J. C. S. & Jiang, X., (2014)"Design and implementation of an android host-based intrusion prevention system".s.l., s.n., pp. 226-235.

Tools, S., (2016)"Advanced permission manager". [Online]
Available at: https://play.google.com/store/apps/details?id=com.gmail.heagoo.pmaster&hl=en
[Accessed 18 March 2019].

Tools, L., (2015)"LBE (ROOT)". [Online] Available at: https://play.google.com/store/apps/details?id=com.lbe.security&hl=en
[Accessed 18 March 2019].

Tools, O., (2014)"App Ops starter". [Online] Available at: https://play.google.com/store/apps/details?id=com.schurich.android.tools.appopsstarter&hl=en
[Accessed 18 March 2019].

Wang, D. et al., (2015)"A secure, usable, and transparent middleware for permission managers on Android". IEEE Transactions on Dependable and Secure Computing,Volume 14, pp. 350-362.

Wang, X., Sun, K., Wang, Y. & Jing, J., (2015)"DeepDroid: Dynamically Enforcing Enterprise Policy on Android Devices". s.l., s.n.

Kennedy D, O'Gorman J, Kearns D, Aharoni M (2011): "Metasploit: the penetration testers guide". No Starch Press.

Tedi Heriyanto, Lee Allen (2014). "Kali Linux – Assuring Security by Penetration Testing.", Packt Publishing Ltd.