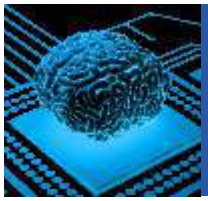

Sountra et al LGU (IJECI) 2019

LGU (IJECI)
ISSN: 2522-3429 (Print)
ISSN: 2616-6003 (Online)

LGU International Journal for
Electronic Crime Investigation

Research Article | Vol. 4 Issue 2, April - June 2020

Review paper on Android Application Security

Muhammad Arslan Tariq[1] ,Wahid Qayyum[2] ,Rehmatullah[3]
University of Bedfordshire UK[1],
University of Sargodha,Lahore[2],
University of Sargodha,Lahore[3]
Muhammad.arslann@outlook.com[1],
wahidqayyum5@lgmail.com[2],
rehmat.ravian@gmail.com[3]

Abstract

The revolution of application market involves smart phones security. There are many problems that occur in real world regarding android application security. Hackers everywhere in the world, are stealing data for their use that has led to great loss to the business world. This analysis will uncover the use and misuse of phone identifiers, issues in android applications security and how to improve the android application security.

Keywords: android, app security, security, attack, hacking, mitigation, mobile application.

1. Introduction

The rapid or instant growth of smart phones usage has brought a renaissance for the mobile services. Where ever you go you will definitely find smartphones there and those phones consist of applications that provide services which include social distancing, messaging, video chat and gaming etc. Android application market such as Google play store and Apple's iOS store provide a vast variety of free and paid applications. These application face enormous daily challenges like security issues, malwares etc. Users desire the application to be safe and usage friendly, however, markets or developers somehow fail to provide security at its best, that has led to many problems. Many developers have failed to securely or fairly use Android APIs that leads to leakage of personal information. In this article, we have characterized the android application security [22].

Android is a Mobile operating system with a user interface based on a Linux kernel, most of the android applications work on touch mobile or smartphones and on tablets as well. The source code of the android is provided by the Google (under open source license) to different mobile companies like redmi, samsung etc that uses it as mobile operating system. IOS is an operating system developed by Apple used on iPhone devices. IOS apps are around :1,567,356 and Android apps :1,234,976 available in the market. While using those apps one should be careful about the security issues, security of our personal data , information stored in our mobile phones. The most commonly used android virus or malware is that text message or SMS that is sent to unknown persons without any knowledge of the user and sending someone's personal information to the third parties. Root kit is type of malware that provides the personal rights or administrative rights to the hacker or third parties. There are some root kit detectors used to avoid these malwares [1].

Every MOS pprovides file and resource permission control to secure inter process

communication. Memory is protected by Address Space Layout Randomization.

I. Structure of Android OS

As we see in the figure below the android architecture consist of four layers.

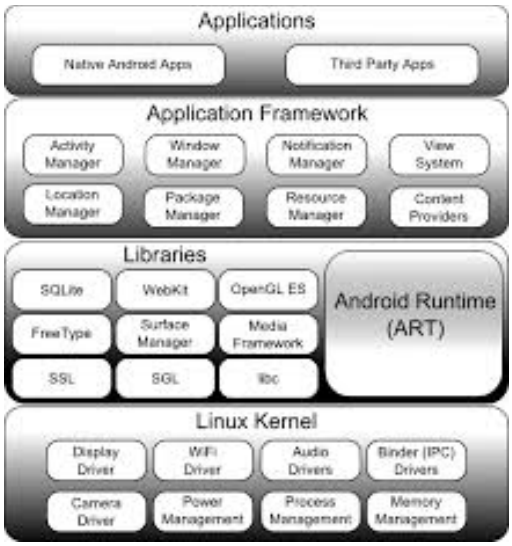

Fig1. Android Runtime [16]

A. Application Framework Layer

This layer is used to provide services of high level for application development in Java classes form. Application developers are allowed to use these classes in their applications. These services include location manager, activity manager, notification manager, content provider and so on.

B. Libraries

The Android operating system provide different libraries which may in include C or C++ libraries. Some components of the system can use or utilize these libraries. These libraries are accessible with application framework.

C. Android Runtime

It usually deals with the assortment or compilation of android app with the help of Dalvik Virtual Machines (DVM) that provides optimized code for smartphones.

D. Linux Kernel

Linux kernel is the last part or layer of the architecture which have direct link or communication with the hardware and provide some services like security, Inter Process Communication (IPC) and so on.

II. Secuirty Model

Android has a lot of predefined permissions. These permission should be satisfied by the manifest file of the app. When the application is being installed all the permissions are displayed and are visible to the users, if the users accept it then the application is installed, otherwise it is rejected. But users have to accept the application permission whether some of them are unnecessary, because the application will only be installed if permissions are given [12].

Javed Parvez build an Advance Security Framework (ADF)which apply Advanced Encryption Standard (AES) to all the files stored in device or media, If any app is found dangerous or malicious app that want to change or modify the content of device, it is restricted to be installed on the device. Malware or virus protections are done using these techniques follow as:

1) Heuristics
2) Hashes
3) Signature matching

A new and advanced security technique is introduced in package installer by Aparna Bhonde, this technique will ensure that Android will able to identify the virus or malware carrying app before the installation. Installation package manager is introduced to ensure the app is installed safely without harming the device.

A. Reverse Engineering

Reverse Engineering is the process in which users and programmers get source code of the app using apk tool. It can be done by the following tools such as 7zip, apktool and

dex2jar etc. It was observed that source code of most of the apps was analyzed using dynamic and static analysis, the results show that most of the data was leaked, like location of the person through Email. From this reverse engineering of the app, we consider a strong security technique is required for the testing and security of the app. It was known by reverse engineering that many permissions were of no use that some app require, in this way personal data is stolen. Reverse Engineering also found many security issues in source code of the application [11].

B. TLS / SLS (TRANSPORT LAYER SECURITY /SECURED SOCKET LAYER)

Many applications have to connect to the server for usage. For example, login area/access, has to get data form server. This part of communication can go through Man in The Middle (MITM) attack and it can steal data from the servers and may leak it later on. Even SSL and TLS are secured but some attackers can win against them. Sacha Fahl developed a tool Mallodroid, this tool uses static analysis of code to detect the apps that uses TLS/SLS incorrectly and inadequately [1].

Security at time of development:
We already know that no app is perfect, it becomes better with time and updates. There are lot of bugs in apps, some of them occur or are caused at the time of designing of the application and some of them are caused by the developer itself. Due to those bugs or malwares personal data is compromised and can be attacked by the hackers or attackers. In android, the responsibility goes to the developer to isolate its application form the other system app or resources by sandboxing. App sandboxing in Android is controlled by the app itself and the permissions given by the user to continue using the application. We should give permission to those app which we trust, these can improve the security. Each app has its own permission and sandboxing [2].

C. Security at publishing app to the market

Publishing means that when the app is complete or build by the developer and tested. It's time to give it or distribute it to the users for use. Mostly the app is published on Google playstore and iOS app on apple iTunes or apple store. Before publishing every app, it is to be signed. Signing is the process in which each app is provided a key. Two types of keys are there one of them is debug and the other is Release. If the developer or programmer wishes to publish the application on the Google store, they need to have access to the developer console. Developer console is a tool that allows the programmer to have control and monitor the apps. Users can download similar apps available on the play store through websites but some of the apps on website don't provide certificate of security that can harm your device. On the other hand, developer console first checks the flaws and issues that are in the app, if any error or security issue are found, google will not accept it [7].

III. ANDROID APPLICATIONS SECURITY ISSUES

Android security issues are classified into 4 major areas:

1) Data protection
2) Intellectual property protection
3) Secure authentication
4) Code vulnerability

A. Data protection

Mobile devices are posing a challenge against the data vulnerability due to its dependence on data of transmission over the air. In enterprise the complexity of risk is more other than other apps like game apps because the enterprise apps may consist of personal or financial information and demographics too. Losing this data may have direct/indirect impact on the company repetition or revenue. Losing the data of the users may result in loss to the extermination of business or company. Losing sensitive information or demographics detail may result in legal cases/issues. Android applications can be developed as web application and native applications. Androids also provide permission mechanisms and these mechanisms do not indicate data policies that can be adopted by the apps. We have considered the data protection issues into 4 parts [8].

i. Local data store

Most of the Android operating system or mobile operating system consists of small or lightweight databases in stack. Application only stores important or useful data in the stack. If the data is clear text and also stored in database, it can pose a serious threat and it may result in loss of data also attackers can use this data and can cause malfunctioning of the apps [14].

ii. Cache Usage

Cache is important thing and it may enhance the customer experience about Android operating system. Cache can be different at multiple level of the architecture. Most of the time web application usually uses cache data. If some critical information is stored in cache like some plain text, some phishing app can cause harm to cache data like tampering of data or loss of data.

iii. Data Sharing

Data loss may take place if the application is not partitioned correctly. Virtual machines are used to isolate apps from other applications. However, if the data is stored in external medium or removable storage in unencrypted form, data vulnerability can occur. In such cases the apps can still become vulnerable to attacks from hackers.

iv. Transit Data

The other big risk is data on transit. Securing the transmission data over network is also main focus of industrial focus. The major part of protecting data transit is on the network. The loss of data can happen during request over unsafe SMS protocol or data request through TCP/IP protocol [18].

B. Intellectual property protection

The second big reason of loss of data or information is accessing apps through un authorized means. Android uses Apk (Android Application Package) extension and apple uses IPA (IPhone Application) to distribute or publish the application over the Play store and App store, since the binary of the application is easily available hackers can access the app source code through reverse engineering easily [35].

i. Critical information hard coded

Having code access can reveal user credentials and crypto keys as well. If some apps deal with the payment, then it may lead to exposure of the payment method details that can cause monetary impact. Leakage at this level can put the company in danger.

C. Secure authentication

In commonly used architecture of Android app, the app connects with middle layer of architecture which connects with the main or core system at backend. User authentication is important at backend system and it is needed to be planned in such a way that it poses no risks to the security. There are some risk factors that can appear in improper handling of data between middle layer and the app [24].

i. Session Management

Session are type of setup between middle layer and the mobile application to handshake information, master data and status. These sessions are needed to be handled carefully to avoid session fixation attacks and session hijacking.

ii. Password Management

The app authentication must be secure in such a way that if the credentials are lost, it didn't cause any trouble to the user in having access to the application. Setting the password is a good attempt and the password should be complexed that it can't be predicted.

D. Code vulnerability

Vulnerability in the apps coding and design can lead to big security issues. vulnerability includes [28]:

i. Validation

The Android application uses script languages for the front-end development. As scripting language can be modified which is very risky. The data in the front end goes through the validation process. Similar looking UI are there

to change the scripting language so data validation is not secure enough. SQL attacks are possible in Android applications that can be a threat to the business companies.

ii. Exception Handling

There is a possibility that data can be leaked if exception handling of the code is not properly managed. Business logic is exposed to the users by raw exception dump. It increases the chance for attacker can identify areas to attack the application.

IV. RISKS MITIGATION

With some techniques we can reduce or mitigate the risk to a larger extent [29].

a. Data protection

We have mentioned about the issues regarding the data in the devices and application. In this section we will discuss how to remove the threats and risk with some approaches.

b. Local data store

Application stores its data on local data stores or lightweight databases like SQLite. The design of the application has to be made in such a way that duration and size of data should be minimized. Data audit is used to check the data criticality that remain in mobile for short interval of time. It should be ensuring that the local database modification can only happen using the application code. This process can avoid data avoid data modification from the external sources and stop malfunctioning of application.

c. Cache usage

There is some app that stores their data in cache for the better use of that application. If the data is stored in cache there is a chance that reverse engineering will not work. That can reduce the leakage of data.

d. Data Sharing

The data should be partitioned in their required boundaries, including images of data in document can help reduce decision time for application. The application can be partitioned in areas based on vulnerability and sensitivity of data. Some mechanism should be there to check that secure data is not moving to insecure place.

e. Transit data

Data encryption should be there to securely transfer data over the air. Data communication must be done through HTTPS instead of HTTP. SSL should be used to send any data including sensitive information, tokens and session data. SMS that are sent through the apps should also be encrypted. Key exchange can help in SMS encryption process.

f. Intellectual Property protection

We have discussed this issue earlier in a topic, now we will discuss the approaches to mitigate those risks:

g. Reverse Engineering

There are some compilers that are used to generate the source code using binary coding or byte code. This code can create issues with debug information like class details, file name and line number etc. The programmer can decompile the code and create its own high-level code using decompiled code, which is not a good thing. To avoid loss of information or data via reverse engineering, obfuscation must be done. Obfuscation is a technique that is used to remove debugging information as well as object name and can be replaced by meaningless words or entities without disturbing the behavior of application.

h. Critical Information Hardcoded

It is better that your information is hardcoded in the code, it should not be limited to user credentials, crypto keys and some other important information which include credit and debit card details. The crypto keys should be protected by the play store or where-ever the application is published instead of saving it with application code.

i. Secure Authentication

We have discussed about the secure authentication earlier. In this section we will discuss about technical ways to reduce the risk.

j. Session Management

Session ID is used for transaction between middleware and android application, this can avoid session attacks. We can add some additional and unique information in session ID that can identify the users and devices so the other applications cannot use the same password or other useful information as that.

k. Password Management

Password management is an important topic, we will discuss some important points to password vulnerability, the vulnerability can arise due to unauthorized access to the passwords. The application can't only rely on passwords completely but should include multilevel authentication. Password paging is a technique that forces the users to change their password if something went wrong, this can reduce the risk. The password history should be retained that the user can't change its password to some last passwords. Algorithms should be built to detect some dictionary words that can be detected by the attackers and minimum complexity of passwords must be present at development time.

l. Code Vulnerability

Code vulnerability causes a big threat as we have discussed earlier. In this section we will use some techniques to overcome code vulnerability threats.

m. Validation

Scripting languages are used in Android apps to reduce the time in market band it gives a good turnaround time. Application is also exposed to tweaks of script and sending non valid data to the servers. As a result, the application can be bringing us down or break us to bypassing the validations in front end. To overcome this issue the application must have replication of the servers and front end also. This should reduce the attacks through front end bypassing. SQL injection attacks are removed by server end validation through professional UI design [30,32].

n. Exception Handling

Stack trace is important to the developer through which they can review possible issues occurring in application, but it is important that stack trace should not be visible to end users. Customized messaging with good exception handling creates a better app and avoid security details to revealed.

o. Other source codes

App using source codes is good to mention the deprecated API's. Many platforms usually upload the deprecated API's if deprecated API's are not given then preprocessor approach is considered by the other development platforms where API's are not marked.

V. APPLICATION ANALYSIS

When the application is available or published in the market, it is ready for use by the users and everyone. However, prior to publishing the application it should be analyzed and tested under all circumstances. Internal element of the app should be able to import the necessary information with the tool without leakage of data [34].
In order to determine the possible leakage of data two techniques are used: dynamic and static analysis.

a. Static analysis:

It is a technique that is used to examine the application without executing it. Since many applications use obfuscation the area is wide on which action can be performed, dynamic code loading is also adopted by applications because it restricts access to the internal information. It is good to understand that application association files are encrypted like backup, database and log files. Entropic techniques are quite useful for that.

b. Dynamic analysis:

Dynamic analysis mainly focuses on execution of application. The main theme is to collect the data or values (at runtime) that are used by called instructions. The advantage of using this technique is that it is slowly suspected by obfuscation. Since android application have many behaviors so it is necessary to monitor their activities. Through automatic event injectors and some interfaces.

But there is also another approach known as hybrid analysis, a system that is using this technique should be designed in such a way that if the first one is not working, second one should take place that can overcome the gap. In analysis of smartphones there are no such standards (dynamic or static) to collect the data optimally. Firstly, we collect the data from static analysis and then put that data in dynamic scanning. This is done through hooking techniques; hooking is method that each application is called by signature key [37].

VI. CONCLUSION

From the research we can conclude that Android provides good security and is trying to improve day by day but still, there are some vulnerabilities and security problems that can occur because of improper development and some security flaws. I also have discussed about how to avoid security issues and how to improve them but there should be a strong mechanism developed for Android app security to avoid malicious apps and risk for the security of data.

Because it is a billion dollar industry and still it's growing. Making the money is very easy by using RaaS and that cyber security expert have to go deeper in order to handle the this issue.

Because it is growing very fast, earning by this is other big issue.

REFERENCES

[1] Attacks on WebView in Android System', 27 th Annual Computer Security Applications Conference, 343-352 [11] A.D. Schmidt and S. Albayrak 'Malicious Software for Smartphones', https

[2] T. Luo, H. Hao, W. Du, Y. Wang and H. Yin 'Attacks on WebView in Android System', 27 th Annual Computer Security Applications Conference, 343-352 [11] A.D. Schmidt and S. Albayrak 'Malicious Software for Smartphones', https://www.dailabor.de/fileadmin/files/publications/smartphone_malware.pdf [12]

[3] W. Enck, D. Octeau, P. McDaniel and S. Chaudhri 'A study of Android Application Security', The 20 th USENIX conference on Security, 21-21.

[4] S. Powar, Dr. B. B. Meshram 'Android security framework', International Journal of Engineering Research and Applications.

[5] S. Kaur and M. Kaur 'Implementing Security on Android Application', Journal of Environmental Sciences, Computer Science and Engineering & Technology.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[6] Android. (2017). Android Security Overview. Retrieved from https://source.android.com/security/

[7] Appbrain. (2017). Number of android applications. Retrieved July 28, 2020, from http://www.appbrain.com/stats/number-of-android-apps

[8] Brähler, S. (2010). Analysis of the Android Architecture. Karlsruhe Institute for Technology, 52. Retrieved from http://os.ibds.kit.edu/downloads/sa_2010_braehler-stefan_android-architecture.pdf

[9] Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). Crowdroid: Behavior-Based Malware Detection System for Android. Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices - SPSM '11, 15. https://doi.org/10.1145/2046614.2046619

[10] Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M. S., Conti, M., & Rajarajan, M. (2015). Android security: A survey of issues, malware penetration, and defenses. IEEE Communications Surveys and Tutorials, 17(2), 998–1022. https://doi.org/10.1109/COMST.2014.2386139

[11] Faruki, P., Ganmoor, V., Laxmi, V., Gaur, M. S., & Bharmal, A. (2013). AndroSimilar : Robust Statistical Feature Signature for Android Malware Detection. Proceedings of the

6th International Conference on Security of Information and Networks, (September 2015), 152–159. https://doi.org/10.1145/2523514.2523539

[12] Grace, M., Zhou, Y., Zhang, Q., Zou, S., & Jiang, X. (2012). RiskRanker: Scalable and Accurate Zero-day Android Malware Detection. 10th International Conference on Mobile Systems, Applications, and Services, 281–294. https://doi.org/10.1145/2307636.2307663

[13] Jing, Y., Ahn, G.-J., Zhao, Z., & Hu, H. (2014). RiskMon : Continuous and Automated Risk Assessment of Mobile Applications. Proceedings of the 4th ACM Conference on Data and Application Security and Privacy - CODASPY '14, 99–110. https://doi.org/10.1145/2557547.2557549

[14] Rashidi, B., Fung, C., & Vu, T. (2014). On lightweight mobile phone application certification. Proceedings of the ACM MobiCom Workshop on Security and Privacy in Mobile Environments, 235–245. Original article | doi: 10.25007/ajnu.v6n3a971 4 0 Academic Journal of Nawroz University (AJNU) https://doi.org/10.1145/1653662.1653691

[15] Rashidi, B., Fung, C., & Vu, T. (2014). RecDroid: A resource access permission control portal and recommendation service for smartphone users. 2014 ACM MobiCom Workshop on Security and Privacy in Mobile Environments, SPME 2014, 13–17. https://doi.org/10.1145/2646584.2646586

[16] Russello, G., Jimenez, A. B., Naderi, H., & van der Mark, W. (2013). FireDroid: hardening security in almost-stock Android. Proceedings of the 29th Annual Computer Security Applications Conference, 319–328. https://doi.org/10.1145/2523649.2523678

[17] Xu, R., Saïdi, H., & Anderson, R. (2012). Aurasium: Practical Policy Enforcement for Android Applications. Proceedings of the 21st USENIX Conference, 27. Retrieved from https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final60.pdf%5Cnhttp://dl.acm.org/citation.cfm?id=2362793.2362820

[18] Yan, L. K., & Yin, H. (2012). DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis. In USENIX Security Symposium. https://doi.org/10.1145/2420950.2420980

[19] Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012). Detecting repackaged smartphone applications in third-party android marketplaces. Proceedings of the Second ACM Conference on Data and Application Security and Privacy - CODASKY '12, 317–326. https://doi.org/10.1145/2133601.2133640

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

[21] E. Kovacs, "Wi-fi direct flaw exposes android devices to dos attacks," Online; accessed at July 8, 2020,http://www.securityweek.com/wi-fi-direct-flaw-exposes-android-devices-dos-attacks

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

[23] J. Crussell, R. Stevens, and H. Chen, "MAdFraud: Investigating ad fraud in android applications,"inProc. 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

[24] A. Gember, C. Dragga, and A. Akella, "ECOS: Leveraging software-defined networks to support mobileapplication offloading," inProc. of the 8th

ACM/IEEE symposium on Architectures for networking andcommunications 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

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

[26] D. Barrera, J. Clark, D. McCarney, and P. C. van Oorschot, "Understanding and improving app installationsecurity mechanisms through empirical analysis of android," inProc. of the 2nd ACM workshop onSecurity and privacy in smartphones and mobile devices (SPSM'12), Raleigh, NC, USA.ACM, October2012, pp. 81–92. [Online]. Available: http://doi.acm.org/10.1145/2381934.2381949

[27] A. Pathak, A. Jindal, Y. C. Hu, and S. P. Midkiff, "What is keeping my phone awake?: Characterizingand detecting no-sleep energy bugs in smartphone apps," inProc. of the 10th international conference onMobile systems, applications, and services (MobiSys'12), Low Wood Bay, Lake District, UK.ACM, June2012, pp. 267–280. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307661

[28] L. K. Yan and H. Yin, "Droidscope: Seamlessly reconstructing the os and dalvik semantic views fordynamic android malware analysis," inProc. of the 21st USENIX Conference on Security Symposium(Security'12), Bellevue, WA, USA.USENIX Association, August 2012, pp. 29–29. [Online]. Available:http://dl.acm.org/citation.cfm?id=2362793.2362822

[29] F. Bellard, inProc. of the 2005 USENIX Annual Technical Conference, FREENIX Track, Anaheim, CA,USA. USENIX Association, April 2005, pp. 41–46.[59] S. Server, "Fuzzy clarity: Using fuzzy hashing techniques to identify malicious code," Online; accessed atJuly 8, 2020, http://www.shadowserver.org/wiki/uploads/Information/FuzzyHashing.pdf.

[30] D. French, "Fuzzy hashing techniques in applied malware analysis," Online; accessed at July 8, 2020,http://blog.sei.cmu.edu/post.cfm/fuzzy-hashing-techniques-in-applied-malware-analysis.

[31] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "Whyper:Towards automating riskassessment of mobile applications," inProc. of the 22nd USENIX Conference on Security (SEC'13),Washington, D.C., USA.USENIX Association, August 2013, pp. 527–542. [Online]. Available:http://dl.acm.org/citation.cfm?id=2534766.2534812

[32] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "Pscout:Analyzing the android permissionspecification," inProc. of the 2012 ACM Conference on Computer and Communications Security(CCS'12), Raleigh, North Carolina, USA.ACM, October 2012, pp. 217–228. [Online]. Available:http://doi.acm.org/10.1145/2382196.2382222

[33] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," inProc. ofthe 18th ACM Conference on Computer and Communications Security (CCS'11), Chicago, Illinois, USA.ACM, October 2011, pp. 627–638. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046779

[34] S. Jana and V. Shmatikov, "Memento: Learning secrets from process footprints," inSecurity and Privacy(SP), 2012 IEEE Symposium on (SP'12), San Francisco Bay Area, CA, USA, May 2012, pp. 143–157.

[35] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner, "Analyzing inter-application communicationin android," inProc. of the 9th International Conference on Mobile Systems, Applications, andServices (MobiSys'11), Bethesda, Maryland, USA.ACM, June 2011, pp. 239–252. [Online]. Available:http://doi.acm.org/10.1145/1999995.2000018

[36] G. Paller, "Dedexer," Online; accessed at June 25, 2020, http://dedexer.sourceforge.net/.

[37] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, andP. McDaniel, "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-

aware taint analysisfor android apps,”ACM SIGPLAN Notices, vol. 49, no. 6, pp. 259–269, Jun. 2014. [Online]. Available:http://doi.acm.org/10.1145/2666356.2594299

[38] F. Wei, S. Roy, X. Ou, and Robby, “Amandroid: A precise and general inter-component data flow analysisframework for security vetting of android apps,” inProc. of the 2014 ACM SIGSAC Conference onComputer and Communications Security (CCS'14), Scottsdale, Arizona, USA.ACM, November 2014,pp. 1329–1341. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660357