# Android Security Overview: A Systematic Survey

Xuwei Xia
School of Computer
National University of Defence Technology
Changsha 410072, China
Email: spc cth lf@163.com

Chen Qian
School of Management
Nanjing University of
Posts & Telecommunications
Nanjing 210023, China
Email: lanchendieying@163.com

Bo Liu
School of Computer
National University of Defence Technology
Changsha 410072, China
Email: Kyle.liu@aliyun.com

***Abstract*—Android is now the most popular mobile system and much attention is paid to its security. In this paper, contributions are made in three aspects. First, main work from 2010 to 2016 about Android security is surveyed. Second, a novel aspect proposed in this paper which divide the Android ecosystem into seven spheres. Third, it is found out that system development sphere, application development sphere, application usage sphere(including raw application and normal application usage) and data collection and usage sphere is of great concern. The manufacturing sphere and the administrate sphere lack enough attention. The most used techniques are taint analysis and permission related auditing.**

***Keywords-overview, android, mobile system, security***

## I. Introduction

Following the steps of Google, Android has become an essential part of our daily life, in particular, of our communication method and social media. Since the deployment of the first smart phone powered by Android and named Dream by HTC, Android has gained much attention. Android-powered smart phone has had the biggest market share in the world: 82.8% in the second season of 2015 from IDC. Android brings us conveniences but the threat follows.

In 21st October 2015, a report uncovered the 'WormHole' was submitted to Wooyun(www.wooyun.org, a website exposes vulnerabilities). 'WormHole' was a rootkit threats the security of Android phone and the phone user's privacy. 'WormHole' can get the IMEI of the phone it infects, get the location of the phone, which often is the location of the user, and silently install programmes into Android phone. All these behaviours destroy or partially destroy the confidentiality, completeness and usability of Android phone. That is only a tip of iceberg. According to GDATA, 2333777 malware showed up in 2015, giving a 50% rise compared with the number in 2014.With such pessimistic situation, researchers have done a lot of work to reinforce the security of Android phone. There were so much work having done that the researchers often get lost in mountains of essays. Misunderstandings and unnecessary repetitive readings occupy much of the valuable time. To give a clear view of the Android security, both the security situation and security research work, an overall and systematic view angle is proposed in this paper. A systematic look into the android security is conducted using the view angle, giving a clear view of the security circumstance and researchers' work.

The paper is organised as follows. Section II introduces the background of the android security and the related work. Section III proposes the systematic look view angle. Section IV puts all the technical review to the systematic survey method. Conclusions and future work is conducted in section V, followed by acknowledgements in section VI.

## II. Background

In 24th of November 2014, the well-known listed company, VIA Technologies revealed the information that there was a backdoor in their security chip with model number VT3421 (also with model number TF376). VIA can do a lot with the backdoor, like getting access to the call log, contacts, credit card number and locations on the phone. Furthermore, researchers from University of Michigan revealed a method to attack the chip during the procedure of its manufacturing. They named it 'A2' and proposed it in the recent IEEE Security & Privacy. 'A2' is actually a backdoor and can not be detected with the technology that we have for now.

In July 2015, the security engineer Joshua Drake in Zimperium uncovered a vulnerability in Android and named it Stagefright. It is a 0-day vulnerability that enables the attackers get access to the control of whole system simply with a sms and the user do not have to read the sms. It is estimated that 1 billion android powered devices were under threat.

In September 2014, a white hat in wooyun disclosed the backdoor named CoolReaper. This backdoor is a set of software installed in Coolpad's Android phone when the phone is assembled. That is to say, this backdoor exists from the beginning of the life of a Coolpad's Android phone. The backdoor gives the attacker the system administration authority.

In the above 'WormHole', Baidu planted the source of the hole in its SDK(software development kit), and the developers used the SDK without informed. The hole still exists as the

978-1-4673-9026-2/16/$31.00 ©2016 IEEE

SDK is widely used.

All the above circumstances reveal only part of the mask of the vulnerabilities of android security yet gained a lot of attention. Research has been taken up soon after the release of Dream.

This overview is done with the exquisite and delicate work of the predecessors and with great appreciation, I would mention two of them.

In 2009, William Enck generously shared his understanding of Android security and his thoughts in security enhancement in his glorious paper 'Understanding Android Security'[36]. He detailedly explained the mechanism of Android security and talked about the shortcomes of Android security architecture. It is not only an overview of Android security, but a guidance of the researchers in Android security. In 2014, [8] concluded the work that is done in the permission architecture. This overview contributes in its novel aspect that includes much of the entities in the Android ecosystem. This paper also reveal the research circumstance of Android security.

## III. Technical Methods

In this paper, a systematic method to give a clear view of the current study circumstance is proposed. This method views the android relevant circumstance as a ecosystem. Ecosystem, which is mostly used in the biology, contains two main integrants: entities and correlations between them. The systematic method arranges the view just like that. First, entities are concluded into investors, hardware manufacture, hardware, system developers, system develop environment, Android system and system administrator in manufacture level and application developer, software development kit(SDK), application, service provider, service, normal application, users and user data in consume level. The correlations between the entities are shown in figure 1 named systematic overview of Android ecosystem.

There are 7 main sphere in the system.The investors invests the hardware manufactures. The hardware manufactures produce the hardware and the hardware profits the investor. That is sphere 1, named manufacturing sphere. Sphere 2, the system development sphere, also starts from investors and aims at the system developer. The system developer uses system develop environment to develop the Android system. The system runs in the hardware and finally profits the investor. Sphere 3, the administrate sphere, starts from the investors. The system administrator maintains the function of the Android system after the invest is positioned. The Android system runs in the hardware and profits the investor. Sphere 4, the application development sphere, circles around the application developers. The application developers use SDK to develop the application with the investment from the investor. The applications run in the Android system and finally gain profit for the investors. Sphere 5, the raw application usage sphere, starts from the users. The users are divided into the root users for they have access to the root permission of the Android system in the hardware they use and the normal users for they do not have that access. In sphere 5, the root users use the application and finally profits the investor. Sphere 6, the most ordinary sphere, is named as normal application release and usage sphere. In this sphere, the investor invests the application developers for their development with the SDK. Their production, the application, is then audited by the service provider, which is

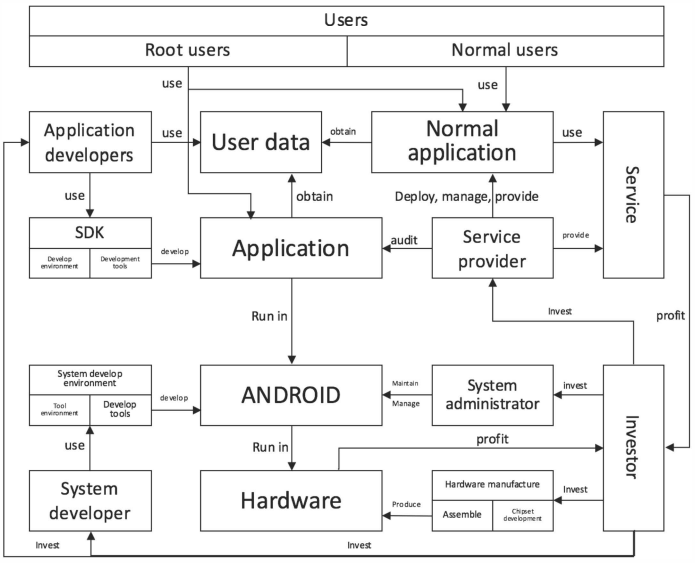


Figure 1.Systematicoverviewofandroidecosystem

also invested by the investor. Users, both root users and normal users, get to the usage of the applications deployed by the service provider. The users use the service that provided by the service provider when they use the applications. This sphere profits the investor at last and is the most ordinary invest-profit method. The last sphere, called the data collection and usage sphere, contains sphere 5 and sphere 6. It is unique as its key point is the data. When the users use the application, the data about the users is created. This data is then obtained by the application developers. The data should be used by the application developers to advance their applications; but just like the two sides of a coin, it is also the original sin.

Figure 1 works as a map to direct the research work. More specifically, every article about android security could find its destination in this map and the over research field and the barren land can be figured out easily from this map. That is the work this paper actually do.

## IV. Systematic Survey

This section reveals the systematic survey according to the technical methods in the previous section.

### A. Sphere 1:Manufacturing Sphere

There was not much study about the manufacturing of the hardware that runs Android. However, in the 37th IEEE Symposium on Security and Privacy in 2016, Kaiyuan Yang and his team revealed the threat in chips with model number OR1200 in his paper 'A2: Analog Malicious Hardware' [1].

This shows that Trojan can be placed in the chip in the manufacturing sphere. Yang's study also showed that such attack is 'small and stealthy'. Android powered hardware can not function properly without chips. Such kind of threat needs more attention.

### B. Sphere 2: System development sphere

In 2010, [2] applied Self Organise Method(SOM) to Android's permission model. This method uses the U-matrix to display the correlations between permissions. It points out that Android's permission mechanism flawed Android in its complicated design. The misunderstanding of the permissions occurs quite often in software developers. This leads to the abuse of some permissions. Barrera also gave his advice that permissions should be improved according to their request frequency. The more often a permission is used, the more fine-granted permissions it should be divided into.

In 2010 as well, [3] released Apex, a policy enforcement framework, to consolidate Android security. Apex is a user-audited-permission-control installer. It leaves the judgement of whether a permission should be granted to a particular software to the users. The usage of a permission is often fuzzy even to a software developer and the power shifting to ordinary user is probably useless.

In 2011, [4] proposed a middleware to reinforce the Android Security. This study uses TaintDroid to label the information in system with private or public. When malware wanted to grab the private information, the middleware would return fake information. This is ingenious but still has its deficiency. It protects the information well but some software do need some information that is labeled with private such as the IMEI of the device to complete its functionality.

In 2012, [5] supplied android security with a symbolic model of the Android security architecture. This work makes it possible for automatic security analysis in Android.

In 2012, [6] released a tool named MalloDroid. This tool aims at the analysis of the man-in-the-middle attack using the SSL protocol. The paper reports that in 13500 applications analysed, 1074(8%) of them are under the threat.

In 2014, [7] provided a set of tools and named them Android Security Modules framework. This framework wants to gain more developers to pay attention to and participate in the security circumstance of Android. This framework is a middle ware and opens lots of APIs that can reveal the status of the system and critical security events in the system to developers. The APIs are so powerful that it is rather dangerous if used by attackers. It is a double-edged sword.

In 2014, [8] investigated the Android security architecture and concluded the drawbacks as coarse granularity of permissions, incompetent permission administration, insufficient permission documentation, over-claim of permissions, permission escalation attack and time of claim to time of use attack(TOCTOU). This conclusion gave a better understanding of the drawbacks of the Android system comparing with [2] because the classification in this paper is more detailing and the countermeasures are of pertinence.

In 2015, [9] revealed two vulnerabilities in the Android System. This paper showed how to bypass the lock screen and the permission check mechanism. It rose concern about the Android security and gave the researchers a better understanding of how the attacks were taken out.

In 2015, [10] uncovered a 0-day vulnerability in Android. This vulnerability, named CVE-2015-3825, is caused by the unchecked deserialization of class member. The publish of the paper reflected the risen concern about Android security.

In 2016, [11] equipped Android with a more fine-granted separation method. It used machine learning method to classify the applications according to their similarity and diversity. This sandbox like separation method differs from the sandbox mechanism with the protection of inter application communications. This finally reinforce the security of Android.

In 2016, [12] proposed an attack method using the inappropriate interrupt handle mechanism. This gives the information that software security guarantee alone or hardware security assurance alone is far from being safe. Safe or not should be judged from the aspect of the entire system, both software and hardware.

### C. Sphere 3: administrate sphere

In 2013, [13] took the vendors into security attention. The researchers doubted that whether the vendors' modification of Android system lead to the fortification of Android system in their products. The result is worrying as they found 85.78% of pre installed applications over claimed the permissions and at least 64.71% vulnerabilities were introduced in by the modification of vendors. The belief 'the newer the system is, the safer the system is' is now unconvincing.

### D. Sphere 4: application development sphere

In 2011, William Enck gave a report with paper 'A Study of Android Application Security'[14]. He pointed out that the developers lack the cognition of APIs about the Internet. Privacy leakage is a huge problem when developers assemble their applications.

In 2011, [15] provided a tool named Stowaway to scan for the over claims of applications. In a survey using the tool, researchers found that almost one third of the applications were over claimed. The developers were struggling to obey the least priviledge rule by failed due to the insufficient API documentation as was revealed in [8].

In 2015, [16] proposed a method to detect the malicious repackaged applications(MRAs). SVDD was used to dynamically compare the difference of the API invoking between MRAs and BOAs(benign original applications). Its functionality lose efficiency when the malicious code only invokes the APIs from local library or do not invoke any APIs.

In 2015, [17] introduced a novel idea that used data mining and machine learning method to handle the distinguish of benign applications and malicious applications. K-means was used as a prototype. This may be a real solution.

In 2015, [18] elaborated VetDroid. VetDroid dynamically analyses the sensitive behaviours of applications according to the permissions they use. This tool gives accurate statistics of the usage of permissions and could lead to a better permission mechanism.

In 2015, [19] carried out both static and dynamic analysis of malicious applications using improved ASEE. The analysis was done in the server so that the method can supply Android powered devices with real-time protect while do not have to concern about the capability of computing.

In 2016, [20] changed the identification from the researcher to the attacker and introduced the technique to run a broadcast receiver without an activity. This work bypassed the security policy that the broadcast receiver should run with an activity and set alarm to the system developers and software developers.

In 2016, [21] compared the malicious libraries in Android and iOS and found that 6.84% Android libraries were malicious. This means software developers can not be more careful and vigilant not to be an accomplice.

In 2016, [22] introduced a method to detect malicious applications based on their behaviours in four aspects: kernel, application, user and packages. It uses both static analysis and dynamic analysis. This method tends to use more dynamic analysis and takes more information into account compared to [16].

### *E. Sphere 5: raw application usage sphere*

In 2015, [23] used pin tool to dynamically protect the functionality of applications. It supervises the execution of the applications and is a implementation of hook.

### *F. Sphere 6: normal application release and usage sphere*

In 2012, [24] records the reputation of applications according to the unique user id in the cloud server. When users download applications from the Internet, the reputation record system will give the the user advice of whether the application is safe and the reputation of the application. This system is a third party application review system. How the users feedback to the system decides the efficiency of the system.

In 2012, [25] released DroidMOSS. This tool uses fuzzy hash method to calculate the similarity of applications to detect the repackaged applications in the third party market. It directly runs with the byte code of the virtual machine in Android and is of compatibility among applications.

In 2012, [26] reported the result of a one-year survey into the 1200 malicious applications. The result said that only 20.2% to 79.6% malicious applications can be detected using the anti virus applications in Android. More powerful anti virus softwares are in great demand.

In 2015, [27] spread the security concern into the IOT(internet of things) area. This paper sorted the security threat in IOT and found that the accessibility is of greatest threat in this area. It contributes to the Android security in broaden the security aspects.

In 2015, [28] surveyed the users altitude to the permissions and adware in Android. This survey reveals the effect of permission mechanism in the public. Survey uncovered that women are seen to be less cautious about their device.

In 2016, [29] used deep learning to detect the malware. It used deep belief networks to map the application features to application categories. It is the advance implementation of machine learning method.

### *G. Sphere 7: data collection and usage sphere*

In 2013, [30] released a tool named AppIntent. AppIntent uses symbolic execution to draw two pictures: the GUI change graph and the invoking graph. Experts then distinguish whether the information delivery in the application is user intended according to the two graph. If the transfer of the information is user intended, it is safe to go even the information is sensitive. That is reasonable and novel and reinforce the taint analysis with reasonable behaviour handle policy. However it is semi supervised as man, in particular expert, is necessary in the system.

In 2014, [31] improved the static taint analysis technology with the ability to analyse the reflect function call, which is often used to bypass the detection of TaintDroid. The improved method is named with FlowDroid.

In 2015, [32] proposed the tool DroidJust. It has a novel idea that the information an application grep is reasonable and safe only if the information is used to change the state of the Android device. It used static taint analysis to follow the information flow in the application.

In 2015, [33] used dynamic detection to monitor the information flow in applications. AndroBlare is used in the work to draw the information flow graph and the graph leads to the conclusion that it is a benign application or a malicious one.

In 2016, [34] used improved taint analysis and restriction of information flow to reinforce the Android system. It is a protection to the data as well.

In 2016, [35] improved the taint analysis by taking events into account. Events are treated like the information so that the analysis is more reasonable and accurate.

## V. Conclusions and Future Work

With the papers listed in Section IV, conclusions can be made that system development sphere, application development sphere, application usage sphere(including raw application and normal application usage) and data collection and usage sphere is of great concern. The manufacturing sphere and the administrate sphere lack enough attention. The most used techniques are taint analysis and permission related auditing. In the following work, evidence that manufacturing sphere and administrate sphere are of necessary importance will be dug out. Probably some research work about the manufacturing and administration will be done.

## ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China under grant No.61572513 and the Education Ministry Doctoral Research Foundation of China under grant No.20134307110016.

## REFERENCE

[1] Kaiyuan Yang, Matthew Hicks, Qing Dong, Todd Austin, and Dennis Sylvester. A2: Analog malicious hardware. In *2016 IEEE Symposium on Security and Privacy*. IEEE, 2016.

[2] David Barrera, H Güneş Kayacik, Paul C van Oorschot, and Anil Somayaji. A methodology for empirical analysis of permission-based security models and its application to android. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 73–84. ACM, 2010.

[3] Mohammad Nauman, Sohail Khan, and Xinwen Zhang. Apex: extending android permission model and enforcement with user-defined runtime constraints. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 328–332. ACM, 2010.

[4] Giovanni Russello, Bruno Crispo, Earlence Fernandes, and Yuri Zhauniarovich. Yaase: Yet another android security extension. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 1033–1040. IEEE, 2011.

[5] Alessandro Armando, Gabriele Costa, and Alessio Merlo. Formal modeling and reasoning about the android security framework. In *International Symposium on Trustworthy Global Computing*, pages 64–81. Springer, 2012.

[6] Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumgärtner, Bernd Freisleben, and Matthew Smith. Why eve and mallory love android: An analysis of android ssl (in) security. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 50–61. ACM, 2012.

[7] Stephan Heuser, Adwait Nadkarni, William Enck, and Ahmad-Reza Sadeghi. Asm: A programmable interface for extending android security. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 1005–1019, 2014.

[8] Zheran Fang, Weili Han, and Yingjiu Li. Permission based android security: Issues and countermeasures. *computers & security*, 43:205–218, 2014.

[9] Michael Heinl. Android security: Creation of a virtual learning environment. *Hochschule Offenburg*, 2015.

[10] Or Peles and Roee Hay. One class to rule them all: 0-day deserialization vulnerabilities in android. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, 2015.

[11] Haoliang Cui, Tianchang Yang, and Shaozhang Niu. Android reinforcement scheme based on the container. *Computer Science and Application*, 6(3):65–71, 2016.

[12] Wenrui Diao, Xiangyu Liu, Zhou Li, and Kehuan Zhang. No pardon for the interruption: New inference attacks on android through interrupt timing analysis. In *2016 IEEE Symposium on Security and Privacy*. IEEE, 2016.

[13] Lei Wu, Michael Grace, Yajin Zhou, Chiachih Wu, and Xuxian Jiang. The impact of vendor customizations on android security. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 623–634. ACM, 2013.

[14] William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri. A study of android application security. In *USENIX security symposium*, volume 2, page 2, 2011.

[15] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 627–638. ACM, 2011.

[16] Wenhao Fan, Yuan'an Liu, and Bihua Tang. An api calls monitoring-based method for effectively detecting malicious repackaged applications. *International Journal of Security and Its Applications*, 9(8):221–230, 2015.

[17] MD Hussain Khan and G Pradeepini. Machine learning based automotive forensic analysis for mobile applications using data mining. *Indonesian Journal of Electrical Engineering and Computer Science*, 16(2):350–354, 2015.

[18] Ms Nigam Paridhi Pravin. Vetdroid: Analysis using permission for vetting undesirable behaviours in android applications. *International Journal of Innovative and Emerging Research in Engineering*, 2(3):131–136, 2015.

[19] Sebastián Londoño, Christian Camilo Urcuqui, Manuel Fuentes Amaya, Johan Gómez, and Andrés Navarro Cadavid. Safecandy: System for security, analysis and validation in android. *Sistemas y Telemática*, 13(35):89–102, 2015.

[20] Milan Oulehla and David Malanik. Techniques that allow hidden activity based malware on android mobile devices. *International Journal of Scientific Engineering and Applied Science*, 2(3):409–419, 2016.

[21] Kai Chen, Xueqiang Wang, Yi Chen, Peng Wang, Yeonjoon Lee, XiaoFeng Wang, Bin Ma, Aohui Wang, Yingjun Zhang, and Wei Zhou. Following devil's footprints: Cross-platform analysis of potentially harmful libraries on android and ios. In *2016 IEEE Symposium on Security and Privacy*. IEEE, 2016.

[22] Andrea Saracino, Daniele Sgandurra, Gianluca Dini, and Fabio Martinelli. Madam: Effective and efficient behavior-based android malware detection and prevention. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1, 2016.

[23] C Sisara Lalji and BV Buddhadev. Mitigation of security attack in android application using pin tool a review. *IJSRST*, 1(4):157–163, 2015.

[24] Welderufael Berhane Tesfay, Todd Booth, and Karl Andersson. Reputation based security model for android applications. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 896–901. IEEE, 2012.

[25] Wu Zhou, Yajin Zhou, Xuxian Jiang, and Peng Ning. Detecting repackaged smartphone applications in third-party android marketplaces. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*, pages 317–326. ACM, 2012.

[26] Yajin Zhou and Xuxian Jiang. Dissecting android malware: Characterization and evolution. In *2012 IEEE Symposium on Security and Privacy*, pages 95–109. IEEE, 2012.

[27] Reijo M Savola, Pekka Savolainen, Antti Evesti, Habtamu Abie, and Markus Sihvonen. Risk-driven security metrics development for an e-health iot application. In *Information Security for South Africa (ISSA), 2015*, pages 1–6. IEEE, 2015.

[28] Gregor Robinson and George RS Weir. Understanding android security. In *International Conference on Global Security, Safety, and Sustainability*, pages 189–199. Springer, 2015.

[29] Zhenlong Yuan, Yongqiang Lu, and Yibo Xue. Droiddetector: android malware characterization and detection using deep learning. *Tsinghua Science and Technology*, 21(1):114–123, 2016.

[30] Zhemin Yang, Min Yang, Yuan Zhang, Guofei Gu, Peng Ning, and X Sean Wang. Appintent: Analyzing sensitive data transmission in android for privacy leakage detection. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1043–1054. ACM, 2013.

[31] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *ACM SIGPLAN Notices*, 49(6):259–269, 2014.

[32] Xin Chen and Sencun Zhu. Droidjust: automated functionality-aware privacy leakage analysis for android applications. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, page 5. ACM, 2015.

[33] Radoniaina Andriatsimandefitra and Valérie Viet Triem Tong. Detection and identification of android malware based on information flow monitoring. In *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*, pages 200–203. IEEE, 2015.

[34] Nariman Tm Ammar. Dynamic privacy management in services based interactions. *Wayne State University Dissertations*, 2016.

[35] Songyang Wu, Yong Zhang, and Xiong Xiong. Efficient privacy leakage discovery for android applications based on static analysis. *International Journal of Hybrid Information Technology*, 9(3):199–210, 2016.

[36] William Enck, Machigar Ongtang, Patrick Drew McDaniel, et al. Understanding android security. *IEEE security & privacy*, 7(1):50–57, 2009.