INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 8, Issue 2 - V8I2-1275)
Available online at: https://www.ijariit.com

Android privacy leakage detection using static analysis

Harsh Jha
harshjha19101997@gmail.com
Sharda University, Greater Noida, Uttar Pradesh

Manish Verma
manish.verma1@sharda.ac.in
Sharda University, Greater Noida, Uttar Pradesh

Prashant Gautam
2018014021.prashant@ug.sharda.ac.in
Sharda University, Greater Noida, Uttar Pradesh

ABSTRACT

The quantity of Android apps have grownexplosively in re- cent years and the wide variety of apps leaking non-public facts have additionally grown. it's far necessary to make sure all of the apps are not leaking non-public information before putting them to the app markets and thereby a privateness leaks detection tool is wanted. We endorse a static taint evaluation approach which leverages the control-flow graph (CFG) of apps to locate privateness leaks among Android apps. We tackle three troubles related to inter- element verbal exchange (ICC), lifecycle of components and callback mechanism making the CFG imprecision. To bridge this gap, we ex- plicitly join the discontinuities of the CFG to provide a unique CFG. based totally on the ideal CFG, we goal at offering a taint evaluation method to locate intra-factor privateness leaks, inter-factor privacy leaks and additionally inter-app privacy leaks.

Keywords: **Static analysis, Taint analysis, Privacy Leaks, ICC, CFG**

1. INTRODUCTION

Android has turn out to be the maximum popular cell telephone running machine over the last 3 years. There are masses of hundreds of packages emerging every day. As of may additionally 2013, 48 billion apps had been installed from the Google Play keep, and as of September 3, 2013, 1 billion Android gadgets were acti- vated1. in the meantime, the Android working gadget also becomes a worthwhile goal for safety and privateness assaults. a major hassle in Android is non- public records leaks. a number of statistics leaks had been reported this years, consisting of sending short messages, making telephone calls and HTTP connections.

We use a static taint analysis approach based totally on control-waft graph (CFG) of analyzed apps to come across privacy leaks in Android. Static taint evaluation technique is a type of facts flow evaluation method which maintains tune of values derived from touchy statistics. We first label the private statistics that we call source (for instance a technique returning GPS coordinate), and then track the data by using statically analyzing the code. If the non-public facts is going toa method which sends it outside the application, alsoreferred to as sink method, we perceive this as a personal statistics leak and we tag the course from the source to the sink as a detected tainted direction. within the CFG, a tainted direction means it is available from the source technique to the sink technique. for this reason, privateness leak detection identifies paths among pre-defined source and sink techniques inside the CFGof analyzed apps.

Since we detect privacy leaks through identifying paths from source methods to sink methods in the generatedCFG of analyzed apps, a precise CFG is essential. However, there are due to the fact we detect privateness leaks via figuring out paths from source methods to sink strategies insidthegenerated CFG of analyzed apps, a specific CFG isessential. but, there are3 problems that make the generated CFG imprecision. The three issues are proven in Fig. 1. the first problem isrelated to inter-aspect communication (ICC) methods in Android, we element it in segment 2.1. the second one hassle is related to Android's lifecycle strategies and the remaining trouble is associated with callback strategies. We element them in section 2.2.

2. BACKGROUND

Android applications run in a digital device known as Dalvik [4]. A big portion of the Android framework and the appli- cations themselves, are first of all coded in Java, then compiled into Java bytecode before being converted into the Dalvik Executable (DEX) format. fortuitously for our analysis, the very last conversion to DEX byte code keeps sufficient informa- tion that the conversion is reversible in most cases the use of the dex2jar tool [13].

Android applications are allotted in compressed applications known as Android applications (APKs). APKs incorporate the entirety that the software needs to run, including the code, icons, XML files specifying the UI, and alertness records. Android programs are available each through the

reputable Android marketplace and different third-celebration markets. those opportunity markets allow customers the liberty to select the supply of their packages.

The respectable Android market is commonly consumer regulated. The ratings of packages within the marketplace are decided by the effective and bad votes of customers. higher ranked programs are proven first in the marketplace and consequently are much more likely to be located. customers can also proportion their experiences with an software through filing a overview. this can alert other customers to avoid the software if it behaves poorly. Google is capable of dispose of any application now not only from the marketplace, however also from customers' phones at once, and has performed so these days while users pronounced malicious packages [12, 16]. however, current research [7] indicates that many famous packages nonetheless leak their users' non-public information.

Android packages are composed of numerous widespread com- ponents that are accountable for specific elements of the ap- plication capability. those additives encompass activities, which manage UI displays; services, which might be returned- floor methods for functionality no longer immediately tied to the UI; Broadcast Receivers, which passively get hold of messages from the Android application framework; and content providers, which offerCRUD operations1 to utility-controlled records. so one can communicate and coordinate between com- ponents, Android presents a message routing system primarily based on URIs. The sent messages are known as Intents. Intents can inform the Android framework to begin a brand new service, to switch to a one-of-a-kind hobby, and to skip facts to any other compo- nent.

Leak Type	Unique Leaks	% of all Leaks	# apps with leak	% apps with leak
Phone	1558	88.9%	2939	16.2%
Location	939	9.96%	434	2.40%
WiFi	59	0.61%	36	0.20%
Record Audio	55	0.57%	32	0.18%

Table 1: Breakdown of Leaks by Type

	Type of Leak					
Ad Library	Phone	Location	Wifi	Microphone	# apps using	% apps using
Mobclix		O	X	X	597	3.3%
Mobclick	G	O	X	X	436	2.4%
adHUB		O	X	X	442	2.4%
Millennial Media	G	O	X	X	162	0.9%

Table 2: Ad Library Leaks by Type.G: found by our analysis, O: missed by analysis but found manually, X: not found by either

Content providers, which offer CRUD operations1 to utility-controlled records. so one can communicate and coordinate between com- ponents, Android presents a message routing system primarily based on URIs. The sent messages are known as Intents. Intents can inform the Android framework to begin a brand new service, to switch to a one-of-a-kind hobby, and to skip facts to any other component. every Android software carries an important XML file called a show up[1]. The show up file informs the Android framework of the utility additives and the way to route Intents between additives. It additionally broadcasts the precise display sizes treated, available hardware and most importantly for this paintings, the application's required permissions

3. ICC Methods

A factor is the simple unit to build Android apps. There are four sorts of additives: a) hobby, representing the user interface; b) carrier, execut- ing duties in history; c) Broadcast Receiver, receiving messages from other components or the device; and d) content provider, performing as the same old interface to share dependent facts among packages. a few particular Android devicestrategies are used to trigger factor conversation. We call them Inter-factor conversation (ICC) methods. Themaximum used ICC approach is startActivity method which starts offevolved a brand new hobby. components use purpose to talk between them. All ICC methods take at least one intent as their parameter. Intents can also encapsulate records and for that reason transfer them among additives.

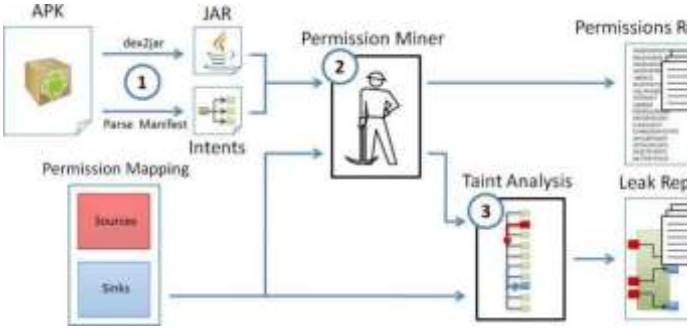
Take listing 1.1 for instance. Activity1 and Activity2 are two com- ponents. One ICC technique exists in Activity1 is startActivity. Activity1 carries one supply technique getDeviceId which returns the particular tool identification (e.g., the IMEI for GSM and the MEID or ESN for CSMA telephones) thinking about that the device identity is touchy information. Activity2 consists of one sink technique ship which sends records outside the software. Neither Activity1 nor Activity2 contains taint route. however In reality, it does exist one statistics leak from source method getDeviceId in Activiy1 to sink method send in Activity2.

Because of the component communication mechanism of Android, we can't locate crossing factor taint paths with the aid of monitoring tainted records due to the fact that there may be no actual code connection among additives however rather best glue code for inter aspect communication. What we need is to attach components collectively so we will construct a particular CFG and thereby allowing us to track tainted records across multiple additives. To acquire this, we need to tackle the following demanding situations.

Getting unique ICC links among additives kind of ICC hyperlinks exist in An- droid: specific ICC linksand implicit ICC links. identifying implicit ICC hyperlinksis more hard than identifying explicit ICC links because they've complicated matching mechanism for two components. The Android device introduces 3 conditions (action, class and information) to perform implicit ICC (additionally IAC). To exactly get all of the implicit ICC hyperlinks, we need to address all of the situations relatedto implicit ICC.

Distinguishing intent statistics. Intents are used to transfer information between components. One motive can containa variety of information but best part of those information may be tainted. We want to distinguish them to keep away from fake effective consequences.

Resolving special ICC strategies. some ICC techniques, that are called unique ICC techniques, have more complicated semantics evaluating with not unusual ICC strategies that only cause one-way verbal exchange between additives (e.g., start Activity method). We want to deal with them especially. With the approach start Activity For Result, a result can be dispatched returned from an hobby whilst it ends. as an example, thing A makes use of start Activity For Result technique to begin issue B and waits for until B ends. while B ends, thing A retrieves effects returned from element B and runs again.



4. LIFECYCLE METHODS

Inside the lifecycle control of the additives in Android, there may be no essential method as in a conventional Java utility. instead, the Android system switches between states of a components lifecycle by calling callback methods including on Start, on Resume or on Create. The lifecycle of an activity is proven in Fig. 2.

As a minimum six lifecycle strategies (e.g., on Pause) are involved in an pastime's lifetime. those techniques arenot without delay connected in the app's code, however rather they're achieved by using the Android machine.

5. CALLBACK METHODS

With the user-centric nature of Android apps, a consumer can interact plenty with the apps (or machine) via the contact screen. The management of consumer inputs is specially done via handling unique callback techniques inclusive of the on Click method that is referred to as whilst the user clicks on a button. for example, method on Click (line 5 in list 1.1) is a callback approach with a purpose to be carried out while its associated button is clicked. but, there aren't any code at once connected to the strategies in the utility.

6. AIMS AND GOALS

Our principal purpose is to locate personal data leaks in Android applications. For this we outline the specifications, layout and put in force a static analysis tool to be able to discover sensitive taint paths between customized assets and sinks in Android packages. The device will not only discover intra- component touchy paths, but also inter-component and inter-software touchy paths. The main predicted contribution of this research to the sector of Engineering secure software program and systems is a taint evaluation tool that is particular, sound, effi- cient and that produces less false tremendous for analysts who work inside the field of Android protection. the primary utilization of the toolis to detect privateness leaks. This device can build name paths between two techniques, one being a source and the opposite a sink. We define resourcesand sinks for our intention to hit upon personal statistics leaks. How- ever, the tool might be used for other reason. as an instance, it is able to be used to come across that a aid is opened however in no way closed through defining the open useful resource technique as the supply and the near resource method because the sink.

7. FUTURE WORK

Android-precise manage and data glide. As defined in section 6, there are numerous specific ways execution may additionally waft in Android that we plan to address inside the future. the use of Intents, one approach canb call every other, either immediately by means of name or indirectly by way of sort of preferred task.

Both cases are more complicated to research than trendy manipulate waft. in the former case, we might want to introspect at the values of the arguments in reason passing and within the latter case we might want to build up a model of each the application's configured environment and doubtlessly the opposite established applications on the smartphone to understand what might be known as.

We goal at presenting a static taint evaluation tool with more precise and sound results comparing with the prevailing tools. To attain this we depend on go with the flow- Droid [2] a surprisingly particular taint evaluation device for Android and Epicc [1] a especially powerful ICC mapping (from ICC method to destination aspect) device in Android.

Modern-day release: To discoverprivacy leaks in Android apps, we have realized a prototype deviceprimarily based on our previous work2: Epicc [1] whichgenerates links between components of Android programsand Flowdroid [2] which performs intra-thing taint analysis. The modern version has essentially solved the three problems distinctive in phase 2. but in a few special case (e.g., distinguishing the data in an motive), still want tobe advanced.

Assessment Plan. We intention at analyzing non-public records leaks in both third-birthday celebration apps or preloaded apps. We intend to check intra-element, inter-element and inter-app based totally privacy leaks. We plan to test our approach towards pattern apps written by way of ourselves due to the fact that we know the anticipated output of the apps in order that we can evaluate the precision of our tool with the other present gear. Then, we look for privacy leaks within the real word programs.

8. SOLUTION

We plan to statically detect privacy leaks with the CFG of analyzed apps. But three problems detailed in Section 2 exist in Android system which make the CFG imprecise. We cannot rely on an imprecise CFG to detect privacy leaks. Thus, our approach first builds a precise CFG by connecting all the isolate CFGs of Android apps. Then, based on the precise CFG, we check whether a source method can reach a sink method or not. If a sink method is reached from a source method, it means that a privacy leak is detected. Since the precise CFG also models the intercomponent communication, we can detect ICC based as well as IAC based privacy leaks . The precise CFG of the example illustrated in Listing 1.1 is shown in Fig. 3. In the CFG, we connect startActivity and onCreate methods to resolve ICC problem. We connect onCreateand onResume methods to resolve Lifecycle Prob- lemand we connect onCreate and onClick methods to resolve Callback prob-lem. Based on the accurate CFG, we can detect an ICC based privacy leak fromgetDeviceId in Activity1 to send in Activity2.

9. RELATED WORK

Android privacy leaks have recently attracted lots of attentions. AppIntent [5] analyzes user-intended sensitive data transmission in Android. Woodpecker [6] studies capability leaks which analyze the reachability of a dangerous permission from a public, unguarded interface. Yajin et al. [7] report passive content leaks which cause affected applications to passively disclose in-application data. Our approach is different from them that we provide a generic taint analysis tool which can detects all the

above leaks with low false alarms. CHEX [8] uses taint analysis technique to detect component hijacking vulnerabilities in Android Applications. However, it does not analyze calls into the Android framework itself.

10. CONCLUSION

We've got described the problems of detecting privacy leaks in Android apps. we've got also defined the solutions of resolving the above problems. Our motivation is to build an particular CFG and thereby to discover ICC primarily based as well as IAC primarily based privateness leaks in Android apps.

As Android profits even more marketplace percentage, its customers want a way to determine if personal statistics is leaked through 0.33-birthday party applications. while iOS contains a overview and approval system, Android is based on consumer regulation and a permissionsversion that limits programs' get entry to to restricted resources. Our primary purpose become to analyze privateness violations in Android programs. alongside the way, we identified a mapping between API techniques and the permissions they require, created a device to find out the permissions an application requires, gathered a database of over 23,000 Android applications and detected over 9000 capability privacy leaks in over three,200 packages.

11. REFERENCES

[1] Damien Octeau et al. "Effective inter-component communication mapping in an- droid with epicc: An essential step towards holistic security analysis". In: Pro-ceedings of the 22nd USENIX Security Symposium. 2013.

[2] Steven Arzt et al. "FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps". In: Proceedings of the 35th Conference on Programming Language Design and Implementation. 2014.

[3] Patrick Lam et al. "The Soot framework for Java program analysis: a retrospec- tive". In: Cetus Users and Compiler Infastructure Workshop (CETUS 2011). 2011.

[4] Alexandre Bartel et al. "Dexpler: Converting Android Dalvik Bytecode to Jimple for Static Analysis with Soot".In: ACM Sigplan International Workshop on the State OfThe Art in Java Program Analysis. Beijing, China, 2012.

[5] Zhemin Yang et al. "Appintent: Analyzing sensitive data transmission in android for privacy leakage detection". In: Proceedings of the 2013 ACM SIGSAC confer- ence on Computer & communications security. ACM. 2013, pp. 1043–1054.

[6] Michael Grace et al. "Systematic detection of capability leaks in stock Android smartphones". In: Proceedings of the 19th Annual Symposium on Network .

[7] Yajin Zhou and Xuxian Jiang. "Detecting passive content leaks and pollution in android applications". In: Proceedings of the 20th Annual Symposium on Network and Distributed System Security. 2013.

[8] Long Lu et al. "Chex: statically vetting android apps for component hijacking vulnerabilities". In: Proceedings of the 2012 ACM conference on Computer and communications security. ACM. 2012, pp. 229–240.

[9] Android developer reference. http://d.android.com/.

[10] Android security and permissions. http://d.android.com/guide/topics/security/security.html.

[11] Go Apk. Go apk. http://market.goapk.com.

[12] Dan Bornstein. Dalvik vm internals, 2008. Accessed March18, 2011. http://goo.gl/knN9n.

[13] IBM T.J. Watson Research Center. T.j. watson libraries for analysis (wala), March 2011.

[14] The Nielsen Company. Who is winning the u.s. smartphone battle? Accessed March 17, 2011, http://blog.nielsen.com/nielsenwire/online mobile/who- is-winning-the-u-s-smartphone-battle.

[15] William Enck, Peter Gilbert, Byung-Gon Chun, Landom P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones.

[16] OSDI, 2010. http://appanalysis.org/tdroid10.pdf.

[17] Adam P. Fuchs, Avik Chaudhuri, and Jeffrey S. Foster. Scandroid: Automated security certification of android applications.

[18] http://www.cs.umd.edu/ avik/papers/scandroidascaa.pdf.

[19] Google. Android market. http://market.android.com.

[20] Apple Inc. App store review guidelines. http://developer.apple.com/appstore/guidelines.html.

[21] Engin Kirda Manuel Egele, Christopher Kruegel and Giovanni Vigna. Pios: Detecting privacy leaks in ios applications. NDSS'11. http://www.iseclab.org/papers/egele- ndss11.pdf.

[22] Peter Pachal. Google removes 21 malware apps from android market. March 2011. Accessed March 18, 2011. http://www.pcmag.com/article2/0,2817,2381252,00.asp.

[23] pxb1988. dex2jar: A tool for converting android's.dex formatto java's .class format. https://code.google.com/p/dex2jar/.

[24] SlideMe. Slideme: Android community and application marketplace. http://slideme.org/.

[25] O. Tripp, M. Pistoia, S. J. Fink, M. Sridharan, and

[26] O. Weisman. Taj: Effective taint analysis of web application. In PLDI '09: Programming language design and implementation, pages 87-97. ACM, 2009.

[27] Sara Yin. 'most sophisticated' android trojan surfaces in china. December 2010. Accessed March 18, 2011. http://www.pcmag.com/article2/0,2817,2374926,00.asp.

[28] Yajin Zhou, Xinwen Zhang, Xuxian Jiang, andVincent W. Freeh. Taming information-stealing smartphone applications (on android). TRUST, 2011. http://www.csc.ncsu.edu/faculty/jiang/pubs/TRUST11.pdf.

