ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

Offensive Protection in Android for RIG

[1] Atul Chandrakant Jadhav, [2] Sunil P. Khachane
[1] [2] Department of Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, India

Abstract: **- Pilfering of sensitive data from apps is at all times measured to be one of the most precarious threats to Android arena. This can happen to the apps without palpable execution and implementation flaws, by molesting some design blemishes of the mobile operating system, e.g., common communication channels a rogue app needs to run side by side with the sincere app (such as phone book, Internet browser, Bluetooth control service, messaging, dialer, IoT interface, etc.) to save its runtime information. To dispose for protection from this new species of attacks, here is a research of a potent & advanced system which does not require any adjustment of principal systems such as on operating system or existing applications. This system will be proactively protecting any app from any category exist today in android arena. This new approach of protection, called Offensive Protection in Android from RIG attacks, spoils a rogue app's runtime monitoring encounter by suspending (ending & pausing) all alarmed background processes when the sincere app is running in the front, and restoring the state of all alarmed background processes from the state where they had paused after the sincere app finishes execution entirely and its runtime environment is sterilized. The trial studies show that this new Offensive Protection is sovereign of OS version and works well with small impacts on the ability of sincere apps and the routine of OS. Most essentially, the notion primary behind this approach, comprises providing security at the level of application, defense at the level of common channel with no conciliation to routine of the device because offense is the best defense.**

Keywords**: Android Security, Mobile Security, Offensive protection in android, RIG Attack, Runtime Information Gathering.**

I. INTRODUCTION

Android systems are enormously widespread because of masses of paid & free applications titled as apps. Apps are motive for huge spreading of android based mobile systems and nowadays become mandatory part of it with express development. Every now & then android market is engulfed with newer apps from diverse categories with resolve to provide facilities & services in fields such as Medical, Finance, Media, Education, Entertainment, Security, Banking, etc. All of these apps desire to access, process and convey delicate information such as financial, personal, business related activities (e.g., investment secret, bank details, disease and medical history, social media credentials, etc.) that needs to be protected from illicit or rogue packages installed and residing on same device and running at concurrent time. Android OS maintains security machinery for preventive apps from accessing each other's data by providing sandboxing atmosphere for apps execution and by providing unique process Id's. This defense though is not adequate to protect runtime information gathering through common communication channels (e.g., audio, Bluetooth) or open resources (e.g., memory, CPU usage). Delicate data could still be unprotected to the rogue app that constantly monitors the decent app's events and collects its runtime information from those public resources. Runtime information gathering posture severe threat to latest version of android systems and even to secrecy of its users[1].

Runtime information gathering is an activity that comprise of malevolently saving the data processed, produced, transferred or received by another app that does not mutually approves to share its delicate or any other form of data during its execution, in an effort to directly steal or indirectly conclude delicate user information. Such an attack can chance by molesting the permission the rogue app sanctioned by the user, e.g., a non-messaging app reading all inward and outward messages without user's accord, recording phone dialogue by a non-dialer app [2], [3], which was approved RECORD_AUDIO permission at the time of install and extracting subtle data such as transaction PIN or CVV no. [4]. Also, a Bluetooth activated medicinal device's decent app can bounce out delicate & vital information [5]. A big apprehension here is that even zero permission grants can still gain greatly delicate & vital data from a variety of public or common channels, signifying the imperative vulnerability of mobile systems in extrication of an app's processes from its data. Examples can comprise Internet browsers give out web content and other delicate information detected by simply reading the memory foot prints; phone's accelerometer coordinate values pilfered from public channel can reveal key strokes logged [6]. Case in fact comprises recording audio of phone talks from the genuine phone app, gathering medical history data to conclude the ailment condition of user, etc. This runtime information gathering (RIG) threat is convincing and severe, as validated by earlier study and new discoveries, these rogue apps can read entire message threads, can control Bluetooth data transmission, can read, alter and even spoof contact in the device (which can lead to

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

a Social Engineering attack through which device user can fall prey to the name displayed but with attackers contact no. underneath, and can misinterpret or overlook it as a trusted contact), can steal passwords and other credentials no matter if they are encrypted or plain format, can control IoT enabled devices which are connected by means of shared communication channel to handset, can disclose user location, can collect phone call recordings and decipher pin from voice sample, etc.

In Android operating system applications are divided in front and background occurrences where any app can be stopped and resume as per need of operating system. Offensive Protection in Android from RIG takes advantage of this treatment given to applications by android systems. Offensive Protection in Android from RIG will also be sensibly designed to select the true moments to start and end the protection process, and efficiently defense itself against rogue apps. The experimental studies show that this new Offensive Protection in Android from RIG system is independent of OS version and works well with small influences on the efficacy of legitimate apps and also the performance of the OS is affected marginally less.

II. EXISTING SYSTEM

Existing solutions for defying Runtime Information Gathering attack needs a revision of either the Android Operating System or the unprotected applications which are at risk. Refining the access control mechanism in android system is also possible to dodge the risk of Runtime information gathering but all of these approaches unenviably disturb the system's efficiency and usability and at the same stage is time as well as resource consuming. Android provides security to each application by sandboxing its space which delights each app exclusively and thus providing process identification and file system access control. Each app is treated as a user thus assigned a user ID (UID), in order to isolate them from each other. This keeps public and shared resources out of the scope of this security mechanism causing easy resource sharing such as Bluetooth, Internet connection, audio, camera captures, etc. among multiple apps and users. Each app at the time of its installation must be granted a permission to access these shared resources. There are different protection levels [7] assigned to each permission, such as some permission are automatically granted to the apps when prompted, some risky permissions need user's consent, and critical system permissions for system apps. Apps can access these shared resources only with a proper permission granted.

There are some serious flaws in this security mechanism. Such as once a permission is granted then there is no control of user or OS over how and when that permission grant is utilized, For example, a non-messaging app with MESSAGE_READ permission can access all messages as per its will and wish, an app with CONTACT_READ and INTERNET_CONNECT permission can transmit sensitive contacts from phonebook, an app with AUDIO_RECORD permission can catch all phone conversations and all of this takes place without discrete knowledge of the user. Added to this no protection provided to runtime information flow between apps which can lead to a RIG attack. Rogue apps running in the background can cause severe damage to confidentiality of user and integrity of the system. When any such specific RIG attack happens and identified then Google & sometimes manufacturer of mobile handset come up with a security patch which again does not possesses a huge scope of success against the rest of the other types of RIG attacks. Scope of application developers or android programmers revisiting to develop patch for vulnerable app is too much time consuming and technically impossible because no app can override the permission grants given to any other app.

III. PROBLEM DEFINITION

Android is ingenuous to endure the RIG menace. The operational restrictions, such as public channels and shared resources, expose it to abundant practices of runtime information gathering, which resulting into the disclosure of intense user information. This vulnerability is genuine, persistent and severe, thus only new methodologies or techniques can possibly tackle and provide actual solution with suitable installation over millions of android devices across globe.

Thus there is immense need to develop a security system capable to offensively defend known RIG attacks and provide secured environment for execution of sincere app and defend user sensitive data from leaking through public & shared communication channels on the application level, without touching the OS or the sincere apps under protection, this system will be called Offensive Protection in Android from RIG, that can be accessed from Google Play store and installed on any Android device to acquire immediate protection of user's sincere apps by taking advantage of side channel information and detection of rogue apps with suspending these rogue apps and avoid permission exploitation granted at the time of install.

IV. CONTRIBUTION

In the aggressive world of information security, one must believe offence Is the best defense and applying the philosophy "to catch a thief think like a thief" so in this scenario of Runtime information gathering where shared channels are exploited by misusing the permissions granted, proposing here an offensive defense system called Offensive Protection in Android from RIG where this system will detect rogue apps by reading their shared

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

channel data and other logs representing their behavior which is then stored and used for detection of these rogue apps. That means trying Runtime information gathering (RIG Attack) on rogue apps to see if they are involved in Runtime information gathering.

Our system works in two different modes as workflow described and shown in Fig.1 & Fig.2 below. Our system can be started from any mode thus providing detection and protection against message and phonebook related exploitation by third party apps or rogue apps.

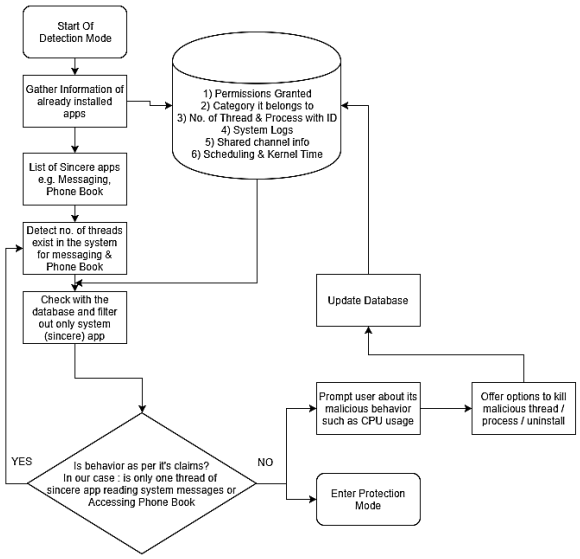

Fig 1: Detection Mode

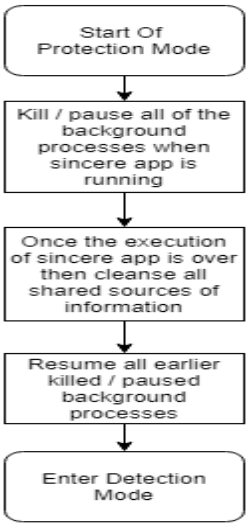

Fig 2: Protection Mode

A. First Attack Vector

First attack vector is on messaging service where situation is worst and even neglected by users & developers because READ_SMS permission is granted for more than 70% (i.e. more than 2 million) of apps on Google play store. Whereas 95% out of these apps need not to read more than 1% of messages out of message book during its entire lifecycle of existence (i.e. between install till uninstall of that app) and that too for reading verification code sent over message for 2-way authentication. But with this READ_SMS permission all of these apps can read complete message book from user's device, where every user have one system message management app who reads 100% of messages all the time. Thus absolutely no need of granting READ_SMS permissions for any other apps.

Table I : Sampling results for messaging service

App	Kill/Pause%	oom_score_adj	Effective
Facebook	87	9	Yes
Google play	72	6	Yes
Imo	85	8	Yes
Olacabs	91	9	Yes
Quickr	93	9	Yes
Truecaller	88	8	Yes
Twitter	95	8	Yes
Uber	94	9	Yes
Kotak bank	69	5	Yes
Amazon	89	9	Yes

Performance measurement is done against applications which are not system's default messaging apps but still acquired READ_SMS permission at the time of install and tries to expose user information received or sent in the messaging service. The performance of our system is examined by sampling 100 to 200 attempts for each of the application listed in Table I and results of killing or pausing non system messaging app is given below in Fig. 3. Apps with more than 8 score were found to be killed or paused for more than 85% on android device running more than 10 different applications simultaneously.

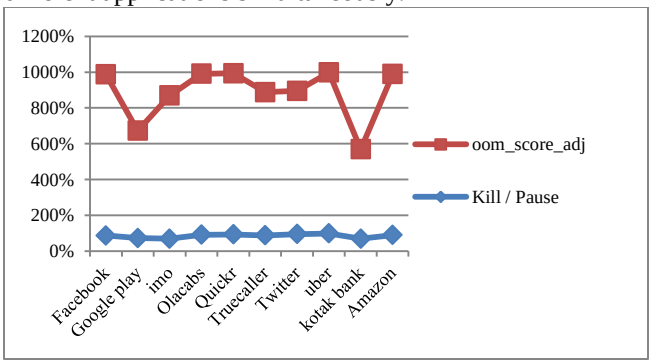

Fig 3: Performance Measure for messaging service

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

B. Second Attack Vector

Second attack vector is also tried and tested. In this attack vector another area in mobile systems which deals with the contact details information called as Phone Book. The contact details in the mobile device are very sensitive and should be considered utmost private property of the mobile user. Contact information is second most favorite thing in RIG attacks that are besieged by rogue apps around the globe where rogue app attempts to access, update, transfer and wipe out partial or entire contact list without mobile user's consent in the runtime. Android operating system provides a permission grant control for protection of phone book information such as READ_CONTACT & WRITE_CONTACT but once these permissions are granted then application is allowed to modify and play with phone book information at its will. Because RW_CONTACT permission is granted for more than 82% (i.e. more than 2.4 million) of apps on Google play store. Whereas 98% out of these apps need not to read more than 2% of contacts out of phone book during its entire lifecycle of existence (i.e. between install till uninstall of that app) and that too for informing user about any contact in phone book is using this app for social networking or automatic service utilization. But with this RW_CONTACT permission all of these apps can read complete phone book from user's device, where every user have one system (default) phone book management app who need to read 100% of contacts all the time. Thus absolutely no need of granting RW_CONTACT permissions for any other apps.

Table II: Sampling results for Phonebook service

App	Kill/Pause%	oom_score_adj	Effective
Facebook	85	8	Yes
Google play	90	9	Yes
Imo	68	6	Yes
LinkedIn	83	7	Yes
CiscoWebex	87	7	Yes
Truecaller	94	9	Yes
Twitter	75	7	Yes
Uber	81	8	Yes
Kotak bank	62	4	Yes
Amazon	94	9	Yes

Performance measurement is done against applications which are not system's default phone book management apps but still acquired RW_CONTACT permission at the time of install and tries to expose or misuse user information stored in phone book service. The performance of our system is examined by sampling 100 to 200 attempts for each of the application listed in Table II and results of killing or pausing non default phone book management app is given below in Fig.4. Apps with more than 7 score were found to be killed or paused for more than 80% on android device running more than 10 different applications simultaneously.

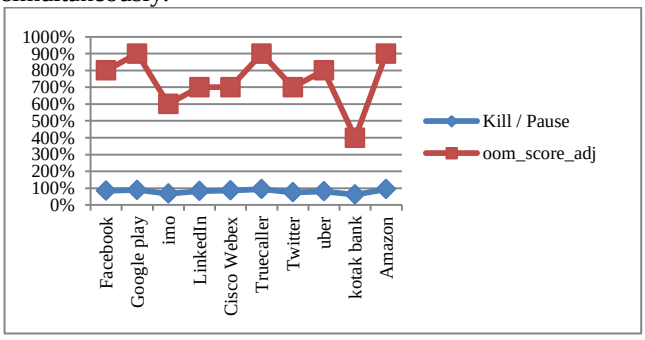

Fig 4: Performance Measure for Phonebook service

V. EXPERIMENTAL RESULTS

Fig.5 shows first attack vector trying to exploit message read permission granted to it and reads messages all the time.

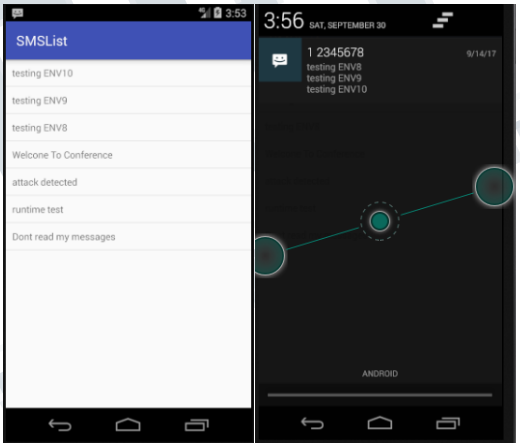

Fig 5: Testing attack vector for messaging service

Fig.6 shows first attack vector being detected by our offensive protection system and prompted to user discretion for its removal

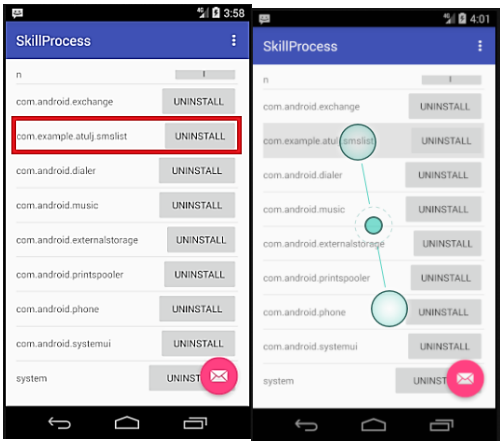

Fig 6: Detection of attack vector

VI. CONCLUSION

In Offensive Protection in Android from RIG a function is introduced which exploit the runtime information, system logs, and other related behavioral information (such as no. of threads present, CPU usage when running in background, source of installation, category it belongs, etc.) of rogue apps before killing or pausing them and avoid sensitive user information from rogue access. Concentrating on two vectors type of RIG attack that is illegal message read & exploitable Phone book access, it is examined that Our system is found efficient against different category of apps who are not message management apps or default system message apps in case of first attack vector and default phone book management apps in case of second attack vector but still posing threat for stealing information in message system & contact information in phone book of device. This approach can be further extended to study and mitigate other types of RIG attacks such as contact spoofing, phone call recordings, Shared channel information stealing, etc.

REFERENCES

1. Nan Zhang, Kan Yuan, Muhammad Naveed, Xiaoyong Zhou and XiaoFeng Wang, "Leave Me Alone: App-level Protection Against Runtime Information Gathering on Android" in IEEE Symposium on Security and Privacy, 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7163068/

2. X. Zhou, S. Demetriou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and K. Nahrstedt, "Identity, location, disease and more: Inferring your secrets from android public resources," in Proceedings of 20th ACM Conference on Computer and Communications Security (CCS), Nov. 2013. [Online]. Available: http://www.cs.indiana.edu/~zhou/files/fp045-zhou.pdf

3. S. Jana and V. Shmatikov, "Memento: Learning secrets from process footprints," in Proceedings of the 2012 IEEE Symposium on Security and Privacy, ser. SP '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 143–157. [Online]. Available: http://dx.doi.org/10.1109/SP.2012.19

4. R. Schlegel, K. Zhang, X. yong Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A stealthy and context-aware sound trojan for smartphones." in NDSS. The Internet Society, 2011.[Online].Available: http://dblp.uni-trier.de/db/conf/ndss/ndss2011.html#SchlegelZZIKW11

5. M. Naveed, X. Zhou, S. Demetriou, X. Wang, and C. A. Gunter, "Inside job: Understanding and mitigating the threat of external device misbonding on android," 2014.

6. L. Cai and H. Chen, "Touchlogger: inferring keystrokes on touch screen from smartphone motion," in Proceedings of the 6th USENIX conference on Hot topics in security, ser. HotSec'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 9–9. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028040.2028049

7. "Android permission,"http://developer.android.com /guide /topics / manifest /permission- element.html/, 2014.