Android App Collusion

Ervin Oro
ervin.oro@aalto.fi

Tutor: Jorden Whitefield

set proper margins and document format

Abstract

abstract

KEYWORDS: *list, of, key, words*

1 Introduction

Android is an operating system (OS) that is primarily designed for mobile devices. With more than two billion active devices [1], it is estimated to be the most widely used OS, surpassing even Windows [2, 3]. Android is designed to be an open platform: developed and maintained by Google LLC, but largely released as the Android Open Source Project for everyone to study and build upon [4]. It also includes support for apps, which are easily installable application packages that may extend the functionality of devices. Apps can be developed and distributed by anyone with very low barrier of entry.

While this popularity of Android is not reflected by the proportion of malware attacks, most of which still target Windows, both the number and complexity of attacks against Android are increasing [5]. This is especially

troublesome as many people increasingly rely on their phones – often to keep their personal data, online account credentials, money, and more. McAfee estimates that revenues for mobile malware authors could be in the billion-dollar range by 2020 [6].

Given the increasing potential damage from Android malware, defending against it is an active area of research. Android uses a multi-layer security approach [1]: Google regularly removes potentially harmful applications from its Play Store, and has developed Play Protect[1] to also scan applications from other sources. Additionally, Android's platform security has been enhanced over the years with features like SELinux protections[2], exploit mitigations[3], privilege reductions[4], encryption, and Verified Boot. Recent versions of Android make use of hardware security features and receive regular updates. These measures have been partially successful, as exploit pricing and difficulty are growing by some estimates [1].

However, malicious actors are looking for ways to bypass existing protections, and a number of threats, e.g., app collusion, cannot yet be reliably detected nor defended against. App collusion is a secret collaboration between apps with malicious intent (Section 2). This can be facilitated by any of the numerous ways for apps to communicate with each other that the Android system provides (Section 3). Methods for apps to collude also exist on the iOS platform [7]. Given a malicious app that would be detected and blocked with state of the art security systems, its functionality could be easily split into several apps, so that each of them would be categorized as benign when analysed separately [8].

Android app collusion is not a new concept [9], and multiple attempts have been made to develop a suitable detection system.

brief overview of existing approaches based on Section 5

However, there still does not exists any robust and usable ways to detect app collusions. Most proposed solutions have a large number of false positives due to an inability to differentiate collusion from legitimate collaboration. Furthermore, since the number of possible combinations is exponential in the number of apps, that is, N^N, most proposed solutions apply very aggressive filtering, causing only some malicious combinations to be included into analysis, and others to be reported as false negatives.

[1] https://www.android.com/play-protect/

[2] https://source.android.com/security/selinux/

[3] https://lwn.net/Articles/695991/

[4] https://android-developers.googleblog.com/2017/07/seccomp-filter-in-android-o.html

Finally, approaches attempting to overcome both of these issues have been computationally infeasible thus far. Therefore, app collusion remains an open research challenge.

This report aims to provide an overview of app collusion on the Android platform as follows. Section 2 discusses the nature of app collusions in general, Section 3 provides specific overview of methods that can be used for colluding on Android, Section 4 describes known examples of colluding apps, and Section 5 gives a more in-depth systematic overview of approaches that have been taken to collusion detection and their limitations.

2 Description and definition of app collusion

The Oxford English Dictionary defines collusion as "Secret agreement or understanding for purposes of trickery or fraud; underhand scheming or working with another; deceit, fraud, trickery" [10]. Asăvoae et al. [11] define collusion for the case of Android apps as the situation where several apps are working together in performing a threat. According to these definitions, app collusion must have following three properties:

1. Colluding apps must be working together secretly. Conversely, apps working together in collaboration is common and encouraged practice when such collaboration is well documented [!] .

Citation needed

2. All colluding apps must be in agreement. A distinctly different but related concept is the "confused deputy" attack, where one app mistakenly exposes itself to other installed apps [!] .

Hardy, N.: The confused deputy:(or why capabilities might have been invented. ACM-SIGOPS Operating Systems Review 22(4), 36–38 (1988)

3. Colluding apps must have malicious intent. The intent of Android app collusion would then be to violate one of Android's security goals, which are defined in [12] as:

   (a) to protect app data, user data, and system resources (including the network),
   (b) to provide app isolation from the system, other apps, and the user.

   It is important to note that the goal b is not to enforce isolation, but merely to provide isolation for those who want it. As such, apps working

jorden

I also wonder if it is worth mentioning OWASP Mobile top 10 security... https://www.owasp.org/index.php/

together do not automatically violate goal b, but it would be collusion if apps worked together to break isolation with some other non-content app, the system, or the user.

many things affect, including non technical

difficult to distinguish

alternative definition [13]

3 Methods for colluding

By default, all android apps run in separate sandboxes, and by default, all communication between sandboxes is blocked [14]. However, there are exceptions to both of these statements, as described below.

3.1 Overt channels

Android Open Source Project describes officially allowed channels for inter-app communication in [14] and [15].

This entire subsection is based on data from these two sources. Is this kind of citation sufficient?

Apps published by the same entity may share a sandbox using the shared UID feature. In this case, there are no restrictions for their communication. They may use any of the traditional UNIX-type mechanisms, including filesystem, local sockets, or signals.

When apps are running in different sandboxes, the Linux kernel prevents them from accessing each others processes or files. In older Android versions, apps had the ability to make their files world-accessible, opening up a channel that way, but this is forbidden on newer versions of Android. Apps can still access files on external storage, but this requires a user-granted permission, giving users some visibility into the fact that such communication channel may be used by the app.

However, Android also provides a method for apps to communicate without any user-granted permission or visibility. This is enabled by a remote procedure call mechanism called binder. Any app can send any message to binder at any time, but other apps must explicitly start listening and accept incoming communications. Apps have three main ways to do that:

1. Services. Apps may start services, which can provide interfaces that are directly accessible using binder.

2. Intent filters. Intents are simple message objects that represents an "intention" to do something. Apps may ask some part of them to be executed when an intent with certain properties matching their filter is initiated.

3. ContentProviders. Apps can define ContentProviders to expose some of their data.

Binder provides an easy way for apps to communicate with each other, promoting openness and allowing separation of concerns. Examples include apps using an intent to ask the camera app to take a photo instead of asking camera control permission, and communication apps allowing other apps to share data through itself. Binder has a well-defined interface that could be theoretically monitored.

3.2 Covert channels

Mention Access Control policies, SEAndroid

4 Examples of Android app collusion

Completely rewrite; include a diagram to illustrate

~~A very widely cited example of collusion is an imaginary situation as follows.~~ Blah et al. [x] define an example of an Android app collusion as follows:

jorden: Make the points below a numbered list perhaps. Easier to follow. With a list of numbered steps I would then also think about if a nice diagram could be made, and numbered to correspond to each step described?

One app , APP_A, has access sensitive information, but no access to internet. Another app, APP_B, ~~on the other hand,~~ has access to internet, but no access to any sensitive information. Many authors [!] argue that in this case, one app could pass information to the other one, which could in turn then exfiltrate the information. Some authors [!] have extended this concept to also cover cases where data is passed to multiple apps before being finally exfiltrated. All current research focuses on detecting such situations.

Citation needed

Citation needed

jorden: The steps are not clear as it is mixed in with discussion. Need to spend some time separating these.

There is one known case of Android app collusion in the wild [16]. Interestingly enough, even though this example is also widely referred to [!], it does not follow the pattern described above.

list some references

jorden: This last paragraph is very informal and chatty. This is ok for the draft but would need to be rewritten.

short description of MoPlus SDK

5 Existing methods for detecting collusions

Bibliography

[1] Android Open Source Project *et al.*, "Android security 2017 year in review," Mar. 2018. [Online]. Available: https://source.android.com/security/reports/Google_Android_Security_2017_Report_Final.pdf

[2] Awio Web Services LLC, "Browser & platform market share," Dec. 2018. [Online]. Available: https://www.w3counter.com/globalstats.php?year=2018&month=12

[3] StatCounter, "Operating system market share worldwide," Dec. 2018. [Online]. Available: http://gs.statcounter.com/os-market-share

[4] Android Open Source Project, "Legal notice," 2019. [Online]. Available: https://developer.android.com/legal

[5] AV-TEST GmbH, "Security report 2017/18," Jul. 2018. [Online]. Available: https://www.av-test.org/en/news/the-av-test-security-report-20172018-the-latest-analysis-of-the-it-threat-scenario/

[6] McAfee, "Mobile threat report," Apr. 2018.

[7] L. Deshotels *et al.*, "SandScout," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*. ACM Press, 2016. [Online]. Available: https://dl.acm.org/citation.cfm?id=2978336

[8] H. Chen *et al.*, "Malware collusion attack against machine learning based methods: issues and countermeasures," in *Cloud Computing and Security*, X. Sun, Z. Pan, and E. Bertino, Eds. Cham: Springer International Publishing, 2018, pp. 465–477. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-00018-9_41

[9] R. Schlegel *et al.*, "Soundcomber: A stealthy and context-aware sound trojan for smartphones," in *Proceedings of the Network and Distributed System Security Symposium, NDSS'2011*, Jan. 2011. [Online]. Available: https://www.researchgate.net/publication/221655538_Soundcomber_A_Stealthy_and_Context-Aware_Sound_Trojan_for_Smartphones

[10] "collusion, n." in *OED Online*. Oxford University Press, Dec. 2018. [Online]. Available: http://www.oed.com/view/Entry/36460

[11] I. M. Asăvoae *et al.*, "Detecting malicious collusion between mobile software applications: the Android™ case," in *Data Analytics and Decision Support for Cybersecurity*. Springer International Publishing, Aug. 2017, pp. 55–97.

[12] Android Open Source Project, "Security," 2019. [Online]. Available: https://source.android.com/security

[13] M. Xu *et al.*, "AppHolmes," in *Proceedings of the 26th International Conference on World Wide Web - WWW '17*. ACM Press, 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7349085/

[14] Android Open Source Project, "Application security," 2019. [Online]. Available: https://source.android.com/security/overview/app-security

[15] ——, "Application sandbox," 2019. [Online]. Available: https://source.android.com/security/app-sandbox

[16] J. Blasco *et al.*, "Wild Android collusions," in *VB2016*, Oct. 2016. [Online]. Available: https://www.virusbulletin.com/conference/vb2016/abstracts/wild-android-collusions

[17] F. I. Abro *et al.*, "Android application collusion demystified," in *Future Network Systems and Security*, R. Doss, S. Piramuthu, and W. Zhou, Eds. Cham: Springer International Publishing, 2017, pp. 176–187. [Online]. Available: http://openaccess.city.ac.uk/18503/

[18] I. M. Asavoae *et al.*, "Towards automated Android app collusion detection," *arXiv preprint arXiv:1603.02308*, 2016. [Online]. Available: https://arxiv.org/abs/1603.02308

[19] I. M. Asăvoae, H. N. Nguyen, and M. Roggenbach, "Software model checking for mobile security – collusion detection in $\mathbb{K}$," in *Model Checking Software*, M. d. M. Gallardo and P. Merino, Eds. Cham: Springer International Publishing, 2018, pp. 3–25. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-94111-0_1

[20] H. Chen *et al.*, "Malware collusion attack against SVM: issues and countermeasures," *Applied Sciences*, vol. 8, no. 10, 2018. [Online]. Available: http://www.mdpi.com/2076-3417/8/10/1718

[21] D. Davidson *et al.*, "Enhancing Android security through app splitting," in *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer International Publishing, 2018, pp. 24–44.

[22] K. Elish *et al.*, "Identifying mobile inter-app communication risks," *IEEE Transactions on Mobile Computing*, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8587187

[23] Google, Inc., "The Google Android Security Team's classifications for potentially harmful applications," Feb. 2017. [Online]. Available: https://source.android.com/security/reports/Google_Android_Security_PHA_classifications.pdf

[24] McAfee, "Safeguarding against colluding mobile apps," May 2016. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/solution-briefs/sb-quarterly-threats-may-2016-1.pdf

[25] I. Muttik, "Partners in crime: investigating mobile app collusion," in *McAfee Labs threats report*. McAfee, Jun. 2016, pp. 8–15. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-may-2016.pdf

[26] Android Open Source Project, "The Android source code," 2019. [Online]. Available: https://source.android.com/setup

[27] L. Qiu, Y. Wang, and J. Rubin, "Analyzing the analyzers: FlowDroid/IccTA, AmanDroid, and DroidSafe," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis - ISSTA 2018*. ACM Press, 2018.

[28] R. Spreitzer, G. Palfinger, and S. Mangard, "SCAnDroid: automated side-channel analysis of Android APIs," in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks - WiSec '18*. ACM Press, 2018.

[29] F. Wei *et al.*, "Amandroid: A precise and general inter-component data flow analysis framework for security vetting of Android apps," *ACM Transactions on Privacy and Security*, vol. 21, no. 3, pp. 1–32, apr 2018.

[30] J. Zhan *et al.*, "Splitting third-party libraries' privileges from Android apps," in *Information Security and Privacy*. Springer International Publishing, May 2017, pp. 80–94.

format bibliography

remove nocite

Todo list

set proper margins and document format 1
abstract 1
list, of, key, words 1
brief overview of existing approaches based on Section 5 2
Citation needed 3
Hardy, N.: The confused deputy:(or why capabilities might have been invented. ACMSIGOPS Operating Systems Review 22(4), 36–38 (1988) 3
I also wonder if it is worth mentioning OWASP Mobile top 10 security... https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10 . 3
many things affect, including non technical 4
difficult to distinguish 4
alternative definition [13] 4
This entire subsection is based on data from these two sources. Is this kind of citation sufficient? 4
Mention Access Control policies, SEAndroid 5
Completely rewrite; include a diagram to illustrate 5
Make the points below a numbered list perhaps. Easier to follow. With a list of numbered steps I would then also think about if a nice diagram could be made, and numbered to correspond to each step described? 5
Citation needed 5
Citation needed 5
The steps are not clear as it is mixed in with discussion. Need to spend some time separating these. 5
list some references 6
This last paragraph is very informal and chatty. This is ok for the draft but would need to be rewritten. 6

short description of MoPlus SDK 6
format bibliography 9
remove nocite 9
remove list of todos 11
remove list of todos