Android App Collusion

Ervin Oro

`ervin.oro@aalto.fi`

Tutor: Jorden Whitefield

Abstract

abstract

1 Introduction

Android is estimated to be the most widely used operating system overall [1, 2], running on more than 2 billion active devices [3]. While this is not reflected by malware, most of which still targets Windows, both the number and complexity of attacks against Android are increasing [4]. McAfee estimates that revenues for mobile malware authors could be in the billion-dollar range by 2020 [5].

Taking that into account, defending Android devices against malware is an active area of research. Thanks to recent developments, exploit pricing and difficulty are growing [3], indicating that many common attacks can now be detected and protected against. However, especially with increasing incentives, malicious actors are looking for ways to bypass existing protections, and some threats, like app collusion, still can not be reliably detected nor defended against.

App collusion is secret collaboration between apps with malicious intent [6, 7]. Android system provides many documented unrestricted channels for apps to communicate with each other [8]. Additionally, apps can use wide array of covert channels to achieve undetected communication [!] . A **Citation needed:** malicious application that would be detected and blocked with state of the art security systems could be easily split into several applications, so that each of them would be categorized as benign when analysed separately [9].

Android app collusion is not a new concept [10], and multiple attempts have been made to develop a suitable detection system. **brief overview of existing approaches**

However, there still does not exists any robust and usable ways to detect app collusions. Most proposed solutions have large number of false positives due to inability to differentiate collusion from legitimate collaboration. Furthermore, the only known example of app collusion in the wild [11] would be out of scope for most current works, as the exponential explosion of having to analyse all combinations of billions [!] of apps has **Citation needed: number of apps in play store** forced authors to focus on a very narrow subset of threats. Approaches attempting to overcome both of these issues have been infeasible thus far.

This report aims to provide a more general overview of the topic. 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 and section 4 briefly describes known examples of colluding apps. Section 5 gives a more in-depth systematic overview of approaches that have been taken to collusion detection, as well as discusses their shortcomings.

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." [6]. In this context, [7] defines app collusion as the situation where several apps are working together in performing a threat. From this, three aspects of collusion can be derived:

1. Collusion is secret. Conversely, apps working together in collaboration is common and encouraged practice when such collaboration is well documented [!] . **Citation needed:**

2. Collusion is when all colluding parties are in agreement. A distinctly different but related concept is the "confused deputy" attack, where one app mistakenly exposes itself to other installed apps [**!**] .

3. Collusion is with malicious intent. In the Android context, it would be the intention to violate one of its security goals, which are to protect application data, user data, and system resources (including the network) [12]. The Android Open Source Project also lists providing "application isolation from the system, other applications, and from the user" as a separate goal, but in this context, collusion would be when apps work together to break isolation with some third component, as for two willing parties there exist several legitimate communication channels [8].

many things affect, including non technical

difficult to distinguish

3   Methods for colluding

Overt and covert channels

4   Examples of Android app collusion

A very widely cited example of collusion is an imaginary situation as follows. One app has access sensitive information, but no access to internet. Another app, 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:

There is one known case of Android app collusion in the wild [11]. Interestingly enough, even though this example is also widely referred to [**!**] , it does not follow the pattern described above.

Citation needed: list some references

short description of MoPlus SDK

5   Existing methods for detecting collusions

Bibliography

[1] Awio Web Services LLC, "Browser & platform market share," Dec. 2018.

[2] StatCounter, "Operating system market share worldwide," Dec. 2018.

[3] Android Open Source Project *et al.*, "Android security 2017 year in review," Mar. 2018.

[4] AV-TEST GmbH, "Security report 2017/18," Jul. 2018.

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

[6] "collusion, n." in *OED Online*. Oxford University Press, Dec. 2018.

[7] 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.

[8] Android Open Source Project, "Application security," 2019.

[9] 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.

[10] 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.

[11] J. Blasco *et al.*, "Wild Android collusions," in *VB2016*, Oct. 2016.

[12] Android Open Source Project, "Security," 2019.