Research: Securing Android Applications from Screen Capture (FLAG_SECURE)

Published: 2016-04-13 · Archived: 2026-04-05 19:10:26 UTC

Summary—TL, DR

Apps on Android and some platform services are able to capture other apps's screens by using MediaProjection API. Because of the way this API implements "securing" sensitive screens, there exist some possible security issues. The best way to secure your Android app is to use FLAG_SECURE on sensitive screens and DO NOT use the virtual keyboard (here is why).

MediaProjection API

Since Android 5.0, there exists **a new MediaProjection API** that allows apps to record videos and take screenshots of screens belonging to other apps. The API is described as follows:

> Android 5.0 lets you add screen capturing and screen sharing capabilities to your app with the new android.media.projection APIs. This functionality is useful, for example, if you want to enable screen sharing in a video conferencing app. The new createVirtualDisplay() method allows your app to capture the contents of the main screen (the default display) into a Surface object, which your app can then send across the network. The API only allows capturing non-secure screen content, and not system audio. To begin screen capturing, your app must first request the user's permission by launching a screen capture dialog using an Intent obtained through the createScreenCaptureIntent() method.

(On Android versions prior to 5, there are other methods such as undocumented APIs, and ADB, we are focusing on Android 5+)

This API also drives several other functions in the OS:

- Recent apps screenshots
- Pinning
- Casting to other displays
- Google Play Games, video recording feature
- Taking screenshots

All of these functions as well as the **MediaProjection API** can take screenshots and videos of other apps. For apps to use the API, special permission is required, for platform features, no special permission is needed. Additionally, any applications signed by the system key (Google apps) can use this API without permission as well.

A good open source example of an application that uses the API can be found here:

https://github.com/JakeWharton/Telecine

Secure and non-secure content

As mentioned in the Google docs above, “the API only allows capturing non-secure screen content”. What exactly is “secure” and “non-secure” content?

This refers to a special flag which can be applied to views in Android, called **FLAG_SECURE**. It is described in Android docs as follows:

> Treat the content of the window as secure, preventing it from appearing in screenshots or from being viewed on non-secure displays

Setting this flag on Android view will prevent screenshots from being taken manually, and any other app or platform service will show a black screen. This functionality is not global for the entire app, but can be set on specific screens which can be more sensitive, and not set on others. There is no other way or permission that can mark an entire app or any part of it from being excepted from screen capture or recording.

NOTE: Even on views marked with FLAG_SECURE, the virtual keyboard is ALWAYS visible. This is due to a known Android bug which Google has so far refused to fix:

https://code.google.com/p/android/issues/detail?id=129285

How screen capture really works in Android

The term “secure” as used in this context does not mean that the content of the app cannot be captured, rather that it cannot be “viewed on non-secure displays”. This is because screen capture and the concept of secure / non-secure isn’t what developers may think it is.

Behind the scenes, this API and related platform services use the concept of Casting (similar to AirPlay). Apps that capture screenshots and record videos, must create a virtual display to which then the device content is cast to. The FLAG_SECURE flag is also not used for security but rather means copyrighted content in context of DRM and displays—i.e. secure content would be something like a DVD, and a secure display would be an HDTV.

This is clear on the device itself—when an app begins to record the screen, the cast icon is turned on in the notification bar. This is also clear from the Android source code and this doc:

> Display flag: Indicates that the display has a secure video output and supports compositing secure surfaces. If this flag is set then the display device has a secure video output and is capable of showing secure surfaces. It may also be capable of showing protected buffers. If this flag is not set then the display device may not have a secure video output; the user may see a blank region on the screen instead of the contents of secure surfaces or protected buffers.

That would means that an Android device casting to a DRM-protected display like a TV would always display sensitive screens, since the concept of secure really means “copyrighted”. For apps, Google forestalled this issue by preventing apps not signed by the system key from creating virtual “secure” displays, but not for physical

devices. There is also an existing Android bug asking for the concept of DRM and screen security to be separated into different flags:

https://code.google.com/p/android/issues/detail?id=93026

Security issues with the current API

First of all, a basic foundation of mobile app security is a clear separation between apps. One Android app is should never able to read the preferences, data or capture cloud notifications of another app. This paradigm breaks down in case of screen capture/recording. **An app gaining access to the MediaProjection API or any of the platform services using it, is able to capture screen output from other apps including PIN numbers, passwords, credit card numbers, etc.**

Second, by using a flag used for marking copyrighted content, it would make it easier to subvert the system. Some ways this can be subverted include:

- Gaining permissions to create a virtual display marked as secure would show all secure content. Right now this is preventable by using the system key, but a rooted phone or some other way that fools the system into creating such display would by pass this protection.
- Also, casting to a physical secure display, or perhaps a wireless one, would also display content

Third, even with the FLAG_SECURE in use, some parts of the screen can still be captured. The virtual keyboard is one existing example, but there may be others (perhaps notifications?).

[ADDED 06/24/2016: Mark Murphy from CommonsWare points out several other issues with FLAG_SECURE child objects – see his blog post here]

Fourth, there is no clear indication to the user they are being recorded other than the cast icon. Clicking on the icon shows no devices since virtual devices aren't going to be listed. A better warning may be needed.

A better solution as suggested in this bug report, would be to define a separate flag and never allow any app or system service to see its output under any reason, and also to blank out the entire screen even if other apps or service display anything on it. An even better solution would be to make this opt-in for apps, instead of opt-out.

Attack vectors

There are several possible avenues of attack which would result in an app being installed on a user's phone recording their app activity. These include:

- Malicious apps in the app store that masquerade as legit casting apps requiring record permission—since users don't know that casting apps can also record their screen
- Remote install via compromised desktop as described in this paper
- Overlaying permission screens as described here

To record even non-secure screens, the following can be tried but they are not practically feasible:

- On rooted phones, creating a virtual display marked as "secure"

- Fooling the system into thinking that a given app is a system app, allowing it to create secure displays

All of these would result in an app sitting on the phone and recording user activity. However, other than the last two methods, FLAG_SECURE views would not be recordable, although the virtual keyboard is. The only indicator to the user would be the cast icon, but when they click on it, no devices would be listed.

[ADDED 2020-03-23: As per this excellent blog post from Yanick Fratantonio, FLAG_SECURE will NOT protect against attacks using accessibility services (a11y)]

Attacks in the Wild

Some examples of these type of attacks happening in the wild:

- July 2018 – Panoptispy Study – see here, here, here and here
- July 2018 – Anubis Android malware – see here and here

Conclusion—Protect Your Apps

To protect your apps from being recorded by other apps, FLAG_SECURE should be used on any views containing sensitive data. Additionally, since the virtual keyboard is also vulnerable, it is recommended that these screens either use a custom keyboard, or if feasible to use an on-screen custom layout (for numeric input like PIN numbers).

We have surveyed many of the top apps in the Google Play store, and many of them including some Google-owned apps do not use FLAG_SECURE or if they do, do not secure the keyboard.

We also hope that Google would lock down this API and solve the issues highlighted in this article.

Credits

Researched and written by Yakov Shafranovich.

References

Google CID: 3–5606000008769
