From Two Permissions to Complete Control of the UI Feedback Loop

By Yanick Fratantonio, Chenxiong Qian, Simon Pak Chung, Wenke Lee

Archived: 2026-04-05 23:41:08 UTC

Cloak & Dagger

Cloak & Dagger is a new class of potential attacks affecting Android devices. These attacks allow a malicious app to completely control the UI feedback loop and take over the device — without giving the user a chance to notice the malicious activity. These attacks only require two permissions that, in case the app is installed from the Play Store, the user does *not* need to explicitly grant and for which she is not even notified. Our user study indicates that these attacks are practical. These attacks affect all recent versions of Android (including the latest version, Android 7.1.2), and they are yet to be fixed.

TL;DR — Main Takeaways

- We uncover a series of vulnerabilities and design shortcomings affecting the Android UI.
- These attacks abuse one or both of the SYSTEM_ALERT_WINDOW ("draw on top") and BIND_ACCESSIBILITY_SERVICE ("a11y").
- If the malicious app is installed from the Play Store, the user is not notified about the permissions and she does not need to explicitly grant them for the attacks to succeed. In fact, in this scenario, "draw on top" is automatically granted, and this permission is enough to lure the user into unknowingly enable a11y (through clickjacking).
- The possible attacks include advanced clickjacking, unconstrained keystroke recording, stealthy phishing, the silent installation of a God-mode app (with all permissions enabled), and silent phone unlocking + arbitrary actions (while keeping the screen off). See the full list below.
- These attacks are practical: we performed a user study (with 20 human subjects), and no user understood what happened.
- Most of these attacks are due to design issues, and they are thus challenging to prevent. In fact, one may say that some of these functionality work "as intended"; Nonetheless, this work shows that this functionality can be abused.
- To date, all these attacks are still practical (see "Which versions of Android are affected" and "Responsible Disclosure" below).

List of Attacks

Attacks that abuse the “draw on top” permission:

- *Context-aware clickjacking* & *Context hiding*: two techniques that make luring the user to enable the accessibility service practical, even when the latest security mechanisms (e.g., "obscured flag") are correctly implemented and enabled. (Note: others have identified ways to use clickjacking to get a11y. See "FAQ" below.)
- *Invisible Grid Attack*, allowing unconstrained keystroke recording, including password, private messages, etc.

Attacks that abuse “accessibility service” permission:

- *Unconstrained keystroke recording*, including passwords. According to the documentation, this should not be possible (See "security note" here)
- Security PIN stealing
- Device unlock through PIN injection + perform arbitrary actions *while keeping the screen off*!
- Stealing two-factor authentication tokens (SMS-based, Google Authenticator, and other app-based tokens)
- Ad hijacking
- Web exploration

Attacks that abuse both permissions:

- Silent installation of God-mode app (with all permissions enabled)
- Stealthy phishing (for which the user finds herself logged in, as she would expect)

Which versions of Android are affected?

Here is the current status (as of June 19th, 2017). Previous versions are very likely to be vulnerable as well.

Attacks	**Android 5.1.1 (32.0%*)**	**Android 6.0.1 (31.2%)**	**Android 7.1.2 (7.1%)**
Invisible Grid Attack	vulnerable	vulnerable	vulnerable
Clickjacking → a11y	vulnerable	vulnerable	vulnerable
Silent God-Mode	vulnerable	vulnerable	vulnerable**
Stealthy Phishing	vulnerable	vulnerable	vulnerable
PIN stealing	vulnerable	vulnerable	vulnerable
Phone Unlocking (while screen off)	vulnerable	vulnerable	vulnerable
Leaky a11y (passwords, 2FA tokens, CCs)	vulnerable	vulnerable	vulnerable***

*Relative numbers of devices running a given version of Android. The numbers are taken from Google's dashboard, and they are clustered by Android "main versions", e.g., "Android 5.X".

** Google implemented a partial fix (only on Android 7.1.2): "on top" overlays do not appear anymore whenever an app's permission list is shown. However, this is only used for "normal" permissions, and not for "special" permissions, such as "draw on top" and a11y. This is problematic: since the "clickjacking → a11y" is still possible, a malicious app can use the "Phone Unlocking (while keeping the screen off) attack" to enable these permissions while keeping the screen off, thus making the silent installation of a God-mode app still practical. We suggest Google to extend their protection mechanism to the entire Settings app (or, at the very least, to "special" permissions as well).

*** Although Google marked our bug report as "Won't fix", we noticed that the Google keyboard actually received an update that, at first glance, seems to avoid leaking passwords. In fact, when typing passwords, the accessibility events generated by the Keyboard app itself now contains "Dot" instead of the actual character. However, we found a workaround for our attack: each accessibility event has access to the "hashcode" of the node generating the event. Since it's possible to enumerate the widgets and their hashcodes (which are designed to be pseudo-unique), the hashcodes are enough to determine which keyboard's button was actually clicked by the user.

Demos

Invisible Grid Attack

Context-aware/hiding Clickjacking + Silent God-mode Install Attack

Stealthy Phishing Attack

The Team

- **Yanick Fratantonio** (@reyammer), University of California, Santa Barbara (soon at EURECOM!)
- **Chenxiong Qian**, Georgia Institute of Technology
- **Simon Pak Ho Chung**, Georgia Institute of Technology
- **Wenke Lee**, Georgia Institute of Technology

You can reach us by sending an email to team@cloak-and-dagger.org

Publications

Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop
Yanick Fratantonio, Chenxiong Qian, Simon P. Chung, Wenke Lee.

- In the Proceedings of the IEEE Symposium on Security and Privacy (S&P), San Jose, CA, May 2017. [PDF] [Slides] [Talk] Distinguished Practical Paper Award!
- Black Hat USA, Las Vegas, NV, July 2017. [Black Hat White Paper] [Slides] [Talk]

  Please use the following bibtex entry to cite our work:

```
@InProceedings{fratantonio17:cloakdagger,
  author = {Fratantonio, Yanick and Qian, Chenxiong and Chung, Simon and Lee, Wenke},
  title = {{Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Lo
  booktitle = {Proceedings of the IEEE Symposium on Security and Privacy (Oakland)},
  month = {May},
  year = {2017},
  address = {San Jose, CA}
}
```

Responsible Disclosure

We responsibly disclosed our findings to Google's Android security team. A timeline of the disclosure steps and responses from Google are posted here (we will keep this updated as time passes):

- August 22nd, 2016 — We opened several issues on the bug tracker for the Android Open Source Project (AOSP).
- August 31st, 2016 — Android security team sets severity as "Moderate" for one of the bugs ("draw on top" → unconstrained keystroke recording).
- September 30th, 2016 — Android security team marks one of the reported bugs ("a11y" → unconstrained keystroke recording + leaking security PIN + unlocking device while keeping the screen off) as "work as intended".
- October 10th, 2016 — We follow up pointing out that the Accessibility Service's documentation states that a11y should not be able to access passwords (see "security note"), and that a11y should not be able to unlock the device and perform arbitrary actions while keeping the screen off.
- October 13th, 2016 — Android security team states that "The password will be repeated if the user explicitly turns that on in settings under Settings → Accessibility → Speak passwords. The option is off by default. If the user explicitly enables this feature, it is not a security vulnerability."
- October 14th, 2016 — We follow up clarifying that our attack works even without this feature (In fact, our report does not even mention that.)
- October 18th, 2016 — Android security team marks this bug as "High severity".
- November 28th, 2016 — Android security team downgrades this bug as "not a security issue" and marks it as "Won't Fix (Intended Behavior)" because "limiting those services would render the device unusable".
- December 19th, 2016 — A draft of our IEEE S&P'17 paper is shared with Adrian Ludwig, director of Android Security.
- March 15th, 2017 — We follow up, pointing out that the documentation states otherwise. We also follow up on all the other bugs we opened, asking for a status update.
- May 3rd, 2017 — We follow up a second time asking for a status update for the bugs we reported.
- May 4th, 2017 — Android security team keeps our a11y findings as "won't fix", but they state they will update the documentation. We did not receive updates about the other bugs we reported.
- May 8th, 2017 — We have a telco with the anti-malware and a11y Google teams during which we thoroughly discussed all the details of our research.
- May 19th, 2017 — The a11y team confirms the a11y-related issues we reported as "won't fix".
- May 22nd, 2017 — This website and our research paper at IEEE S&P are made public.
- *Current* — All the attacks discussed by this work are still practical, even with latest version of Android (Android 7.1.2, with security patches of June 5th installed).

Frequently Asked Questions

How can an app stealthily obtain the two permissions?

> A malicious app can declare the "draw on top" and the a11y permissions through its manifest file. *If the app is hosted and installed through the Google Play Store, the attacks will not require the user to explicitly notify the user about the two permissions.* In fact, in this scenario, the "draw on top" permission is

automatically granted, and this work shows how it is possible to get access to a11y (through clickjacking) even when the latest security mechanisms are enabled and correctly implemented.

Why is the "draw on top" permission automatically granted?

This behavior appears to be a deliberate decision by Google, and not an oversight. To the best of our understanding, Google's rationale behind this decision is that an explicit security prompt would interfere too much with the user experience, especially because it is requested by apps used by hundreds of millions of users. For example, Facebook requires this permission to implement "Android chat heads", one of its very popular features.

Are these permissions shown to the user after the app is installed?

They are not. For Android 5 (or older versions), the list of permissions requested by the app is shown at installation time: however, the "draw on top" and the a11y are *not* included in this list (because they are treated as "special"). For Android 6 (and later), the list of permissions is not shown to the user at installation time (and, as mentioned above, the "draw on top" is automatically granted).

How difficult is it to get an app with these two permissions approved to the Google Play Store?

A quick experiment shows that it is trivial to get such an app accepted on the Google Play Store. In particular, we submitted an app requiring these two permissions and containing a non-obfuscated functionality to download and execute arbitrary code (attempting to simulate a clearly-malicious behavior): this app got approved after just a few hours (and it is still available on the Google Play Store).

What do you recommend to users?

We recommend users to check which applications have access to the "draw on top" and the a11y permissions. Unfortunately, both permissions are considered "special" and, for this reason, certain versions of Android may show "no permission required" even if, in fact, the app has access to both the permissions required for our attack. Here we provide instructions for several versions of Android (if you have recommendations regarding instructions for others Android versions, please let us know and we will post them here):

- Android 7.1.2:
  - — "draw on top" permission: Settings → Apps → "Gear symbol" (top-right) → Special access → Draw over other apps.
  - — a11y: Settings → Accessibility → Services: check which apps require a11y.
- Android 6.0.1:
  - — "draw on top" permission: Settings → Apps → "Gear symbol" (top-right) → Draw over other apps.
  - — a11y: Settings → Accessibility → Services: check which apps require a11y.
- Android 5.1.1:
  - — "draw on top" permission: Settings → Apps → click on individual app and look for "draw over other apps"
  - — a11y: Settings → Accessibility → Services: check which apps require a11y.

This work shows that the user should not consider her device's UI as a trusted source of information. Thus, from a conceptual point of view, the user should rely on other means than the device's UI itself. An alternative solution is to use command line tools (such as `adb`) or to determine the permissions requested by each app through the Play Store website. For example, to check the permissions of the official LastPass app (which requires both permissions), you can go to

its Play Store page, scroll down, and click "View details" under "Permissions". The "draw on top" permission will appear under the "Others" / "draw over other apps" label, while the a11y will appear under "Others" / "bind to an accessibility service".

Why are these bugs not fixed yet?

Some of the issues uncovered by this work are design-related issues — not simple bugs — and it thus necessarily takes more time to fix them. Moreover, these are not "classic" low-level issues, but UI-related problems. These issues may be more challenging to fix since changes may introduce backward compatibility issues and changes that are "visible" to end users. Finally, some of the bugs were marked as "won't fix", and they will thus not be fixed.

Are these attacks practical?

We believe so. We performed a user study with 20 human subjects, and none detected to be under attack. We report more details in our paper.

These permissions have been abused by existing works. What is the difference?

Previous work has shown that, for example, ransomware abuses the "draw on top" permission to create inescapable fullscreen overlays and block the device. In this work, instead, we show how to abuse these permissions to mount *stealthy* attacks, for which the user does not even suspect she has been infected.

Are you the first ones to use clickjacking to enable a11y?

Nope! Actually, while we independently had the idea of mixing the two permissions, we discovered the possibility to use clickjacking to obtain a11y thanks to a blog post from other researchers. The article also discusses 1) how Google attempted to address the problem by relying on the "obscured flag" mechanism, 2) how a problem in the defense's implementation makes the attack still possible, and 3) that Google considered this new new attack to be not practical and marked this as "won't fix". This blog post motivated us to work in this direction: among the other attacks we uncovered, we show how the "obscured flag" defense mechanism is vulnerable *even if it were implemented correctly*, how it can be abused as a powerful side-channel (see "Invisible Grid Attack"), and how these attacks are extremely stealthy and practical (as showed by our user study). We hope our work will motivate Google to patch these vulnerabilities.

These are "just" UI issues. Why should I care?

We hope that this work shows to both app and system developers how UI issues can be as powerful as classic low-level bugs (e.g., buffer overflows), and that should be considered as first-class bugs.

What do you think Google should do?

Other than suggesting Google to fix the reported bugs and to implement the defense mechanism proposed in our paper, we have a number of short-term recommendations:

- Do *not* automatically grant the "draw on top" permission.
- Perform extensive vetting on apps requiring both permissions.
- The "draw on top" and a11y should be listed under the apps' "normal" permissions (currently, the user needs to navigate to a special, dedicated menu).
- Disable all overlays when the user is interacting with the Settings app.

What is Google's official reply to the press coverage?

According to several news articles, a Google spokesperson said: "We've been in close touch with the researchers and, as always, we appreciate their efforts to help keep our users safer. We have updated

> Google Play Protect — our security services on all Android devices with Google Play — to detect and prevent the installation of these apps. Prior to this report, we had already built new security protections into Android O that will further strengthen our protection from these issues moving forward."

Will you make the proof-of-concept publicly available?

> Since it is still possible to launch these attacks against real users, we will not publicly disclose the proof-of-concept (for now). However, upon request, we will be happy to share the PoC with selected security researchers.

Do you have a question that is not covered here? Send us an email!

Press Coverage

This work received significant press coverage. Here there are some articles: NewsWeek, The Register, ThreatPost, TechCrunch, The Hacker News, On The Wire, Kaspersky Blog, Daily Mail, International Business Times, Android Central, Android Police, EndGadget, Mashable, ComputerBild (in German), Android Authority, SoftPedia, HelpNetSecurity, XDA Developers, The Hill, SlashGear, Express UK, Malaysia Sun, Indipendent UK, Security Now, Silicon UK, Graham Cluley, The Merkle, Inc., 9to5google, NowSecure, Gigazine (in Japanese), Science Daily, Security Affairs, CanalTech (in Portuguese), CyberScoop, Info Security, Appelmo (in Italian), HeadLines.nl (in Dutch), Android Planet (in Dutch), DobreProgramy (in Polish), WccfTech, BGR, AppInformers, Gadgets 360, Android Headlines, TelePolis (in Polish), Redes Zone (in Spanish), Chip Online TR (in Turkish), MojAndroid (in Slovak), N+1 (in Spanish), Popular Mechanics, Android World (in Italian), Unwire (in Chinese), iDevice (in Romanian), Digital (in Bulgarian), NhipSongs (in Vietnamese), Cepkolik (in Turkish), ITMedia (in Japanese), TekCrispy (in Spanish), 3DNews (in Russian), Libero.it (in Italian), QDSS (in Italian), Xakep (in Russian), Unboxholics, Neowin, Blasting News, KCRA, HotForSecurity, SpiceWorks, Latest HackingNews, DeccanChronicle, MobileSyrup, HackRead, RapidMobile, Noah Conference, Digital Trends, AndroNews (in German), Conservative Angle, Tech Hook, GSM Portal, Live 24 News, TiotsGoa, Atman, DailyUpdate, Kalen2utech, MyBroadBand, PC Tablet, Sohu, Digi163, ITWire, BGR India, TechNews Inc., Thaivisa, Ausdroid, Koco, The Christian Post, TechDroider, PPPFocus, Security Intelligence, GuidingTech, OpenSourceforu, SensorTech, EMGIGroup, TcInet, FileHippo, Heise (in German), KCCI, NextInpact, BTCoin Info, Blorge, Yahoo! Tech, Bit Media, HappyTech, Generation NT, GBHackers, TechGround News, AntiCorruption, Notebook Check, Security Online, TecnoExplora, The Viral News, KakNut, HackersPortugal, Spinonews, LongRoom, TechGossip, TechStutn, Craw, Uroft, Bongo Gadgets, UK Star, Best of Viral, Breaking Tech (in Italian), Android-IT, Irish Sun, TeknoFormat (in Turkish), WinFuture (in German), F-Hack (in Italian), WLWT News, RadioRegional (in Portuguese), TVBS (in Chinese), NewReport (in Greek), CERT Italia (in Italian), Mobil Mania (in Czech), Android Community, Ifnar (in Chinese), Sina (in Chinese), Viet Times (in Vietnamese).
