10 November 2022

Accidental $70k Google Pixel Lock Screen Bypass

I found a vulnerability affecting seemingly all Google Pixel phones where if you gave me any locked Pixel device, I could give it back to you unlocked. The bug just got fixed in the November 5, 2022 security update.

The issue allowed an attacker with physical access to bypass the lock screen protections (fingerprint, PIN, etc.) and gain complete access to the user's device. The vulnerability is tracked as CVE-2022-20465 and it *might* affect other Android vendors as well. You can find my patch advisory and the raw bug report I have sent to Google at feed.bugs.xdavidhu.me.

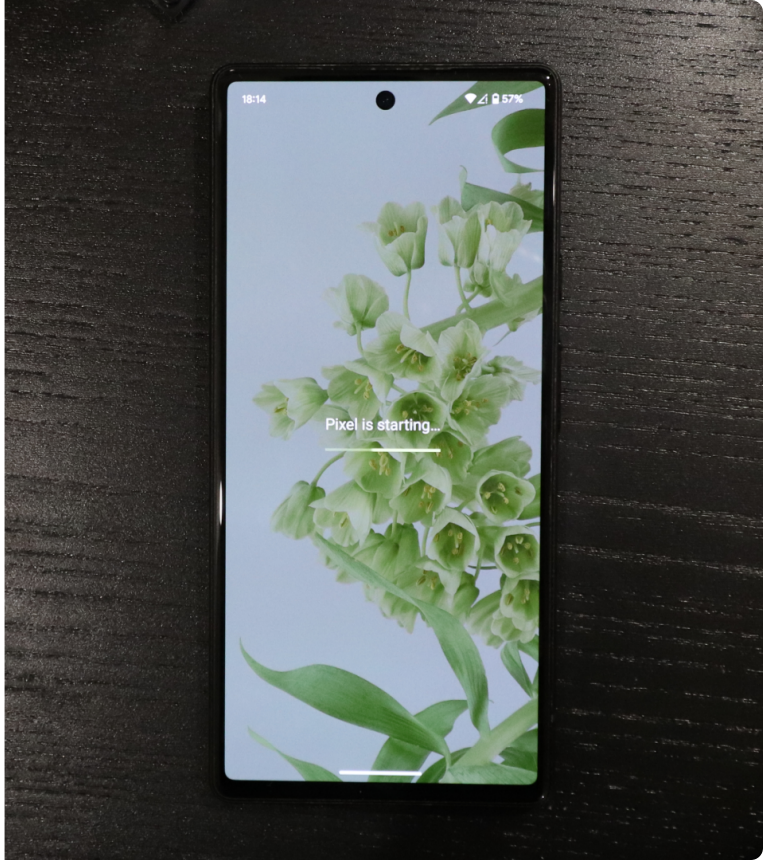
CHAPTER 1:
FORGETTING MY SIM PIN

I'm really glad that this bug is getting fixed now. This was the most impactful vulnerability that I have found yet, and it crossed a line for me where I really started to worry about the fix timeline and even just about keeping it as a "secret" myself. I might be overreacting, but I mean not so long ago the FBI was fighting with Apple for almost the same thing.

I found this bug after 24 hours of travelling. Arriving home, my Pixel 6 was on 1% battery. I was in the middle of sending a series of text messages when it died. I think it was some sort of joke that I couldn't properly finish, so it felt pretty awkward. I rushed to the charger and booted the phone back up.

The Pixel started up and asked for the SIM's PIN code. I usually knew it, but this time I couldn't remember it correctly. I was hoping I might figure it out so I tried a few combinations, but I ended up entering 3 incorrect PINs, and the SIM card locked itself. It now needed the PUK code to unlock and work again.

After jumping into my closet and somehow finding the SIM's original packaging, I scratched off the back and got the PUK code. I entered the PUK code on the Pixel and it asked me to set a new PIN. I did it, and after successfully finishing this process, I ended up on the lock screen. But something was off:



It was a fresh boot, and instead of the usual lock icon, the fingerprint icon was showing. It accepted my finger, which should not happen, since after a reboot, you must enter the lock screen PIN or password at least once to decrypt the device.

After accepting my finger, it got stuck on a weird “Pixel is starting…” message, and stayed there until I rebooted it again.

I mentally noted that this was weird and that this might have some security implications so I should look at it later. To be honest I don’t really like finding behaviors like this when I am not looking for them explicitly, because when this happens, I am prone to feeling obsessively responsible to investigate. I start to feel like I must make sure that there is no serious issue under the hood that others missed. In this case, well, there was.

CHAPTER 2:
WHAT JUST HAPPENED?

As I promised myself, I started looking at this behavior again the next day. After rebooting the phone, putting in the incorrect PIN 3 times, entering the PUK, and choosing a new PIN, I got to the same “Pixel is starting…” state.

I played with this process multiple times, and one time I forgot to reboot the phone, and just started from a normal unlocked state, locked the device, hot-swapped the SIM tray, and did the SIM PIN reset process. I didn’t even realize what I was doing.

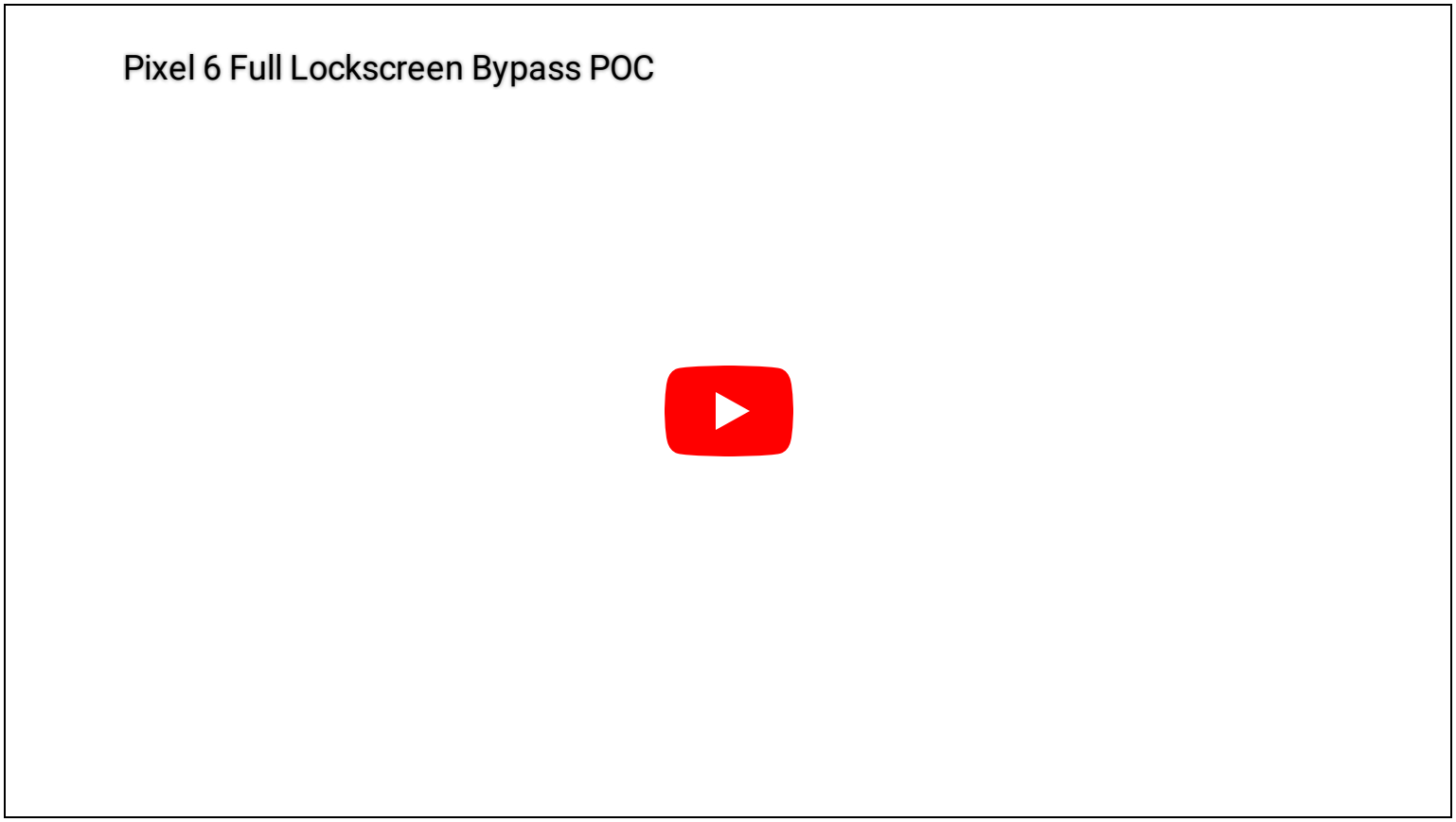
As I did before, I entered the PUK code and choose a new PIN. This time the phone glitched, and I was on my personal home screen. *What? It was locked before, right?*

This was disturbingly weird. I did it again. Lock the phone, re-insert the SIM tray, reset the PIN… And again I am on the home screen. *WHAT?*

My hands started to shake at this point. *WHAT THE F**K? IT UNLOCKED ITSELF?*

After I calmed down a little bit, I realized that indeed, this is a got damn full lock screen bypass, on the fully patched Pixel 6. I got my old Pixel 5 and tried to reproduce the bug there as well. It worked too.

Here is the unlock process in action:



Since the attacker could just bring his/her own PIN-locked SIM card, nothing other than physical access was required for exploitation. The attacker could just swap the SIM in the victim's device, and perform the exploit with a SIM card that had a PIN lock and for which the attacker knew the correct PUK code.

CHAPTER 3:
GOOGLE'S RESPONSE

I sent in the report. It was I think the shortest report of mine yet. Only took 5 simple steps.

Google (more precisely the Android VRP) triaged & filed an internal bug within **37 minutes**. That was really impressive. Unfortunately, after this, the quality and the frequency of the responses started to deteriorate.

During the life of this bug, since the official bug ticket was not too responsive, I sometimes got some semi-official information from Googlers. I actually prefer to only get updates on the official channel, which is the bug ticket and which I can disclose, but since I was talking with some employees, I picked up on bits and pieces.

Also, it's worth mentioning here that before reporting, I checked the Android VRP reward table which states that if you report a lock screen bypass that *would affect multiple or all [Pixel] devices*, you can get a maximum of $100k bounty. Since I ticked all of the required boxes, I

sort of went into this thinking that this bug has a strong chance of actually getting rewarded $100k.

After it got triaged, there was basically a month of silence. I heared that it might actually be closed as a duplicate. Apparently somebody already reported it beforehand, even though it was my report that actually made them take action. Something seemingly went wrong with processing the original report. Indeed, 31 days after reporting, I woke up to the automated email saying that *"The Android Security Team believes that this is a duplicate of an issue previously reported by another external researcher."* This was a bit of a signature bug bounty moment, a bug going from $100k to $0. I couldn't really do anything but accept the fact that this bug is now a duplicate and will not pay.

Almost two months have passed after my report, and there was just silence. On day 59 I pinged the ticket, asking for a status update. I got back a template response saying that they are still working on the fix.

Fast forward to September, three months after my report. I was in London, attending Google's bug hunter event called ESCAL8. The September 2022 patch just came out, I updated my phone and one night in my hotel room I tried to reproduce the bug. I was hoping that they might have fixed it already. No. I was still able to unlock the phone.

This hotel room incident really freaked me out. I felt like I worry and care so much more about the bug getting fixed than Google themselves. Which should not be the case. Even if I am overreacting. So that night I started reaching out to other Googlers who were at the event with us.

The next day I ended up explaining my situation to multiple people, and I even did a live demo with some of the Pixels inside Google's office. That was an experience. We didn't have a SIM ejection tool. First, we tried to use a needle, and somehow I managed to cut my finger in multiple places, and my hand started bleeding. I had a Google engineer put a band-aid on my finger. (Who else can say that??) Since the needle didn't work, we started to ask around and one very kind woman gave us her earrings to try with. It worked! We swapped the SIMs, and manage to, with some difficulties, unlock the devices. Now I felt better that people seemed to care about the issue.

me the day after cutting my finger

I put a disclosure deadline for October 15, but the Android VRP team responded by saying that the bug will not be patched in October yet. They were aiming at December. This seemed way too far for me, considering the impact. I decided to stick with my October deadline.

After talking to some Googlers about this October deadline, a member of the Android VRP team personally commented on the bug ticket, and asked me to set up a call to talk about the bug, and share feedback. We had a Meet call with multiple people, and they were very nice and listened to my whole story about being in the dark for months, only getting template reponses (even for the $100k → $0 duplicate), and overall feeling like I care more about this bug than Google. They said that the fix is now planned to go out in November, not December. Still, my deadline was set to October.

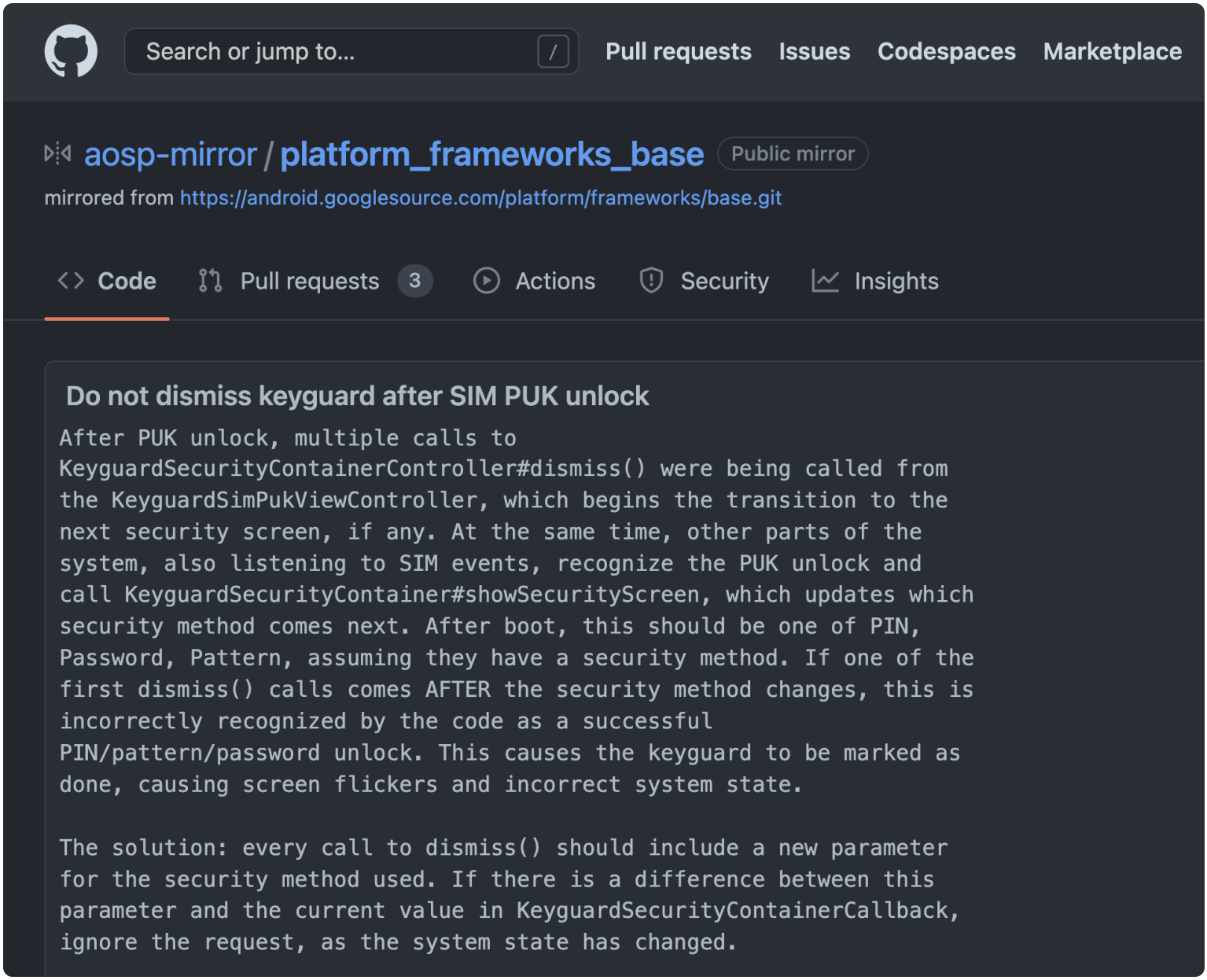
Two weeks after our call, I got a new message that confirmed the original info I had. They said that even though my report was a duplicate, it was only because of my report that they started working on the fix. Due to this, they decided to make an exception, and **reward $70,000** for the lock screen bypass. I also decided (even before the bounty) that I am too scared to actually put out the live bug and since the fix was less than a month away, it was not really worth it anyway. I decided to wait for the fix.

You can read the full conversation on feed.bugs.xdavidhu.me.

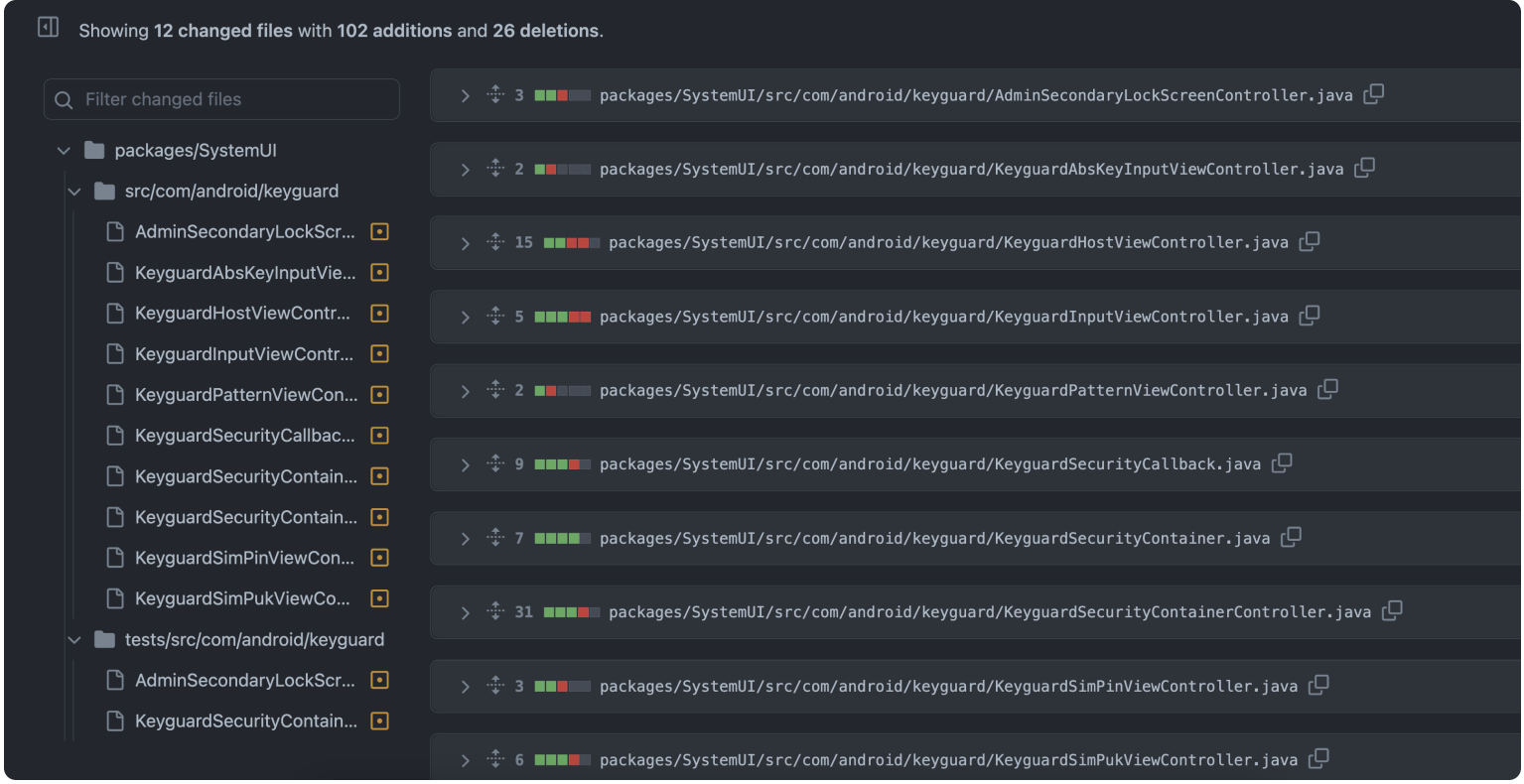
All in all, even though this bug started out as a not-too-great experience for me, the hacker, after I started “screaming” loudly enough, they noticed, and really wanted correct what went wrong. Hopefully they treated the original reporter(s) fairly as well. In the end, I think Google did pretty well, although the fix timelime still felt long for me.

But I’ll let you be the judge of it.

Since Android is open source, the commit fixing this issue with all of the code changes is visible publicly:



The first thing that surprised me when I first looked at this commit was the number of files changed. I previously thought that this bug would only have a simple one-liner fix, removing the incorrect line of code responsible for triggering an unlock. But it was not that simple:



After reading the commit message and the code changes, I think I was able to get a rough picture of what happened under the hood. Keep in mind that I am not an Android engineer, so I want to keep this high level.

Seems like, on Android, there is a concept of a “security screen”. A security screen can be multiple things. The PIN entry screen, the fingerprint scanning screen, the password entry screen, or, in our case the SIM PIN and SIM PUK entry screen.

These security screens can be stacked “on top” of each other. So for example when the phone was locked, and the SIM PIN entry was visible, it had a SIM PIN security screen on top of a “fingerprint security screen”.

When the SIM PUK was reset successfully, a `.dismiss()` function was called by the PUK resetting component on the “security screen stack”, causing the device to dismiss the current one and show the security screen that was “under” it in the stack. In our example that was the fingerprint security screen.

Since the `.dismiss()` function simply dismissed the current security screen, it was vulnerable to *race conditions*. Imagine what would have happened if something in the background would have changed the current security screen before the PUK resetting component got to the `.dismiss()` call? Would the PUK component dismiss an unrelated security screen when it finally calls `.dismiss()`?

This seems like exactly what happened. Some other part of the system was monitoring the state of the SIM in the background, and when it detected a change, it updated which security screen was currently active. It seems like this background component set the normal e.g. fingerprint screen as the active security screen, even before the PUK component was able to get to its own `.dismiss()` function call. By the time the PUK component called `.dismiss()` function, it actually dismissed the fingerprint security screen, instead of just dismissing the PUK security screen, as it was originally intended. And calling `.dismiss()` on the fingerprint security screen caused the phone to unlock.

The Android engineers seemingly decided to refactor the `.dismiss()` function and made it require an additional parameter, where the caller can specify what type of security screen it wants to dismiss. In our case, the PUK component now explicitly calls `.dismiss(SecurityMode.SimPuk)`, to only dismiss security screens with the type of `SimPuk`. If the currently active security screen is not a `SimPuk` screen (because maybe some background component changed it, like in our case), the dismiss function doesn't do anything.

This seems to me like a pretty elegant and robust solution to defend against this, and future race conditions as well. I was not expecting to cause this big of a code change in Android with this bug.