JUST SECURITY
A FORUM ON LAW, RIGHTS, AND U.S. NATIONAL SECURITY

Smartphone “Backdoors” and Open Computing

By Julian Sanchez

October 6, 2014

Recently I wrote a longish post over at the Cato Institute’s blog trying to deflate some of the increasing hysteria over the announcement that Apple and Google will be turning on full device mobile encryption by default. That means that nobody without the user’s passcode—including Apple itself, should the police come knocking with a warrant and a locked phone—will have the ability to unlock the data stored on those phones. Previously, Apple did retain the ability to access much of that data, because some of it was only encrypted with the keys stored on the device itself—keys that didn’t bake in the user’s passcode for extra security.

I’ve spent much of the past week trying to better understand Apple’s security architecture, and the method they formerly used to provide law enforcement with access to user data. What I’ve read, and learned from talking with actual crypto experts, has affirmed my confidence in two core points. First: Apple is not just inexplicably thumbing its corporate nose at law enforcement. They are fixing a poor security design choice that previously left specific types of sensitive data on phones inadequately protected. Second: Apple, with its closed ecosystem, might actually be unusually well situated to satisfy the FBI’s demand for backdoors, but the idea is in profound conflict with more open computing models.

The iPhone Never Had a “Backdoor”—Just Bad Security Design

Much of the coverage of Apple’s design changes for iOS 8, the latest version of their mobile operating system, has somewhat misleadingly characterized the company as having removed a backdoor for law enforcement. You might read such descriptions and think that Apple had previously deliberately programmed in a kind of special “police access” feature on their phones, which they’ve decided to eliminate as some kind of symbolic gesture of resistance. Apple itself may be cultivating that impression, since it’s at the least sexier than the more mundane truth: Previous versions of iOS employed poor security design, which Apple was able to exploit to retrieve certain types of user data, and which they’ve now remedied.

In fact, much personal data on the phone was already being strongly protected on the iPhone by default under iOS 7, and therefore already inaccessible to either Apple or law enforcement. As you can see from their old law enforcement access policy, there was never any all-purpose backdoor allowing Apple to decrypt data for police, and Apple's ability to help execute search warrants was restricted to particular types of information, albeit types frequently desired by law enforcement :

Please note the only categories of user generated active files that can be provided to law enforcement, pursuant to a valid search warrant, are: SMS, photos, videos, contacts, audio recording, and call history. Apple cannot provide: email, calendar entries, or any third-party App data.

Why, you might ask, could Apple provide police with some types of information but not others? Because the iPhone employs four different "classes" of protection for stored data, three of which—including the default class of protection for all data stored in third-party apps—involve "entangling" the user's personal passcode with a set of cryptographic keys stored securely on the device. That means the user's passcode (unknown, of course, to Apple) is itself part of the encryption key protecting the files, and there is simply no way to make the phone decrypt those files unless you know or can guess the code. The exception is a data class Apple's security white paper, aptly enough, calls "No Protection":

This class key is protected only with the UID, and is kept in Effaceable Storage. ***Since all the keys needed to decrypt files in this class are stored on the device, the encryption only affords the benefit of fast remote wipe.*** *If a file is not assigned a Data Protection class, it is still stored in encrypted form (as is all data on an iOS device).*

Read the bolded sentence again, because it's important. Yes, even "no protection" files are encrypted, and that will be enough to keep the average mugger out—at present it appears that even government agencies need Apple's help to get in to newer phones. But Apple fairly clearly does not regard this as ironclad security against a sophisticated attacker.

As you've probably already inferred, the reason Apple would have been able to extract a user's "SMS, photos, videos, contacts, audio recording, and call history" for police without the user's passcode is that until iOS 8, the iPhone marked all those things "NSFileProtectionNone." As iPhone security expert Jonathan Zdziarski bluntly puts it:

With iOS 8, Apple has finally brought their operating system up to what most experts would consider "acceptable security". My tone here suggests that I'm saying all prior versions of iOS had substandard security – that's exactly what I'm saying. I've been hacking on the phone since they first came out in 2007. Since the iPhone first came out, Apple's data security has had a dismal track record. Even as recent as iOS 7, Apple's file system left almost all user data inadequately encrypted (or protected), and often riddled with holes – or even services that dished up your data to anyone who knew how to ask.

Ultimately Apple's big headline-grabbing security change comes down to a change in categories: Apple hasn't removed any "police access" functions from its code, or even added any new encryption. They've just decided to apply the old encryption to your photos as well as your e-mails. The FBI is not really complaining that Apple has closed a backdoor, because there never was a backdoor in the conventional sense. Rather, they're complaining that Apple used to have badly designed security, and now they've fixed it.

The FBI Wants Every Phone to Work Like an iPhone

Thinking about the way Apple previously provided some information from iPhones also helps make clear how their ability to do this at all was closely tied to Apple's tightly integrated, "we control the horizontal, we control the vertical" business model. Lots of computing device manufacturers (and users) prefer a more open model—one we'd effectively have to outlaw if we wanted all manufacturers to be able to do what Apple formerly did.

To see why, I need to say a little more about what I think I now understand about Apple's hardware and software security architecture. Every iPhone's A7 processor contains a dedicated subsystem called the Secure Enclave responsible for the iPhone's security and encryption functions. Burned into that chip when it's manufactured is a Unique ID (UID) that serves as a kind of master key for all the phone's crypto operations. Even the data marked "NSFileProtectionNone" by the iPhone is at least encrypted with (a key based on) this number. Apple itself doesn't keep any record of the UID associated with each device, and it appears that it's not currently feasible to extract the key from the chip—again, probably even for Apple, though the most advanced intelligence services may have a method to do so. Normally the chip won't use its keys to unlock anything without the user's authorization passcode. But of course, if you can change the chip's operating instructions, its firmware, then even if you can't read the keys burned into the chip, you can make it use the keys to decrypt data and effectively unlock the phone… at least if those are the only keys that were used to lock the data.

Obviously, this would be a truly massive security flaw—not a "hole" so much as a gaping chasm—if anybody with physical control of the chip could rewrite those instructions. Anyone who stole an encrypted iPhone would be stealing the encryption keys (or at least, the ability to use the encryption keys) along with it. The reason building the keys into the lock doesn't render the lock completely worthless in this case is that Apple hardware is designed not to run any software but Apple's, even if the user wants to run something else. This is why many users "jailbreak" their phones so they can run software without Apple's permission—a reminder, incidentally, that Apple's exclusive control has not exactly proven foolproof.

The most fundamental way Apple exercises control is by burning their public key into the phone's BootROM, the chip that controls the first steps of the booting up process. That means the chip should refuse to run any start-up code that hasn't been signed by Apple's secret private key. (If that key is compromised, of course, all bets are off—though if that happens, users will have much bigger problems to worry about.) When police sent in an iPhone (and a search warrant), Apple was thus able to use their developer key to install a custom boot-loader that commanded their cryptochip to unlock all the data it could without asking for a passcode, which until iOS 8 included quite a bit of sensitive data in the “no protection” class. In theory no other attacker should be able to pull off a similar trick without Apple’s secret key. In theory.

But of course, not all companies take Apple’s micromangerial, soup-to-nuts approach to their devices. Even manufacturers that employ some version of “code signing” to guard users against malicious modifications to a machine’s default operating system often allow the owner of a device to choose to run different software if they so choose. Google’s Chromebooks, for example, default to checking that they’re running code signed by Google’s private key, but unlike iPhones, they feature a “developer switch” that permits the user to opt out:

If you don’t want to boot Google-verified software — let’s say you built your own version of Chromium OS — no problem. You can flip the developer switch on your device and use the Chromebook however you’d like. It’s yours, after all!

And that’s been the norm for general-purpose computers for decades: Once you’ve bought the hardware, you may purchase, download, or write your own applications—even your own operating system—and run them on that hardware without ever asking the manufacturer for permission. It’s yours, after all.

If a computer is not designed to only run code approved by a single corporation, however, then it becomes much less realistic to rely on the operating system to keep attackers’ grubby mitts away from any keys stored on the device. Which is probably one reason Android devices have encrypted with a user-supplied password—putting the data beyond the reach of Google or the device manufacturer—since 2011. (Google has drawn the ire of law enforcement by announcing that this encryption will now be activated by default—but it has long been present as an option for the user.) Needless to say, regardless of what Google decides to do, so long as the devices are open, nobody can stop the open source community from releasing alternative operating systems—maybe even just modified versions of Android—that provide strong, backdoor-free encryption by default.

Pundits who don’t understand technology very well hear that Apple used to be able to provide some information to law enforcement and simply assume that any smartphone maker could do the same without rendering users’ data hugely less secure against all other attackers. If even government agencies needed Google’s help to unlock newer iPhones, after all, doesn’t that show a company can create a “golden key” that only they can use to unlock data? Well, maybe—if every company chooses an Apple-like architecture. To demand that iPhones have a backdoor for law enforcement would require Apple to implement bad security design, but it would not require any profound or fundamental change to the nature of the iPhone. To demand that all smartphones be created with such backdoors is a hugely more radical proposition, amounting to a demand that all smartphones be designed like iPhones. It is, ultimately, a demand for the abolition of open computing devices. Which, for the most part, is why you’re hearing these demands from people who would have to ask the office IT guy to explain exactly what that means, and why it would be an astoundingly terrible idea.

Julian Sanchez is a research fellow at the Cato Institute and contributing editor for Reason magazine. Follow him on Twitter (@normative).