COM 402 exercises 2024, session 9: Mobile Security

Exercise 9.1

You are at an Android fanboy/fangirl cocktail party. One of the guests approaches you and starts boasting about his new Samsung Galaxy S24. When he goes into arguing that Android is great because all the software on his phone is open source and can be found in the AOSP, you raise your eyebrow and try to convince him that this is just not the case.

Identify three examples to convince this guest that the software stack on his phone is only partially open source.

Exercise 9.2

Great, you convinced the guest, and he slowly walks away with a thoughtful face. While you were talking to him, another person joined the conversation. She is an app developer, and you have a great conversation about the Android Framework with her. At some point during the conversation, the topic of integrating C/C++ libraries in an Android app comes up.

Let us say you want to implement an image parser in a C/C++ native library to gain more performance and have a better user experience. When loading images directly from the web in your app that uses this library, what could go wrong?

Exercise 9.3

You and the developer agree that native libraries should be used with care in Android apps. As the evening continues, the developer tells you that a couple of months ago, she found a very similar app to one of her apps in the Google Play Store. In fact, the two apps look almost identical. Your blood is pumping faster. This might be a republishing scheme!

Explain to the developer what a republishing scheme is and why it is possible on Android.

Exercise 9.4

You are about to leave the party. On your way out, you stumble into a conversation of security researchers. They are talking about PartEmu and FirmWire, and how these research prototypes enable the dynamic analysis of proprietary software components used on mobile devices. One of the researchers starts to argue that he could just use an emulator (e.g., QEMU) that supports the ARM instruction set of this proprietary software and would be able to achieve the same results. Since you actually read (parts) of the research articles published with these prototypes, you chime in and explain that a CPU emulator alone is not enough to run these components.

Come up with an argument to convince the security researcher.

Solutions to the Exercises

Solution 9.1

1. The platform-dependent parts for a given hardware platform need to be added to the AOSP to make it run on a given chipset. These platform-dependent parts are usually not open-sourced.
2. OEMs usually heavily modify the AOSP to create a unique user experience on their platform, hoping to get a competitive advantage. Many components of Samsung's "One UI" are not open-sourced.
3. The software/firmware running in the TEE, wifi chip, Bluetooth chip, and baseband chip is not open-sourced.

Solution 9.2

- If there are bugs in this native library, the app might crash, resulting in a bad user experience.
- In the worst case, this bug might lead to code execution in the context of your app, given a specifically attacker-crafted image.

Solution 9.3

1. In a republishing scheme, malware authors download an existing app from an app store, reverse engineer it, and add malicious code to it. They publish this stolen app via app stores to follow their nefarious intentions.
2. For example, one way to make money for the malware author is by adding ads to the stolen app.
3. It is possible to reverse engineer Android apps because the code is delivered in an intermediate representation (Dalvik bytecode) that contains most of the original type information. The Dalvik bytecode can, thus, be translated back to a human-readable assembly representation (e.g., smali) or even to the source code (e.g., Java).

Solution 9.4

PartEmu and FirmWire need to implement all the peripherals (e.g., memory-mapped input/output) the TEE OS or the baseband firmware, respectively, are interacting with. These peripherals are private, and it requires a lot of reverse-engineering effort to implement an emulation layer that works sufficiently accurate.