Inside Android's SafetyNet Attestation

Dec. 2017

Collin Mulliner & John Kozyrakis
collin@mulliner.org john@koz.io

Many app developers often have questions like the following: "Is the device my app runs on reliable?" "Is it trustworthy?" "Is it, god forbid, 'rooted'?"* It turns out that answering these questions is quite difficult. In an area traditionally dominated by "root detection" products and DIY techniques, Google created its own alternative, answering the following question: "OK Google, what do you think about the device I'm running in?" SafetyNet is the primary security platform used by Google to keep the Android ecosystem in check. SafetyNet Attestation is a service offered by the SafetyNet system to all Android application developers, who can use it to gain some insight into what Google believes is the state of tampering of the operating system and the device.

Unfortunately, SafetyNet Attestation is not well documented by Google. How does it work? What checks does it do? Does it really help? How can you implement it in your app without it being trivially bypassable?

Building root and/or tamper detection is hard. Hard coded checks can be easily bypassed. Google built SafetyNet Attestation so developers don't have to reimplement those security checks. Further Google constantly improves SafetyNet Attestation so once your application is using it you will get security improvements for free.

The main pitfalls of SafetyNet Attestation is that you need to implement it in the correct way and are aware of possible bypasses so you can judge the remaining security risk to your application. The keypoints are: Do not implement client side attestation validation, all checks have to be implemented and enforced in your backend. Verify the cryptographic checks before evaluating the attestation result. Implement re-try mechanisms on the client and server side. Assume Attestation is going to fail on a percentage of your clients. Plan for this, since it will happen. Design your backend in a way that allows you to ignore Attestation failures, e.g. if the Google API becomes un-responsive (localized network outage) or gets deprecated.

SafetyNet Attestation behaves differently on older Android versions. The boot state (secure boot) can not be accessed on Android 4 and 5, therefore, SafetyNet will not be able to detect unlocked bootloaders. Attacks based on unlocked bootloaders will be easier since detection is harder. Consider this for what features you support on what Android version.

Attacks:

SafetyNet can be bypassed. Bypasses always rely on the specific device and software version. Vulnerabilities in the OS and device will enable an attacker to bypass Attestation to a certain degree. Bypasses will be more expensive (more time and effort) if the device is up to date. In lab conditions a bypass will almost always work. If the device needs to function normally as a person's day-to-day phone Attestation bypasses will be harder as they impact the usability of the device.

Key Takeaways:

- Validate the Attestation data's signature
- Validate and Enforce on the server side
- Plan for clients that fail attestation
- Re-Try Attestation if it fails
- SafetyNet provides different levels of attestation on older Android versions
- Google constantly improves Attestation and thus improves the security of your app

References:

https://developer.android.com/training/safetynet/index.html
https://developers.google.com/android/reference/com/google/android/gms/safetynet/SafetyNet

https://koz.io/inside-safetynet/
https://koz.io/inside-safetynet-2/
https://koz.io/inside-safetynet-3/
https://koz.io/inside-safetynet-4/

https://www.mulliner.org/android/
https://mulliner.org/collin/publications/inside_safetynet_attestation_attacks_and_defense_mulliner2017_ekoparty.pdf

https://github.com/topjohnwu/Magiskhttp://www.blackhat.com/docs/asia-15/materials/asia-15-Sabanal-Hiding-Behind-ART.pdf

Further reading (nice attack that uses ODEX patching):

https://www.fireeye.com/blog/threat-research/2017/05/gaining-root-on-lenovo-vibe.html