The evolution of hardware isolation for smartphones

To keep a smartphone's most sensitive processes and data protected in case of attack, phone makers have turned to isolation via hardware, first through virtualization on a single processor and then through a dedicated security chip. But as advanced hackers continue to look for ways to break this hardware isolation, a shift in tactics may be required to keep the information of security-conscious users safe.

The need for hardware isolation

As smartphones evolved from on-the-go tools into digital extensions of our lives, these devices have gained more and more attention from hackers. And in many ways, mobile devices are more attractive targets than PCs because of their unique features. These include:

- **Full access:** A smartphone is essentially an all-access pass to a victim's life, acting as a repository of their photos, locations, personal messages and more. It can also serve as an access point into enterprise systems. Captured assets can reveal actionable information or simply provide fodder for additional attacks.
- **Relaxed user security posture:** Whereas an individual might think twice before clicking on a phishing email on their work computer, they may not necessarily exhibit the same awareness on the device they use throughout the day to text friends, browse social media and play games. The device's small screen contributes to this situation as it makes identifying malicious emails and websites more challenging.
- **Consumer focus:** Phone makers are in a constant battle to make their devices more attractive to consumers, meaning that new features are being added at a breakneck pace. All of this new code opens up additional opportunities for hackers to find vulnerabilities and exploit them.

  Cellular connectivity: Smartphones are designed to be always on and always connected to the cellular network. This works to a hacker's advantage by increasing the device's capacity for remote monitoring and adding to its attack surface with an interface typically outside of the control of mobile security protections.
- **Susceptibility to zero-click attacks:** While rare, zero-click attacks against smartphones enable threat actors to stealthily take total control of a device without any interaction needed from the user.
- **Capacity for spying:** When infected with spyware, a smartphone's cameras and microphones can be remotely activated to listen in on the user's in-person conversations and look in on their private spaces, essentially turning the device into an always-present spying tool.

Protecting the security subsystem

Against this backdrop, phone makers have needed a way to ensure that a smartphone's security subsystem – its set of most critical operations – is protected in case the main operating system is compromised, lest the attacker gain the ability to read encrypted data or load malicious code with more powerful permissions than even the device's operating system.

A security subsystem differs slightly between device models but typically includes cryptographic operations, the secure boot process and trusted applications.

Cryptographic operations

A device's unique, private root key (or its unique ID number used to generate a temporary private key) is critical to the cryptographic foundation of any smartphone. For these reasons, the root key is generated on the smartphone during the manufacturing process and is meant to never leave the device or even be directly accessible to other device functions.

Without the root key, an attacker who has gained control of a device's operating system may be able to exfiltrate the data from the device but not be able to decrypt it.

Along with the device's root key, the security subsystem includes mechanisms for generating additional cryptographic keys; encrypting, decrypting, signing and validating data; and accelerating cryptographic calculations.

Secure boot process

A smartphone's chain of trust typically begins with immutable initialization code that's laid down in silicon during chip fabrication and then stored in read-only memory (ROM). Once the device is powered on, the application processor (or dedicated security chip) executes the code, which starts a process where each newly loaded component cryptographically validates the next component in the chain: first the initialization code, then the bootloader (firmware), then the kernel (operating system) and finally the apps. Isolated execution environments will have their own chain of trust that loads as a prerequisite for loading the main, user-facing OS.

By subverting the secure boot process, an attacker can control the OS and thus gain virtually unlimited control over the device and its data.

Trusted applications

While one of the key goals of a security subsystem is to be as small as possible in order to minimize its attack surface, this protected area can also run trusted applications or provide critical functionality for doing so. Typical applications include the processing of biometric authentication requests, the processing of payments and the storage of passwords.

Approach #1: Trusted execution environment (TEE)

The earliest approach for barricading a smartphone's security subsystem is the trusted execution environment (TEE), a virtualized environment that's separated from the user-facing OS. A TEE is designed to run from the main processor, with hardware providing dedicated processing and memory.

ARM's TrustZone

TrustZone, which was first made available in ARM processors in 2004, is effectively the standard TEE architecture for Android smartphones. TrustZone grew in popularity because it provides easily accessible security without specialized hardware, in effect allowing phone makers to keep costs low, release products more quickly and forgo compromises on battery life and device size. As TrustZone is an architecture only, numerous implementations based on TrustZone have been developed commercially, the most deployed of which are Trustonic's Kinibi and Qualcomm's QSEE.

TrustZone consists of two virtual processors: a "secure world" for the security subsystem and a "non-secure" world for everything else, including the Android OS and user apps. The secure world has its own OS, apps and privileges. By design, anything in the main environment can't "see" or modify what is happening inside the TEE. Hardware logic dictates separate processing and memory, with each physical CPU core providing an abstraction of two virtual ones. So-called monitor mode, which runs in the secure world, manages the transition between worlds via the Secure Monitor Call (SMC) instruction.

Attacks against TrustZone

Given its close proximity to the main OS, a TrustZone-based TEE is susceptible to a number of attack methods from a hacker who's gained kernel-level privileges.

Exploitation of implementation flaws

Because TrustZone implementations are based in software, the potential for bugs always exists. One way that attackers discover coding errors is through fuzzing, which involves hitting the targeted system with massive amounts of random data in the hopes of causing a crash.

In 2019, researchers from Check Point revealed an attack chain[1] against Qualcomm's QSEE that could result in the leakage of data stored in the secure world, including financial information. After using two known bugs to break TrustZone partitions, the researchers successfully edited code to replace a trusted app's hash block after verification, in effect providing the ability to load a secure world app in the non-secure world.

Hijacked switching between worlds

The SMC-based communication system between the two worlds is a common starting point for attacks against the secure world.

With elevated privileges, an attacker can bypass SMC authentication to send a malformed SMC instruction[2] to the secure world, which typically has limited mechanisms for verifying the validity of the message. These instructions can be used to receive encrypted data from the secure world, load a module to the secure world and more.

Attackers can also intercept the switching from the secure world to the non-secure world, spoofing the process to deceive users.

Side-channel attacks

Given that the secure world and non-secure world share the same processor, the TrustZone TEE is susceptible to side-channel attacks in which attackers try indirect methods, like measuring cache access attempts or power consumption, to infer secrets from the secure world.

In the ARMageddon attacks[3], researchers from Graz University of Technology demonstrated a variety of advanced cache timing attacks across CPU cores. In one attack against TrustZone, they used a Prime+Probe approach, which entails occupying specific cache sets, scheduling the targeted program and determining which cache sets are still occupied. This technique can be used to infer private keys from the secure world.

TEEs are also susceptible to speculative execution attacks, which capitalize on the fact that chips will run instructions in anticipation of expected results. For example, a variant of Foreshadow[4] allowed attackers to read the memory of Intel's Software Guard Extensions (SGX) TEE and therefore remotely steal private keys.

Approach #2: Secure element (SE)

A dedicated security chip known as a secure element (SE) takes the concept of the TEE one step further, giving the security subsystem its own processor that's physically isolated from the main application processor. First introduced by Apple in 2013, most Android makers now have their own dedicated security chip as part of the system on a chip (SoC) they use.

Apple's Secure Enclave

Apple's Secure Enclave is a dedicated security chip with its own microkernel and secure boot process. It's not directly accessible by the main OS or apps; its only communication with the application processor occurs via an interrupt-driven mailbox and shared memory data buffers. Apple designed the Secure Enclave with a number of security features: the amount of flashable storage is small, minimizing the attack surface; physical tamper detection prevents forensic examination of protected keys and other material; and a lower clock speed provides protection against side-channel attacks.

The main purpose of the Secure Enclave is to generate the device's Unique ID (UID) number during the manufacturing process and keep it segregated from the rest of iOS. Each time the iPhone starts up, the device uses the UID to create a new temporary key that both encrypts and authenticates the Secure Enclave's memory.

Attacks against the Secure Enclave

Though the Secure Enclave is a giant upgrade in security from the TEE, attacks against it are still possible given that it shares resources with the application processor.

Exploitation of implementation flaws

The Secure Enclave's memory controller can be poked at from the application processor. In 2020, Chinese hackers from the Pangu Team reportedly found an unpatchable exploit[5] for this memory controller, enabling an attacker to take control of the specific register memory that manages the Secure Enclave's memory usage. From there, an attacker can alter how the memory isolation system of shared memory between the Secure Enclave and the main processor functions. In turn, this could feasibly be used to acquire data that would normally be viewed and used only by the Secure Enclave, including private keys.

This particular attack would require physical access to the device but nonetheless demonstrates the potential consequences of having the memory controller accessible from the main processor.

Side-channel attacks

The Secure Enclave shares a power manager with the application processor, opening the door to fault attacks.

As a research team from Columbia University demonstrated with their remote CLKSCREW attack[6] against TrustZone, when a power manager is shared between environments, it's possible for one environment to overstress the other environment's hardware, thus overclocking the CPU. By recording the results and comparing them with the results at normal operation, an attacker can induce and detect single-byte errors that can later be used to extract private keys from the isolated environment.

Apple employs monitoring circuits for power and clock speed to ensure that the Secure Enclave operates within a limited envelope. This raises the difficulty of such a fault attack, but it's feasible that an attacker can stress the hardware to the limits of this range and then infer secrets based on these measurements.

Approach #3: External, phone-independent hardware

At Privoro, we anticipate that advanced attackers will continue to find ways of exploiting the Secure Enclave and other security chips, using any and all overlap between secure and non-secure systems to find and leverage exploits. And yet, by virtue of being on the same SoC, there's only so much separation that can be created between environments. To truly protect the security subsystem, it has to be relegated to a high-security system outside of the smartphone itself. We believe that SafeCase™, our smartphone-coupled security device, will be that system of choice.

A key function of SafeCase is that it serves as a secondary, special-purpose computing device to the smartphone. Critically, SafeCase operates within its own closed, end-to-system, meaning that though it pairs with the coupled smartphone, it remains functionally independent. Architecturally, SafeCase shares many of the same features as the Secure Enclave, including a hardware root of trust, secure processing and secure storage. SafeCase also replicates key smartphone components, including motion sensors.

Elements of the smartphone's security subsystem can be offloaded to SafeCase, including the storage of private keys and any cryptographic operations using these keys. Trusted applications can also be built that leverage the device's integrated hardware components.

This two-system approach is an architectural evolution that materially increases security. Even if an attacker were able to remotely compromise the SafeCase-coupled smartphone, they would have no ability to cross over to the hardened SafeCase platform to access security-critical elements like stored private keys.

For users and organizations subject to sophisticated attackers, the concept of external, phone-independent hardware holds the key to achieving secure mobility while simultaneously leveraging the connectivity and productivity gains represented by commercial smartphones.

SafeCase Carbon

SafeCase Carbon X

SafeCase Onyx

SafeCase Shadow

Sources

1. Makkaveev, Slava, “The Road to Qualcomm TrustZone Apps Fuzzing,” Check Point, November 14, 2019.

2. Shen, Di, “Exploiting Trustzone on Android,” Black Hat USA 2015, August 6, 2015.

3. Lipp, Moritz, Daniel Gruss, et al., “ARMageddon: Cache Attacks on Mobile Devices,” 25th USENIX Security Symposium, August 10, 2016.

4. Lindell, Yehuda, “Foreshadow, SGX & the Failure of Trusted Execution,” Dark Reading, September 12, 2018.

5. Owen, Malcolm, “Security Enclave vulnerability seems scary, but won’t affect most iPhone users,” AppleInsider, August 3, 2020.

6. Tang, Adrian, Simha Sethumadhavan, et al., “CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management,” 26th USENIX Security Symposium, August 16, 2017.