1 SeCloak: ARM Trustzone-based Mobile Peripheral Control

1.1 Introduction

1.1.1 Overview

SeCloack uses ARM Trustzone to provide reliable on-off controls for peripherals even when the platform software is compromised. This is important to protect users against databreaches because lots of personal smart devices contain more and more sensitive data. It is important to note that these devices also contain highly sophisticated hardware security in their architecture. These security features are often used for financial transactions.

1.1.2 Goals

SeCloak would allow users to reliably turn off certain peripherals like sensors or radios using hardware mechanisms. The design goals are

1. Allow users to securely and verifiably control peripherals on their device
2. Maintain system usability and stability
3. Minimize the trusted code base
4. Not change existing software including apps, frameworks or OS kernels

1.2 Background & Related Work

1.2.1 Virtualization

One approach is to run the platform OS as a guest within a virtual machine, leaving a hypervisor in control of peripheral devices. These systems are designed for isolating individual applications from the OS, but don't provide a mechanism to reliably control generic peripherals. These approaches seek to protect the integrity and confidentiality of I/O data paths while maintaining full functionality, which comes at the expense of a large TCB.

1.2.2 Hardware Security

Beyond virtualization, modern architectures offer trusted hardware components e.g., Intel SGX and ARM TrustZone, which can be used to isolate software components from an untrusted platform OS. By default, TrustZone supports a single isolated execution environment (the secure world), with a secure boot process that can be used to verify the bootloader and the secure-world kernel. TrustZone hardware protections are used to protect sensitive memory regions, it doesn't address peripheral access control.

1.2.3 ARM TrustZone

TrustZone is the basis for commercial products that implement support for isolated execution of secure applications and for secure IO. Trust Leases allow applications to request (with user approval) leases to place the device in a restricted mode until some terminal condition is met. Their threat model assumes that the platform OS is trusted and correct. Viola enables custom, per-peripheral notifications whenever the I/O peripheral device is being used. SeCloak and Viola are complementary, the user could use SeCloak to disable devices and rely on Viola to notify them when specific enabled devices are in use.

1.3 Design Overview

1.3.1 NS-bit

TrustZone security model posits that the CPU can be in a "Non-Secure" (NS) mode or "Secure" mode, and all memory and bus accesses are tagged with the CPU mode using an extra "NS-bit" bus line. Different peripherals, including parts of RAM, IO devices and interrupts, can be configured to be available only in Secure mode. In our design, Linux runs as the NS-kernel, and SeCloak (with its secure kernel) controls the secure modes of the CPU.

1.3.2 Security Configuration Register

The ARM Security Configuration Register (SCR) contains the "NS" bit that describes whether the core executing code is in the non-secure or secure mode. It is this setting of the SCR that enables SeCloak to trap and emulate instructions issued by the Linux kernel. The SCR also contains similar configuration bits for interrupts, which SeCloak uses to listen for user input and support secure system reboot.

1.3.3 TrustZone Address Space Controller

ARM provides a TrustZone Address Space Controller (TZASC) that can partition portions of RAM such that they are available only to secure mode accesses, enabling isolation of the s-kernel memory from Linux.

1.3.4 Central Security Unit

A TrustZone compatible component, the Central Security Unit (CSU), that extends the secure/non-secure access distinction to peripherals. The CSU can be used to enable secure-only access for different peripherals (which SeCloak uses), and also for programming access to various bus DMA masters (e.g., the GPU). SeCloak uses a purpose-built kernel (the s-kernel) that runs in TrustZone secure mode. The s-kernel programs each of these components (the SCR, the TZASC, the CSU) as required, both at initialization and runtime to enable SeCloak.

1.3.5 Threat Model

The threat model assumes that the device hardware is not malicious, and that the state of the hardware (number and type of IO devices, their physical addresses and buses, interrupts, etc.) is encapsulated in a "device tree" file that is signed by a trusted source. Beyond the hardware, we assume that the boot ROM and boot-loader are trusted and correctly loads the s-kernel. The s-kernel is also trusted and assumed correct. All other software in the system may be faulty or malicious. This includes any app the user may run, any framework layer (such as Android), any kernel modules, and the NS-kernel itself.

1.3.6 Secure Kernel

SeCloak's secure kernel is called the s-kernel. A signed device tree describes available hardware and protections to the s-kernel. Modern devices are equipped with secure, tamper-proof, non-volatile storage, into which device manufacturers embed (hashes of) public keys. The devices contain a one-time programmable Boot ROM that has access to these keys, which are "fused" onto the hardware.

1.3.7 Device Tree

The device tree structure describes the hardware devices present in a given system and how they are interconnected, with each node representing an individual device. It is important to note that the device tree, by our assumptions, must be an accurate and complete description of the hardware.

1.3.8 Booting

Upon boot, the s-kernel initializes hardware defaults prior to launching the NS-kernel. Specific steps include setting control and security registers to appropriate defaults, and setting memory protections such that the NS-kernel cannot overwrite the s-kernel's state. The s-kernel initializes its internal data structures by initializing the system MMU with virtual memory page table mappings for various regions, including regions for non-secure RAM, s-kernel heap, and for MMIO devices. The s-kernel also starts the non-boot CPUs, and initializes per-core threads and their contexts. Faults and calls from the NS-kernel transition the CPU into a monitor mode, and the s-kernel initializes the secure monitor with its stack pointer and call vector. Finally, the s-kernel opens and parses the device tree.

1.3.9 Non-Secure Kernel

The current version of the app is simple, allowing users to set ON/OFF preferences for the devices on our prototype board. Along with individual devices, the app allows users to choose different operating modes (e.g.,

Airplane, Stealth) and also provides the state of groups of peripherals (e.g., all networking devices.).

1.4 Results & Evaluation

1.4.1 Overhead

The download and upload performance shows that there is no visible impact of interception and emulation on WiFi transfers, despite an appreciable increase in execution time for individual load and store instructions. This is because the WiFi driver and controller, like all modern bulk data transfer devices, uses DMA to transfer packets. Once the controller firmware is loaded, and the DMA tables configured, each packet transfer (which can be many thousand bytes) requires very few (tens) MMIO instructions to initiate the DMA. We believe this result indicates that SeCloak can be used, even for high performance peripherals, without significant impact on user-perceived performance.