Shadow-Box v2:
The Practical and Omnipotent Sandbox for ARM

Seunghun Han, Jun-Hyeok Park
(hanseunghun || parkparkqw)@nsr.re.kr

Wook Shin, Junghwan Kang, HyoungChun Kim
(wshin || ultract || khche)@nsr.re.kr
Who Are We?

- Senior security researcher at NSR (National Security Research Institute of South Korea)
- Speaker at Black Hat Asia 2017 and HITBSecConf 2016/2017
- Author of the book series titled “64-bit multi-core OS principles and structure, Vol.1&2”
- a.k.a kkamagui, @kkamagui1

- Senior security researcher at NSR
- Embedded system engineer
- Interested in firmware security and IoT security
- a.k.a davepark, @davepark312
Last Year We Presented...

We introduced **Shadow-box v1**

---

**Myth and Truth about Hypervisor-Based Kernel Protector:**
The Reason Why You Need Shadow-Box

Seunghun Han, Jungwhan Kang
(hanseunghun || ultract)@nsr.re.kr

---

**Shadow-Box:**
Lightweight Hypervisor-Based Kernel Protector

Seunghun Han, Jungwhan Kang
(hanseunghun || ultract)@nsr.re.kr
Goal of This Year is...

Linux
Shadow-Box for x86
VT-x, VT-d (Virtualization Technology)

Linux
Shadow-Box for ARM
TrustZone (Virtualization Technology)

We will introduce Shadow-box v2
Background

Design

Implementation

Demo. and Conclusion
(with Black Hat Sound Bytes)
REMIND:
Linux Kernel is Everywhere!
- The Linux kernel suffers from rootkits and security vulnerabilities
  - Rootkits: EnyeLKM, Adore-ng, Sebek, suckit, kbeast, and so many descendants

Devices that use Linux kernel share security threats
- Kernel-level (Ring 0) protections are not enough
  - Lots of rootkits and exploits work in the Ring 0 level
  - Protections against them are often easily bypassed and neutralized
    - Kernel Object Hooking (KOH)
    - Direct Kernel Object Manipulation (DKOM)

**REMIND:**

Melee Combats at the Kernel-level

Protections need an even lower level (Ring -1)
Leveraging virtualization technology (VT)

- VT separates a machine into a host (secure world) and a guest (normal world)
- The host in Ring -1 can freely access/control the guest in Ring 0 (the converse doesn’t hold)
- VT-equipped HW: Intel VT-x, AMD AMD-v, ARM TrustZone

REMIND:
Taking the Higher Ground

Shadow-Box v2 focuses on ARM TrustZone!
- ARM TrustZone
  - is a security extension of ARM processor and hardware-based security
  - separates a machine into the secure world and normal world

- Trusted Execution Environment (TEE)
  - is a secure area of ARM processor
  - protects integrity and confidentiality of data in memory and storage
Lords of the TEE

Samsung

TEE of KNOX

Red Ocean...

Oh, No...

Qualcomm

QSEE
- TEEs are **proprietary**
  - Their source codes are not published
  - Use of the source code is restricted

- TEEs are **not portable**
  - They are designed for their own processors
  - So, they are not applicable in different processors
Restrictions on Lords of the TEE (2)

- To wrap it up, their TEEs are not suitable for various ARM-based devices
  - There are so many ARM processor vendors such as Broadcom, NXP, MediaTek, Allwinner, etc.
  - Manufacturers choose low-cost ARM SoC for their products
    - The types and vendors of ARM SoC in products are different depending on manufacturing date

We need an open source and portable TEE!
OP-TEE: Open Portable TEE

- OP-TEE is an open source TEE
  - You can change everything that you want
  - Linaro supports and maintains OP-TEE
    - Linaro is an association of ARM, Freescale, IBM, Samsung, ST, TI

- OP-TEE supports many kinds of SoCs and devices
  - OP-TEE supports more than fourteen devices including Raspberry Pi 3 and QEMU
  - OP-TEE has well-defined architecture, so you can port OP-TEE to your device easily
- OP-TEE follows GlobalPlatform specifications
  - GlobalPlatform makes Trusted Execution Environment (TEE) specifications
  - GlobalPlatform is an association of Samsung, Qualcomm, AMD, APPLE, Trustonic, NXP
  - Many companies follow the specifications, so you can port your trusted application to other TEE
Architecture of OP-TEE

Secure World

- Dynamic Trusted Application
- TEE Internal APIs
- Static Trusted Application
- OP-TEE Trusted Kernel

Normal World

- Tee-suppliant
- TEE Client API
- OP-TEE Driver
- Linux Kernel

User

Kernel
Background

Design

Implementation

Demo. and Conclusion
(with Black Hat Sound Bytes)
REMIND:

Security Architecture in Shadow Play

Activities in OS

Ring -1 Monitoring Mechanism

Security Monitor (Shadow-Watcher)

(Light-Box)
We named this architecture "Shadow-box"
Architecture of Shadow-Box for x86

Host (Ring -1)

- Shadow-Watcher (Monitor)

- Shared Kernel (Read/Write Permission)

Guest (Ring 0~3)

- User (Read/Write Permission)

- Shared Kernel (Read-only Permission)

Shared Area

- Monitor, control

- Shadow-Watcher

- Shared Kernel Only

- Shared Kernel and User

Light-Box (Lightweight Hypervisor)
Architecture of Shadow-Box for ARM

Secure World (Ring -1)

- Shadow-Watcher (Trusted App.)
- Trusted Kernel

Normal World (Ring 0~3)

- User Application
- Shadow-Watcher Client
- IMA
- Normal Kernel

Light-Box (Trusted App. and Trusted Kernel)
Integrity Measurement Architecture (IMA)

- Can check hashes or signatures of files and prevent the system from unauthorized executable files
- Can store measurement value in Trusted Platform Module (TPM)
- Is included Linux Kernel since 2.6.30!

IMA needs to manage hashes or signatures

- You need to make hashes or signatures of good executable files
- IMA is hard to be used for general purpose environment, but it is good for special purpose environment such as embedded systems
What can Shadow-Box v2 Do?

- Shadow-box v2 (for ARM) protects Linux kernel from
  - Unauthorized executable file attacks
    - IMA in kernel verifies signatures of executable files
  - Static kernel object attacks
    - Static kernel object = immutable at runtime
    - Code modification and system table modification attacks
  - Dynamic kernel object attacks (x86 only and future work!)
    - Dynamic kernel object = mutable at runtime
    - Process hiding and module hiding, function pointer modification attacks
Static Kernel Object Protection (1)

**Secure World**

- **Shadow-Box Trusted App.**
  - Keys
  - Measured Results
  - Page Hashes

  3. Compare Hashes

- **Integrity Checker**

- **Light-Box Trusted App.**
  - OP-TEE
  - Trusted Kernel

1. Request

2. Calculate Page Hashes

**Normal World**

- **Shadow-Watcher Client**
  - Remote Attestation

- **Periodic Kernel Integrity Monitor**

1. Request

2. Calculate Page Hashes

3. Compare Hashes

4. Return the Result

5. Report Errors

- **OP-TEE Driver**
- **Linux Kernel**
- Page hash-based integrity monitor
  - Is a simple and intuitive mechanism which is widely used!
    - But, the attacker can guess when the page is measured and do transient attack!
  - Needs a mechanism to randomize the measurement timing
- So, Shadow-Box randomizes page order
  - Shadow-watcher trust application shuffles pages after integrity measurement is completed
Workload-Concerned Kernel Monitoring

- Adaptive mechanism
  - Changes check period for measurement depending on system workload
  - Increases the period to keep performance as workload increases

![Graph showing relationship between Check Period and CPU Workload]

- Check Period (time)
- Maximum Check Period (Check Infrequently)
- Minimum Check Period (Check Frequently)
- CPU Workload
Remote Attestation

Secure World

- Shadow-Box Trusted App.
  - Keys
  - Measured Results
  - Page Hashes
  - Integrity Checker
  - Light-Box Trusted App.
    - OP-TEE Trusted Kernel

Normal World

- Shadow-Watcher Client
  - Periodic Kernel Integrity Monitor
  - Remote Attestation
  - Remote Server

OP-TEE Driver
Linux Kernel

1. Request with Encrypted Nonce
2. Send the Request
3. Get Encrypted Results of Measurement
4. Return the Encrypted Result
5. Bypass the Results
Executable File Verification with IMA

**Secure World**

- Shadow-Box Trusted App.
  - Keys
  - Measured Results
  - Page Hashes
  - Integrity Checker
- Light-Box Trusted App.
  - OP-TEE
  - Trusted Kernel

**Normal World**

- Shadow-Watcher Client
  - Signed App.
  - Unsigned App.
  - Modified App.

IMA
- OP-TEE Driver
- Linux Kernel

*Verifying and Executing*
Background

Design

Implementation

Demo. and Conclusion
(with Black Hat Sound Bytes)
Target Board: Raspberry Pi 3

- Raspberry Pi board
  - Is the most famous embedded hardware
  - Supports many kinds of OS such as Raspbian, Ubuntu, and Windows 10 core

- Raspberry Pi 3 model B specification
  - Quad Core 1.2GHz Broadcom BCM2837
  - 1GB RAM and HDMI
  - BCM43438 wireless LAN and bluetooth
  - 40-pin extended GPIO
- Raspberry Pi is the best board for a prototype, but…
  - CPU supports ARM TrustZone feature only
  - DRAM and flash controller do not support it
  - Raspberry Pi does not have secure boot feature
  - The secure world is not really secure and just for a prototype!

- If you want a fully-featured board, choose another board!
  - OP-TEE supports many kinds of embedded boards such as Juno board, HiKey board, ATSAMA5D2-XULT board, and i.MX7Dual SabreSD Board
How to Integrate Shadow-Box with Raspberry Pi

Raspbian OS
- Raspbian’s Kernel
+ OP-TEE’s Kernel with IMA Patch
+ OP-TEE’s Secure Kernel
+ Shadow-Box
= Secure Pi
Secure Pi is an OPEN SOURCE project!
We always welcome your CONTRIBUTIONS!

https://github.com/kkamagui/shadow-box-for-arm
Background
Design
Implementation

Demo. and Conclusion
(with Black Hat Sound Bytes)
Rootkits need to patch kernel code and read-only data
- They usually hide themselves by patching kernel code or function pointers
- But, kernel has page protection mechanism
- In x86 case, they disable page write protection in the CR3 register!
- In ARM case, they also need to disable page protection, too!
- Do we really need to know about the page protection mechanism for patching kernel?
  - Paging mechanism is too much complicated
  - ARM processors have various paging mechanism

- Use **live kernel patch functions** instead!
  - Linux kernel has kernel patch functions for a live patch
    - x86: `text_poke(void *addr, const void *opcode, size_t len)`
    - ARM: `patch_text(void *addr, unsigned int insn)`
  - You do not worry about the paging mechanism anymore!
- Do we really need to know about the page protection mechanism for patching kernel?
  - Paging mechanism is too much complicated
  - ARM processors have various paging mechanisms

- Use **live kernel patch functions** instead!
  - Linux kernel has kernel patch functions for a live patch
    - x86: `text_poke(void *addr, const void *opcode, size_t len)`
    - ARM: `patch_text(void *addr, unsigned int insn)`
  - You do not worry about the paging mechanism anymore!
- Kernel-level (ring 0) threats should be protected in a more privileged level (ring -1)
  - Rootkits can neutralize kernel-level (ring 0) protection
  - We create a ring -1 level protection mechanism with ARM TrustZone

- Shadow-box v2 is practical and portable
  - Shadow-box v2 protects the kernel from rootkits using IMA and OP-TEE
  - We made a reference implementation with Raspberry Pi 3
  - We named it “Secure Pi” and opened as an open source project
Questions?

Project: https://github.com/kkamagui/shadow-box-for-arm
Contact: hanseunghun@nsr.re.kr, @kkamagui1
        parkparkqw@nsr.re.kr, @DavePark312

CONTRIBUTION!