# SKEE: A Lightweight Secure Kernel-level Execution Environment for ARM

APR 11TH, 2016

[论文下载]

## Introduction

为内核提供检测与保护

以前的一些方法：

- Virtualization-based Approaches：TCB 必须足够大，能够处理资源分配、硬件虚拟化。而且虚拟机例如VMware、XEN有自身的安全问题。
- Microhypervisor Approaches：小型的hypervisor，旨在提供隔离。microhypervisors do not provide a good fit to host security tools that require a relatively large code base.
- sandbox：需要虚拟化提供隔离，需要hypervisor来管理分配sandbox，因此会增大TCB的大小。
- hardware protection：Intel: Software Guard Extensions (SGX).在ARM中没有类似的防护，ARM提出

TrustZone，但也会增大code base和需要对高权限的Trustzone层的维护

**SKEE**

- 完美地与内核隔离
- 提供一个切换到隔离区域的接口
- 能够让安全工具检测内核

# Background

## 32-bit ARMv7

内存转换涉及的三个寄存器

- Translation Table Base Control Register (TTBCR)
- Translation Table Base Register 0 (TTBR0)
- Translation Table Base Register 1 (TTBR1)

TTBCR contains a 3 bit called TTBCR.N

If the value of TTBCR.N is 0, then TTBR1 is not used, otherwise both TTBR0 and TTBR1 are used.

Table I. EFFECT OF TTBCR.N ON ADDRESS TRANSLATION ON 32-BIT ARMv7 USING SHORT DESCRIPTOR FORMAT

| TTBCR.N value | Starting address of TTBR1 translation |
|---|---|
| 0b000 | TTBR1 not used |
| 0b001 | 0x8000_0000 |
| 0b010 | 0x4000_0000 |
| 0b011 | 0x2000_0000 |
| 0b100 | 0x1000_0000 |
| 0b101 | 0x0800_0000 |
| 0b110 | 0x0400_0000 |
| 0b111 | 0x0200_0000 |

## 64-bit ARMv8

- kernel: TTBR1
- user: TTBR0

MSR指令可以使用0寄存器（XZR)使任何寄存器为0.

Address Space Identifier (ASID)：非全局访问的虚拟内存页，与一个特定的ASID相连。
Memory Management in Virtualization Layer

# THREAT MODEL, SECURITY GUARANTEES AND ASSUMPTIONS

## Threat model

- 修改、重定位内核执行文件：安全操作
- 通过漏洞修改内核数据或控制流：通过安全工具检测
- 攻击SKEE隔离区域，使防护失效：保证攻击不能到达隔离区域，bypass检测

## SECURITY GUARANTEES

- 禁止内核修改内存布局和系统访问权限
- 内核到隔离区域只能通过一个严格控制的switch gate来访问

## ASSUMPTIONS

- 整个系统 loaded securely
- 内核支持 W⊕X

# SKEE DESIGN

## SKEE Isolation

### Creating a Protected Address Space

1. 去除所有映射到SKEE隔离区域或者内核memory translation tables的entries
2. 将内核代码和SKEE switch gate映射为只读
3. 所有其他内存区域（包含内核数据，用户层内存）为PXN

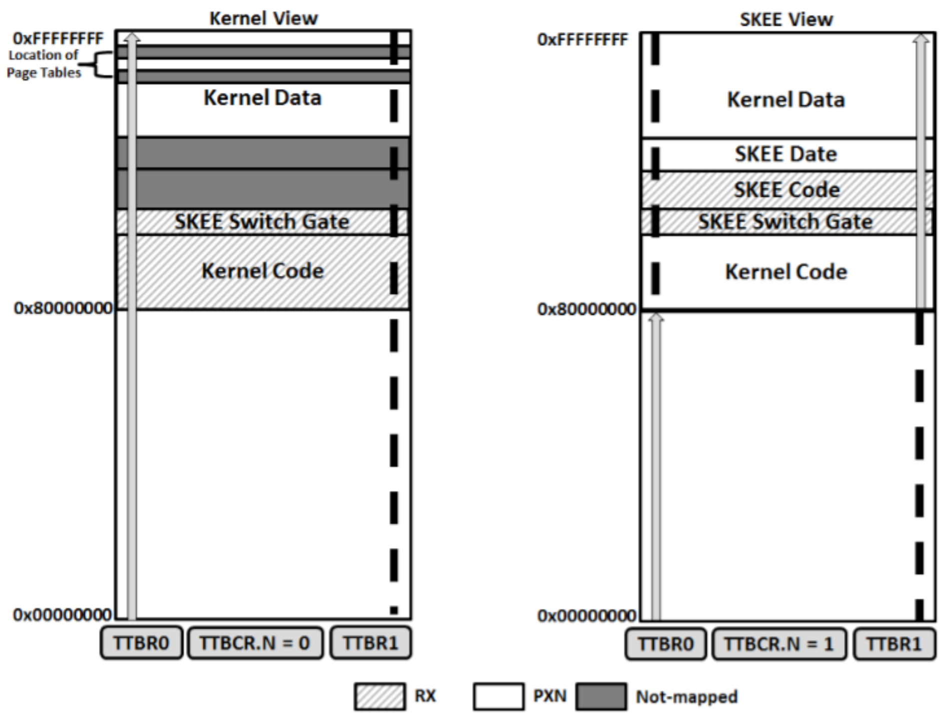


Figure 3. An example of address Space Separation on ARMv7. The vertical arrows show the virtual memory range translated via the corresponding translation table base register. The dashed lines show the memory range that is not addressable via the corresponding translation table base register.

### Restrict Kernel Access to the MMU

SKEE 移除了关于内核的一些控制代码，把它们替换成可以跳到switch gate的hooks

## SKEE Secure Context Switching

switch gate 是 atomic, deterministic and exclusive

- ARMv7

```
/* Start of the SKEE Entry Gate */
mrs     r0, cpsr                // Read the status register
push    {r0}                    // Save the status register value
orr     r0, r0, #0x1c0          // Set the mask interrupts bits
msr     cpsr, r0                // load the modified value

mov     r0, #0x11
isb                             // Syncronization barrier
mcr     p15, 0, r0, c2, c0, 2   // Modify the TTBCR to activate SKEE
isb

mcr     p15, 0, r0, c8, c7, 0   // TLB invalidate
isb

bl      skee_entry              // Jump to SKEE entry point
/* End of the SKEE Entry Gate */

/* Start of the SKEE Exit Gate */
mov     r0, #0
isb
mcr     p15, 0, r0, c2, c0, 2   // Modify the TTBCR to deactivate SKEE
isb

mcr     p15, 0, r0, c8, c7, 0   // TLB invalidate
isb

pop     {r0}                    // Reload status register value
msr     cpsr, r0                // Restore the original status register

bl      kernel_entry            // Jump back to the kernel
/* End of the SKEE Exit Gate */
```

Figure 4. SKEE Switch Gate on ARMv7

- ARMv8

```
/* Start of the SKEE Entry Gate */
mrs     x0, DAIF               // Read interrupt mask bits
str     x0, [sp, #-8]!         // Save interrupt mask bits
msr     DAIFset, 0x3           // Mask all interrupts

mrs     x0, ttbr1_el1          // Read existing TTBR1 value
str     x0, [sp, #-8]!         // Save existing TTBR1 value

msr ttbr1_el1, xzr             // Load the value Zero to TTBR1
isb

tlbi vmalle1                   // Invalidate the TLB
isb

adr x0, skee_entry             // Jump to SKEE entry point
br x0
/* End of the SKEE Entry Gate */
```

Figure 5. SKEE Entry Gate on ARMv8

```
/* Start of the SKEE Exit Gate */
nop                          //no operation
nop                          // Fill the page with no operations to
nop                          // align the last instruction with the
nop                          // bottom of the isolated page boundry

msr    DAIFset, 0x3          // Mask all interrupts

ldr x0, [sp, #8]!            // Reload kernel TTBR1 value
dsb sy
msr ttbr1_el1, x0            // Restore TTBR1 to kernel value

/*------------Isolated Page Boundry---------------*/

isb
tlbi vmalle1                 // Invalidate the TLB
isb


ldr x0, [sp, #8]!            // Reload interrupts mask bits
msr DAIF, x0                 // Restore interrupts mask bits register

ret
/* End of the SKEE Exit Gate */
```

Figure 6. SKEE Exit Gate on ARMv8

- Using ASID for Faster Context Switching

```
/* Start of the SKEE Entry Gate */
push    {lr}                        // save the return address
mrs     r0, cpsr                    // read the status register
push    {r0}                        // save the status register value
orr     r0, r0, #0x1c0              // set the mask interrupts bits
msr     cpsr, r0                    // load the modified value

mrc     p15, 0, r0, c13, c0, 1  // read current CONTEXTIDR
push    {r0}                        // save the CONTEXTIDR value
mov     r0, #0
mcr     p15, 0, r0, c13, c0, 1  // set CONTEXTIDR to 0
isb

mov     r0, #0x11
isb                                 // syncronization barrier
mcr     p15, 0, r0, c2, c0, 2   // modify the TTBCR to activate SKEE
isb

L1:
mrc     p15, 0, r0, c13, c0, 1  // read current CONTEXTIDR
cmp     r0, #0                      // compare current CONTEXTIDR with 0
bne     L1                          // loop back if CONTEXTIDR is not 0

bl      skee_entry                  // jump to SKEE entry point
/* End of the SKEE Entry Gate */

/* Start of the SKEE Exit Gate */
mov     r0, #0
isb
mcr     p15, 0, r0, c2, c0, 2   // modify the TTBCR to deactivate SKEE
isb

L2:
mrc     p15, 0, r0, c2, c0, 2   // read current TTBCR
cmp     r0, #0                      // compare current TTBCR with 0
bne     L2                          // loop back if TTBCR is not 0

pop     {r0}                        // reload CONTEXTIDR value
mcr     p15, 0, r0, c13, c0, 1  // write original CONTEXTIDR
isb

L3:
mrc     p15, 0, r0, c13, c0, 1  // read current CONTEXTIDR
cmp     r0, #0                      // compare current CONTEXTIDR with 0
beq     L3                          // loop back if CONTEXTIDR is 0

pop     {r0}                        // reload status register value
msr     cpsr, r0                    // restore the original status register

pop     {lr}                        // reload the return address
movs    pc, lr                      // jump back to the kernel
/* End of the SKEE Exit Gate */
```

Figure 7.   A Faster SKEE Switch Gate on ARMv7

## Kernel Monitoring and Protection

- 可以 trap kernel critical events
- 可以访问内核内存
- 可以进行内核内存保护

## Security Analysis

- 32-bit ARMv7: Samsung Note4
- 64-bit ARMv8: Samsung Galaxy S6、Samsung Galaxy Note5

# IMPLEMENTATION AND evaluation

Table III. SKEE BENCHMARK SCORES ON ARMV7

| Benchmark | Original | SKEE | Degradation (%) |
|---|---|---|---|
| CF-Bench | 30933 | 29035 | 6.14% |
| Smartbench 2012 | 5061 | 5002 | 1.17% |
| Linpack | 718 | 739 | -2.93% |
| Quadrant | 12893 | 12552 | 2.65% |
| Antutu v5.7 | 35576 | 34761 | 2.29% |
| Vellamo | | | |
| Browser | 2465 | 2500 | -1.42% |
| Metal | 1077 | 1071 | 0.56% |
| Geekbench | | | |
| Single Core | 1083 | 966 | 10.8% |
| Multi Core | 3281 | 2747 | 16.28% |

Table IV. SKEE BENCHMARK SCORES ON ARMV8

| Benchmark | Original | SKEE | Degradation(%) |
|---|---|---|---|
| CF-Bench | 75641 | 66741 | 11.77% |
| Smartbench 2012 | 14030 | 13377 | 4.65% |
| Linpack | 1904 | 1874 | 1.58% |
| Quadrant | 36891 | 35595 | 3.51% |
| Antutu v5.7 | 66193 | 67223 | -1.56% |
| Vellamo | | | |
| Browser | 3690 | 3141 | 14.88% |
| Metal | 2650 | 2540 | 4.15% |
| Geekbench | | | |
| Single Core | 1453 | 1235 | 15.00% |
| Multi Core | 4585 | 4288 | 6.48% |

Table V. SKEE App Load Delay on ARMv7

| App | Original | SKEE | Overhead (%) |
|---|---|---|---|
| Temple Run 2 | 9.31 | 10.33 | 10.96% |
| Hill Climb Racing | 3.66 | 3.71 | 1.37% |
| Angry Birds | 4.72 | 4.79 | 1.48% |
| Crossy Road | 4.81 | 5.24 | 8.94% |
| Subway Surf | 5.45 | 5.95 | 9.17% |

Table VI. SKEE App Load Delay on ARMv8

| App | Original | SKEE | Overhead(%) |
|---|---|---|---|
| Temple Run 2 | 6.08 | 6.58 | 8.22% |
| Hill Climb Racing | 2.42 | 2.73 | 12.81% |
| Angry Birds | 4.12 | 4.32 | 4.85% |
| Crossy Road | 3.42 | 3.80 | 11.11% |
| Subway Surf | 4.42 | 4.71 | 6.34% |

Table VII. SKEE Added Security Checks Benchmark Scores

| Benchmark | Original | SKEE | Degrad. (%) | SKEE + Sec. Checks | Degrad. (%) |
|---|---|---|---|---|---|
| CF-Bench | 63273 | 58903 | 6.91% | 57250 | 2.81% |
| Smartbench | 15820 | 15217 | 3.81% | 15104 | 0.74% |
| Linpack | 1849 | 1697 | 8.22% | 1560 | 8.07% |
| Quadrant | 31429 | 30843 | 1.86% | 29330 | 4.91% |
| Antutu v5.7 | 65242 | 62866 | 3.64% | 58658 | 6.69% |
| Vellamo | | | | | |
| Browser | 4659 | 4256 | 8.65% | 4350 | -2.21% |
| Metal | 2158 | 2139 | 0.88% | 2081 | 2.71% |
| Geekbench | | | | | |
| Single Core | 1508 | 1342 | 11.00% | 1340 | 0.15% |
| Multi Core | 4566 | 4388 | 3.90% | 4207 | 4.12% |