

Security World Vulnerability Discovery Across Diverse Chip Architectures

About US

Juntao Wu(@wugedbl)

aka. Dawuge. Founder of Shuffle Team(@ShuffleTeam_AI). Security Researcher focused on OS, Apps, and Web3. Blackhat ('22), Zer0Con ('24) , OffbyOne ('24) and KanxueSDC ('25) Speaker. Received multiple acknowledgments from major OS vendors. Samsung Hall of Fame Top 10 (4x).

Hangyu Hua(@HBh25Y)

Co-founder of Shuffle Team(@ShuffleTeam_AI). Mainly focus on low-level vulnerability discovery and exploitation, covering the boot chain, the kernel, the Trusted Execution Environment, and related components.

Agenda

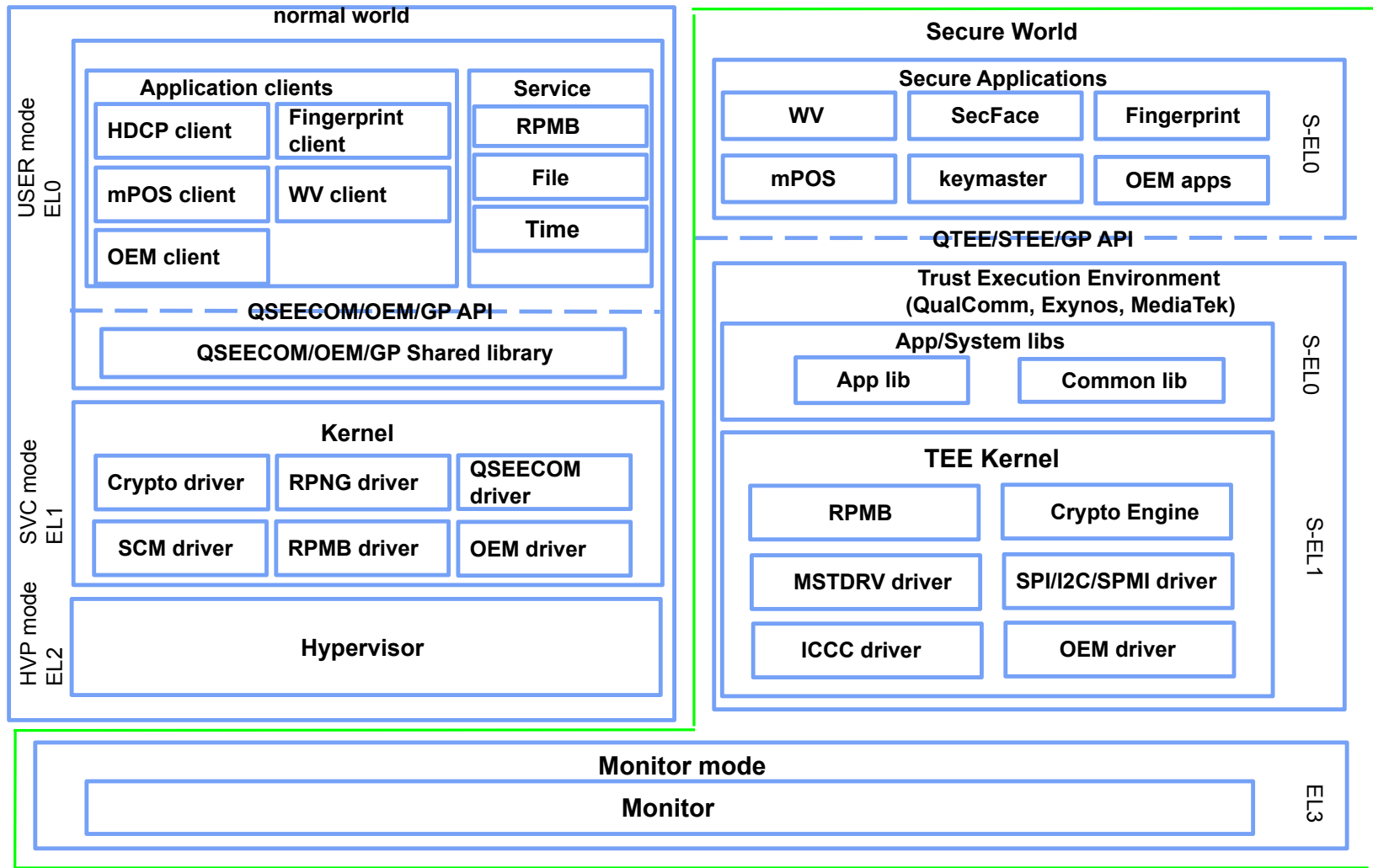
- Background
- Architecture Analysis
- Common modules
 - Common vulnerabilities
 - Unique vulnerabilities
- Special modules
 - Blockchain Wallet
 - Knox Vault
- Summary
- Q&A

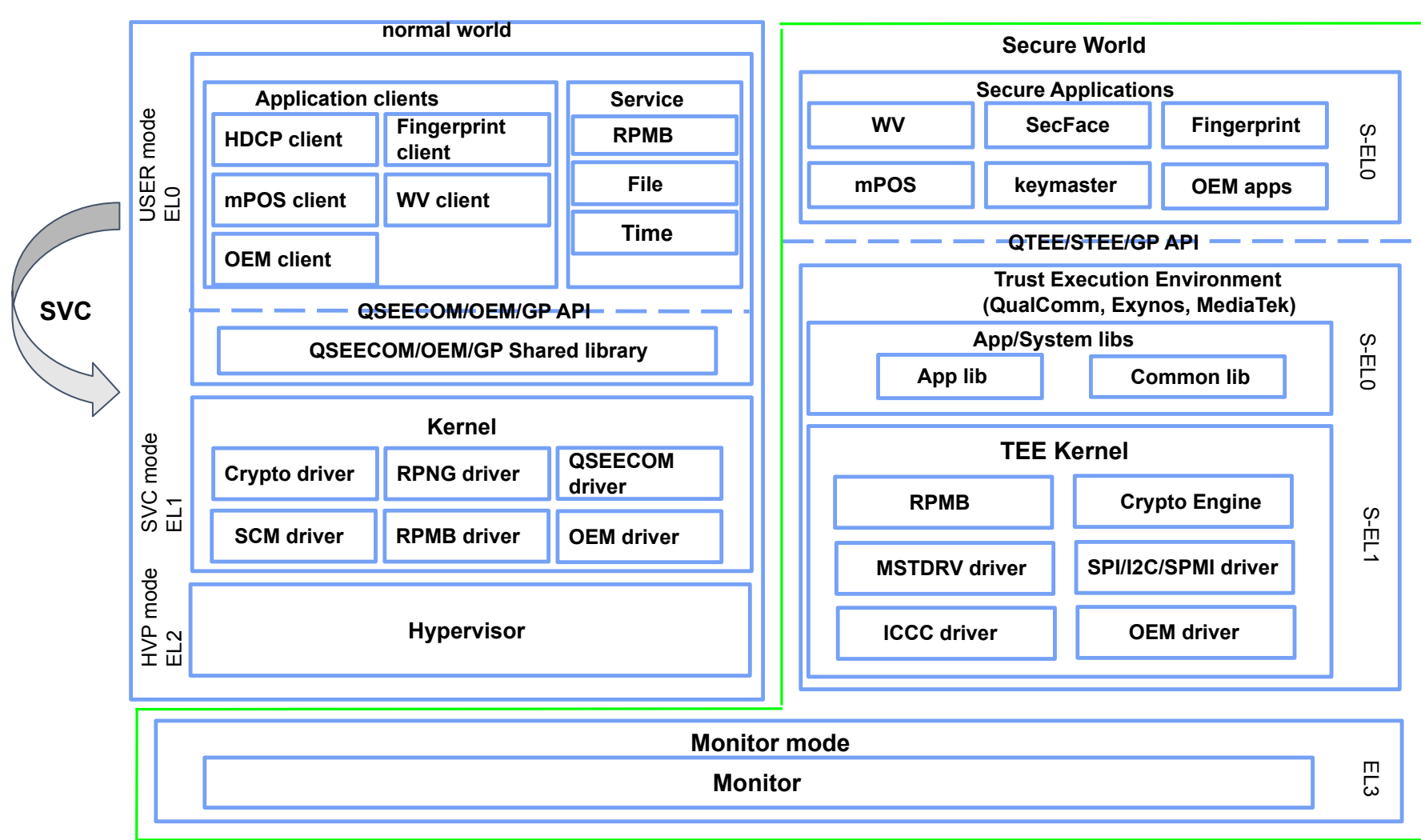
Background

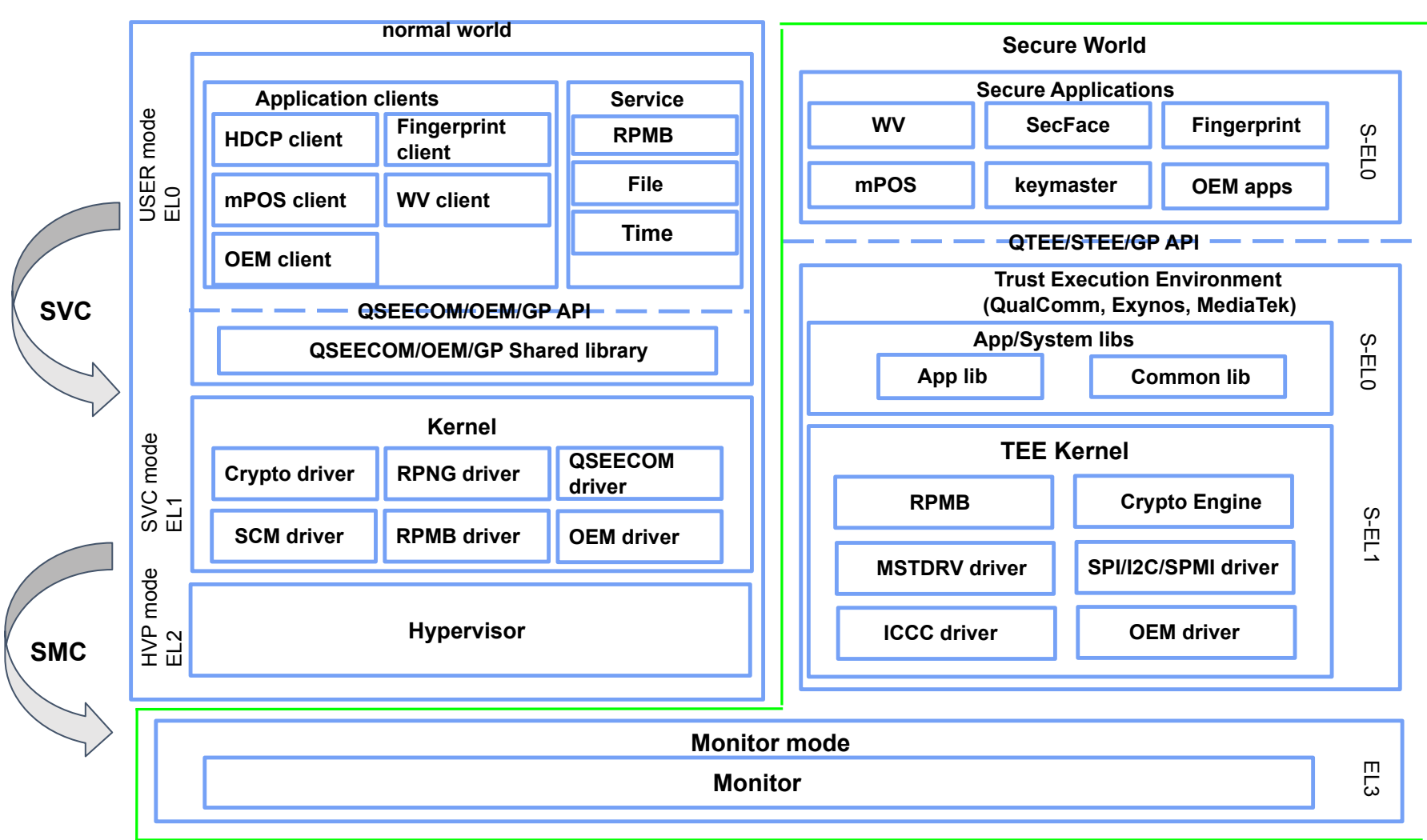
Current Status of TEE

1. The applications of TEE (Trusted Execution Environment) are rapidly expanding beyond traditional security domains into emerging computing fields that demand stringent data privacy and compliance, particularly in **AI/ML** and **Web3/blockchain**.
2. Intel SGX/TDX, AMD SEV-SNP, Apple PCC, ATF, GPU TEE?
3. Attack Surface and Challenges
 - a. Side Channels
 - b. Supply Chain
 - c. Cross-Domain / Heterogeneous Computing
 - d. **Application and Architectural Implementation**
 - e. ...

Architecture Analysis







```

/**
 * __arm_smccc_smc() - make SMC calls
 * @a0-a7: arguments passed in registers 0 to 7
 * @res: result values from registers 0 to 3
 * @quirk: points to an arm_smccc_quirk, or NULL when no quirks are required.
 *
 * This function is used to make SMC calls following SMC Calling Convention.
 * The content of the supplied param are copied to registers 0 to 7 prior
 * to the SMC instruction. The return values are updated with the content
 * from register 0 to 3 on return from the SMC instruction. An optional
 * quirk structure provides vendor specific behavior.
 */
asmlinkage void __arm_smccc_smc(unsigned long a0, unsigned long a1,
                                unsigned long a2, unsigned long a3, unsigned long a4,
                                unsigned long a5, unsigned long a6, unsigned long a7,
                                struct arm_smccc_res *res, struct arm_smccc_quirk *quirk);

/**
 * __arm_smccc_hvc() - make HVC calls
 * @a0-a7: arguments passed in registers 0 to 7
 * @res: result values from registers 0 to 3
 * @quirk: points to an arm_smccc_quirk, or NULL when no quirks are required.
 *
 * This function is used to make HVC calls following SMC Calling
 * Convention. The content of the supplied param are copied to registers 0
 * to 7 prior to the HVC instruction. The return values are updated with
 * the content from register 0 to 3 on return from the HVC instruction. An
 * optional quirk structure provides vendor specific behavior.
 */
asmlinkage void __arm_smccc_hvc(unsigned long a0, unsigned long a1,
                                unsigned long a2, unsigned long a3, unsigned long a4,
                                unsigned long a5, unsigned long a6, unsigned long a7,
                                struct arm_smccc_res *res, struct arm_smccc_quirk *quirk);

```

```

.macro SMCCC instr
\instr #0
ldr x4, [sp]
stp x0, x1, [x4, #ARM_SMCCC_RES_X0_OFFSETS]
stp x2, x3, [x4, #ARM_SMCCC_RES_X2_OFFSETS]
ldr x4, [sp, #8]
cbz x4, 1f /* no quirk structure */
ldr x9, [x4, #ARM_SMCCC_QUIRK_ID_OFFSETS]
cmp x9, #ARM_SMCCC_QUIRK_QCOM_A6
b.ne 1f
str x6, [x4, ARM_SMCCC_QUIRK_STATE_OFFSETS]
1: ret
.endm

/*
 * void arm_smccc_smc(unsigned long a0, unsigned long a1, unsigned long a2,
 *                    unsigned long a3, unsigned long a4, unsigned long a5,
 *                    unsigned long a6, unsigned long a7, struct arm_smccc_res *res,
 *                    struct arm_smccc_quirk *quirk)
 */
SYM_FUNC_START(__arm_smccc_smc)
    SMCCC    smc
SYM_FUNC_END(__arm_smccc_smc)
EXPORT_SYMBOL(__arm_smccc_smc)

/*
 * void arm_smccc_hvc(unsigned long a0, unsigned long a1, unsigned long a2,
 *                    unsigned long a3, unsigned long a4, unsigned long a5,
 *                    unsigned long a6, unsigned long a7, struct arm_smccc_res *res,
 *                    struct arm_smccc_quirk *quirk)
 */
SYM_FUNC_START(__arm_smccc_hvc)
    SMCCC    hvc
SYM_FUNC_END(__arm_smccc_hvc)
EXPORT_SYMBOL(__arm_smccc_hvc)

```

ESR_EL3, Exception Syndrome Register (EL3)

The ESR_EL3 characteristics are:

Purpose

Holds syndrome information for an exception taken to EL3.

Configuration

This register is present only when EL3 is implemented and FEAT_AA64 is implemented. Otherwise, direct accesses to ESR_EL3 are UNDEFINED.

0b010011	SMC instruction execution in AArch32 state, when SMC is not disabled.	ISS encoding for an exception from SMC instruction execution in AArch32 state	ISS2 encoding for all other exceptions	When FEAT_AA32 is implemented
0b010111	SMC instruction execution in AArch64 state, when SMC is not disabled.	ISS encoding for an exception from SMC instruction execution in AArch64 state	ISS2 encoding for all other exceptions	When FEAT_AA64 is implemented

```
__int64 smc_handler_32_or_64()
{
...
if ( v0 == 23 || v0 == 19 )
{
...
monitor_smc_handler((struct smc_param
*)&v29, v0);
...
__WriteStatusReg(ARM64_SYSREG(3, 6, 1,
1, 0), *(_QWORD *)&dword_1C01D058[v16 +
v17]);
__dsb(0xFu);
__isb(0xFu);
__asm { ERET }
...
}
...
}
```

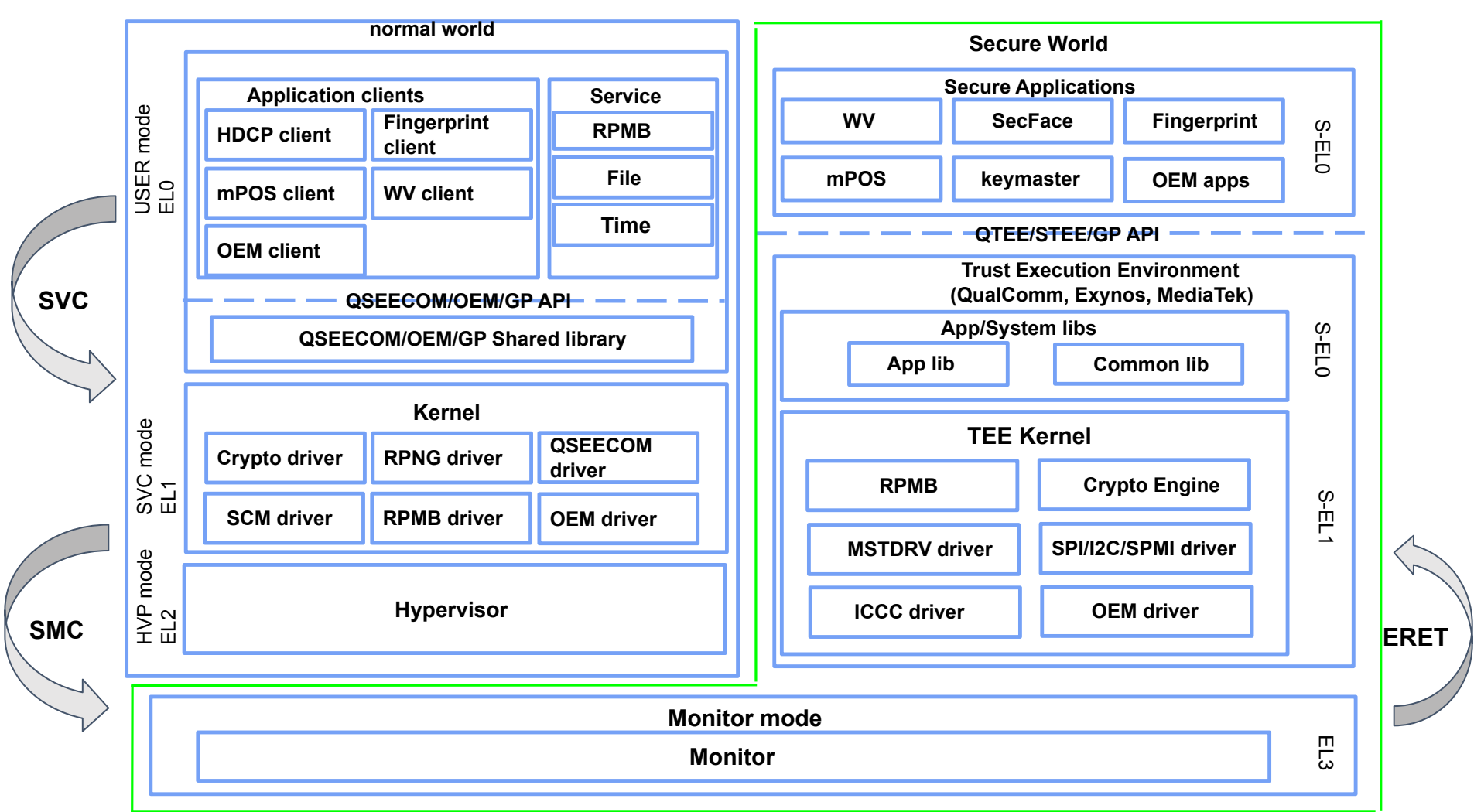
```

__int64 smc_handler_32_or_64()
{
...
if ( v0 == 23 || v0 == 19 )
{
...
    monitor_smc_handler((struct smc_param
*)&v29, v9);
...
    _WriteStatusReg(ARM64_SYSREG(3, 6,
1, 1, 0), *(_QWORD *)&dword_1C01D058[v16
+ v17]);
    __dsb(0xFu);
    __isb(0xFu);
    __asm { ERET }
...
}
...
}

```

- **monitor_smc_handler** is used to perform preliminary processing on the data we pass in.

- The **ERET** instruction is used to terminate the processing of the monitor.



SCR_EL3, Secure Configuration Register

The SCR_EL3 characteristics are:

Purpose

Defines the configuration of the current Security state. It specifies:

- The Security state of EL0, EL1, and EL2. The Security state is either Secure or Non-secure.
- The Execution state at lower Exception levels.
- Whether IRQ, FIQ, SError interrupts, and External abort exceptions are taken to EL3.
- Whether various operations are trapped to EL3.

Configuration

AArch64 System register SCR_EL3 bits [31:0] can be mapped to AArch32 System register SCR[31:0], but this is not architecturally mandated.

This register is present only when EL3 is implemented. Otherwise, direct accesses to SCR_EL3 are UNDEFINED.

MSR **SCR_EL3**, X19
LDP X19, X20,
[SP-0x150+arg_150],#0x10
DSB SY
ISB
ERET

NS, bit [0]

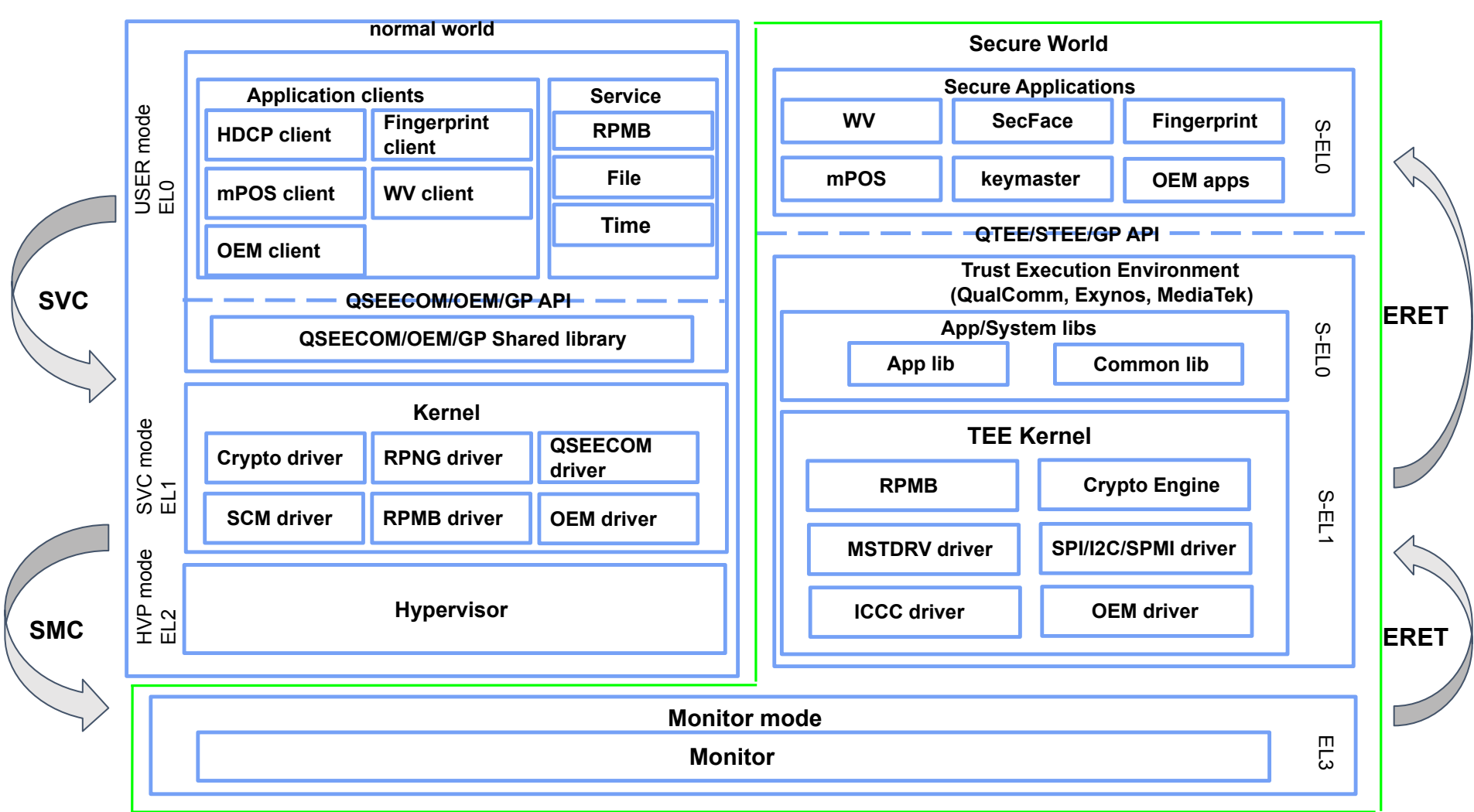
Non-secure bit.

NS	Meaning
0b0	Indicates that EL0 and EL1 are in Secure state.
0b1	Indicates that Exception levels lower than EL3 are in Non-secure state, so memory accesses from those Exception levels cannot access Secure memory.

When SCR_EL3.{EEL2, NS} == {1, 0}, then EL2 is using AArch64 and in Secure state.

The reset behaviour of this field is:

- On a Warm reset, this field resets to an architecturally UNKNOWN value.

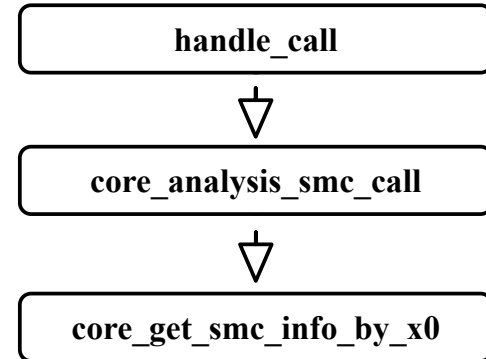


```

_void __usercall std_svc_smc_handler(
    __int64 a1@<X0>,
    __int64 a2@<X1>,...)
{
    ...
    if ( !sub_1C038E58(a7, a8) )
    {
        if ( sub_1C038C00(0LL, v9, v10, v11, v12, v13,
v14, v15, v22.x0) )
        {
            v16 = handle_call(&v22);
            v17 = sub_1C151BDC(v16);
            sub_1C038CBC(v17 - 112);
            __asm { SMC          #0 }
        }
        while ( 1 )
        ;
    }
    off_1C151C58();
}

```

- Invoke an SMC call to return to the monitor.
- Internally calls ERET to return to the TA (Trusted Application).

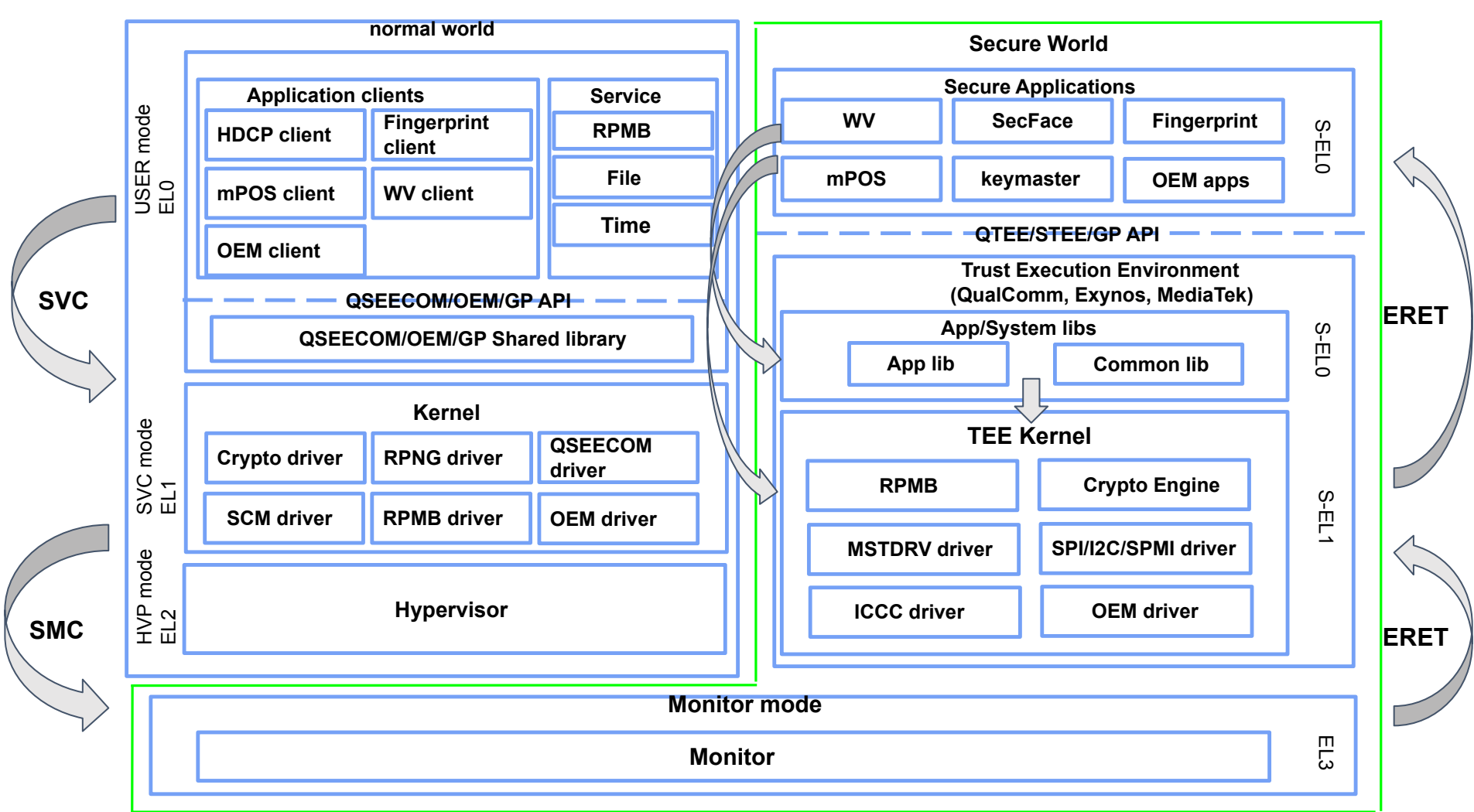


```
#define ARM_SMCCC_CALL_VAL(type, calling_convention, owner, func_num) \
(((type) << ARM_SMCCC_TYPE_SHIFT) | \
((calling_convention) << ARM_SMCCC_CALL_CONV_SHIFT) | \
((owner) & ARM_SMCCC_OWNER_MASK) << ARM_SMCCC_OWNER_SHIFT) | \
((func_num) & ARM_SMCCC_FUNC_MASK))
```

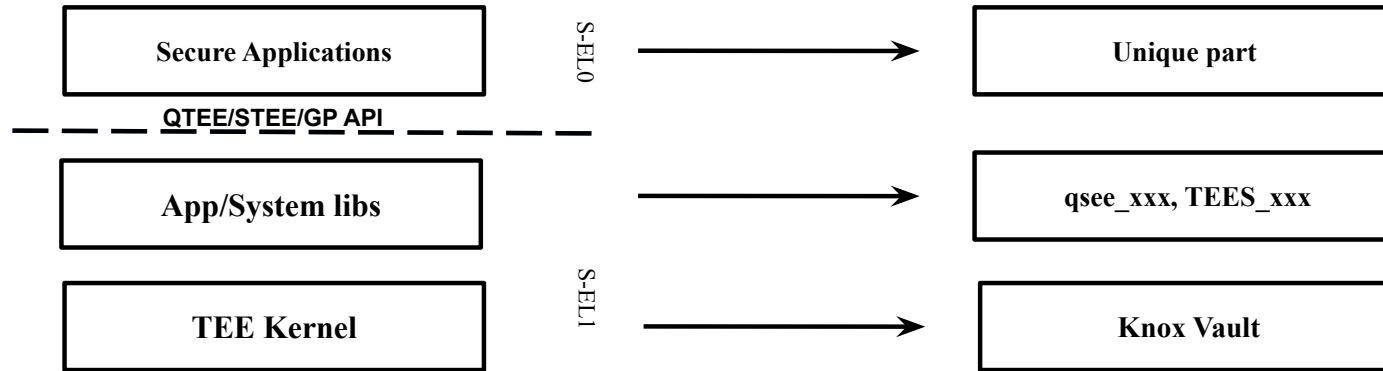
```
int qcom_scm_ice_invalidate_key(u32 index)
{
    struct qcom_scm_desc desc = {
        .svc = QCOM_SCM_SVC_ES,
        .cmd = QCOM_SCM_ES_INVALIDATE_ICE_KEY,
        .arginfo = QCOM_SCM_ARGS(1),
        .args[0] = index,
        .owner = ARM_SMCCC_OWNER_SIP,
    };

    return qcom_scm_call(__scm->dev, &desc, NULL);
}
```

```
__int64 __fastcall core_get_smc_info_by_x0(__int64 x0,
struct smc_call_info *a2)
{
    switch ( x0 )
    {
        ...
        case 0x2001003LL:
            __asm { BTI                }; jumtable
            000000001C0A5EDC case 1 }
            call_type = 0;
            id = 0x2001003;
            func = (__int64
            (*)(void))scm_es_ice_invalidate_key;
            goto LABEL_263;
        ...
    }
    ...
}
```



Architectural differences



- In the TA, functions such as qsee_hash are called to perform operations like hash creation.

```
result = qsee_hash(3LL, a1, a2, a3,
32LL);
if ( (_DWORD)result )
{
    PAL_Log(4LL, "PAL_Hash_SHA256 fail
- qsee_hash error : %d", result);
    result = 4294965281LL;
}
```

- Upon deeper inspection, it can be found that the final call is made to a function pointer returned by qsee_open.

```
if ( !a1 )
    return 1LL;
v4 = *(unsigned int (__fastcall **)(__int64,
_QWORD, _QWORD *, __int64))a1;
v3 = *(_QWORD *)(a1 + 8);
v6[0] = a2;
v6[1] = a3;
if ( v4(v3, 0LL, v6, 1LL) )
    return 0xFFFFFFFFLL;
else
    return 0LL;
```

- These general-purpose functionalities are implemented as separate modules, with each module's final call corresponding to its own invoke function.

```
__int64 Hash_module_open ((__int64 a1, unsigned
__int16 a2, __int64 a3, int a4)
{
...
    if ( !(_DWORD)result )
    {
        *a4 = Hash_module_invoke;
        a4[1] = v7;
        return result;
    }
}
...
}
```

```
__int64 Module_invoke ((__int64 a1, unsigned
__int16 a2, __int64 a3, int a4)
{
...
    v9 = **(_DWORD **)a3;
    if ( v9 <= 260 )
    {
        switch ( v9 )
        {
            ...
            case 38:
                __asm { BTI          j; jumtable
000000001C09BDAC case 38 }
                v16 = 38;
                return Hash_module_open(v16, (__int64)v7,
(__int64)v8, v6);
            ...
        }
    }
...
}
```

- All invoke functions have four parameters, where a2 represents the function ID corresponding to the operation being executed in the current invoke. For example, in Hash_module_invoke, a2 indicates which specific hash operation is being performed.

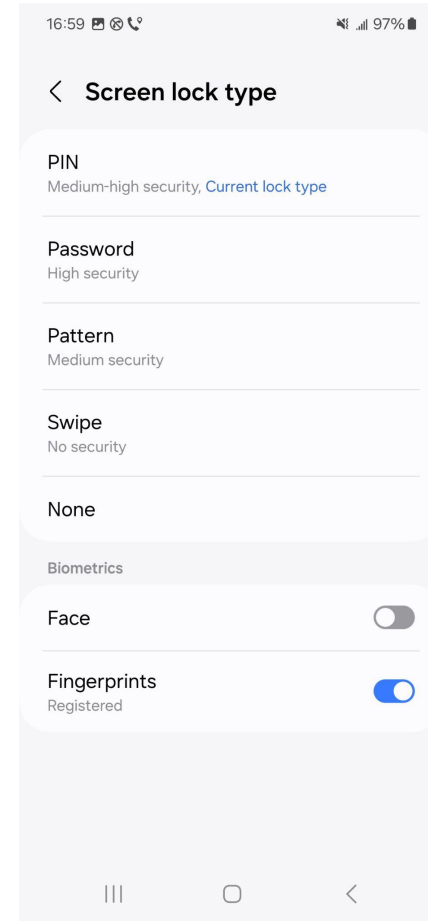
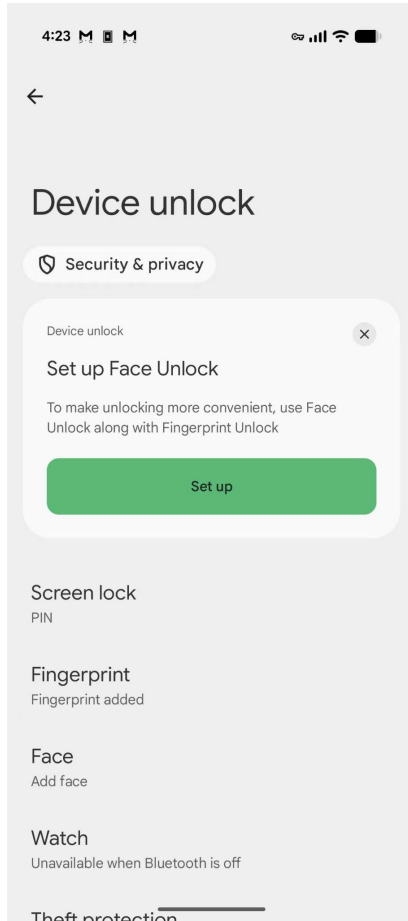
- open hash module
- create hash
- create hmac

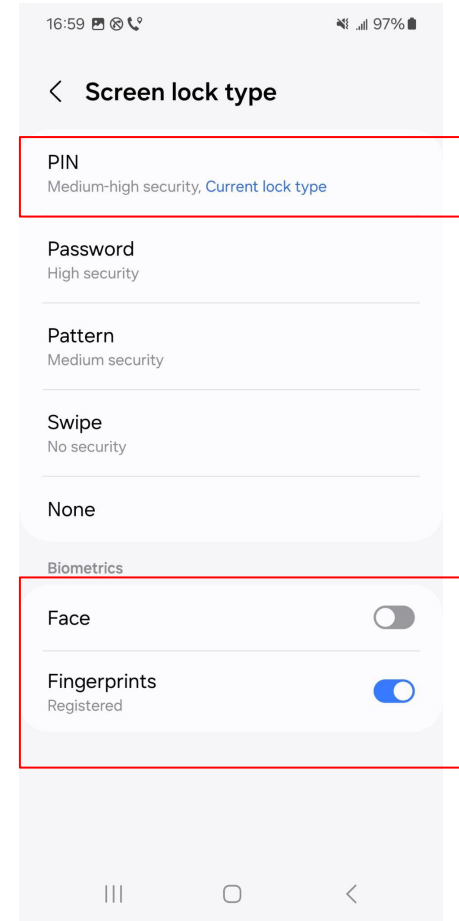
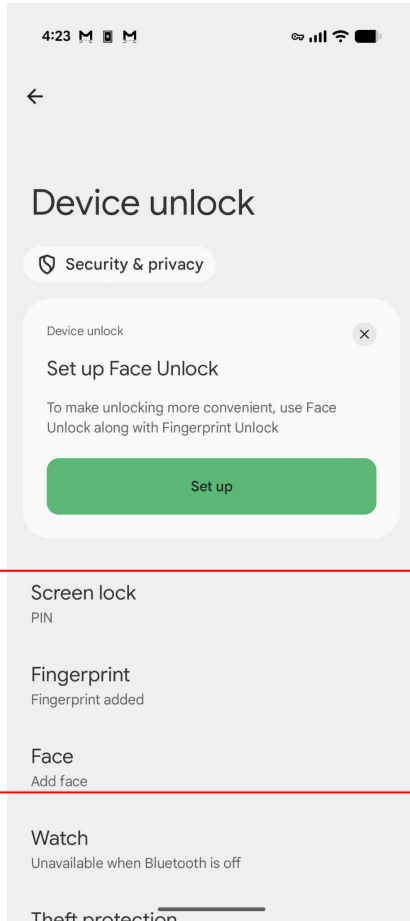
```
__int64 Hash_module_invoke ((__int64 a1, unsigned
__int16 a2, __int64 a3, int a4)
{
...
switch ( a2 )
{
case 0u:
...
case 1u:
...
case 2u:
...
}
...
}
```

Common modules

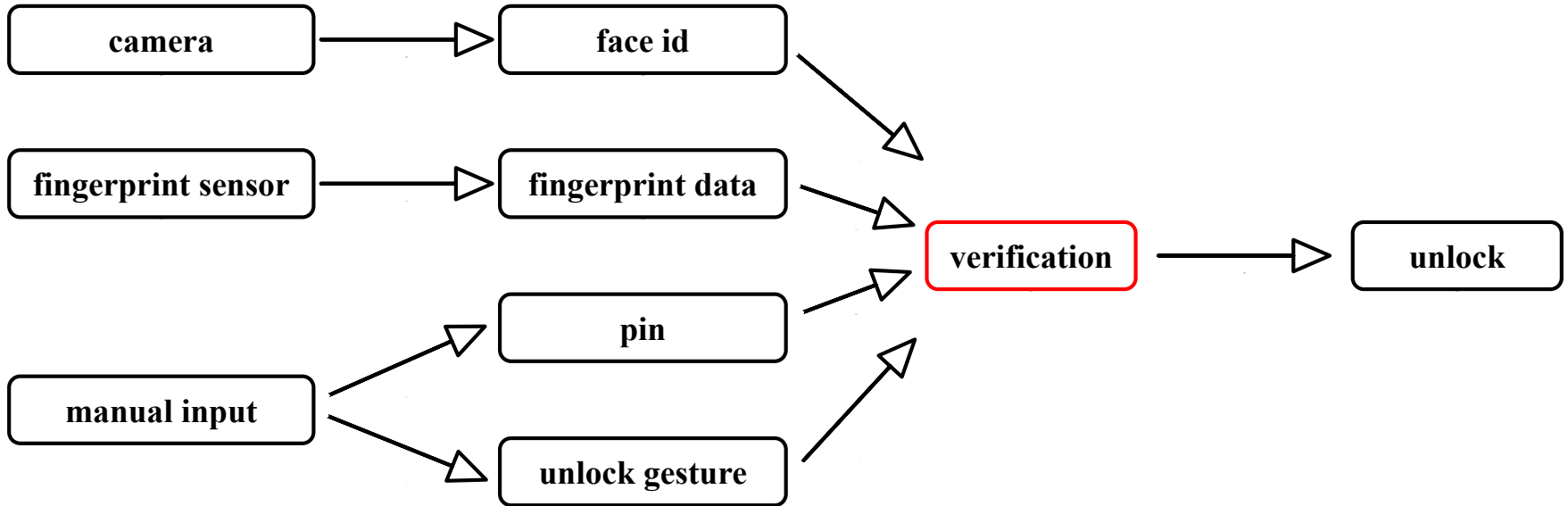
Common functions

- Biometrics
- Payment Security
- Device Unlock
- Key Management
- Digital Rights Management (DRM)
- Secure Boot Chain
- ...

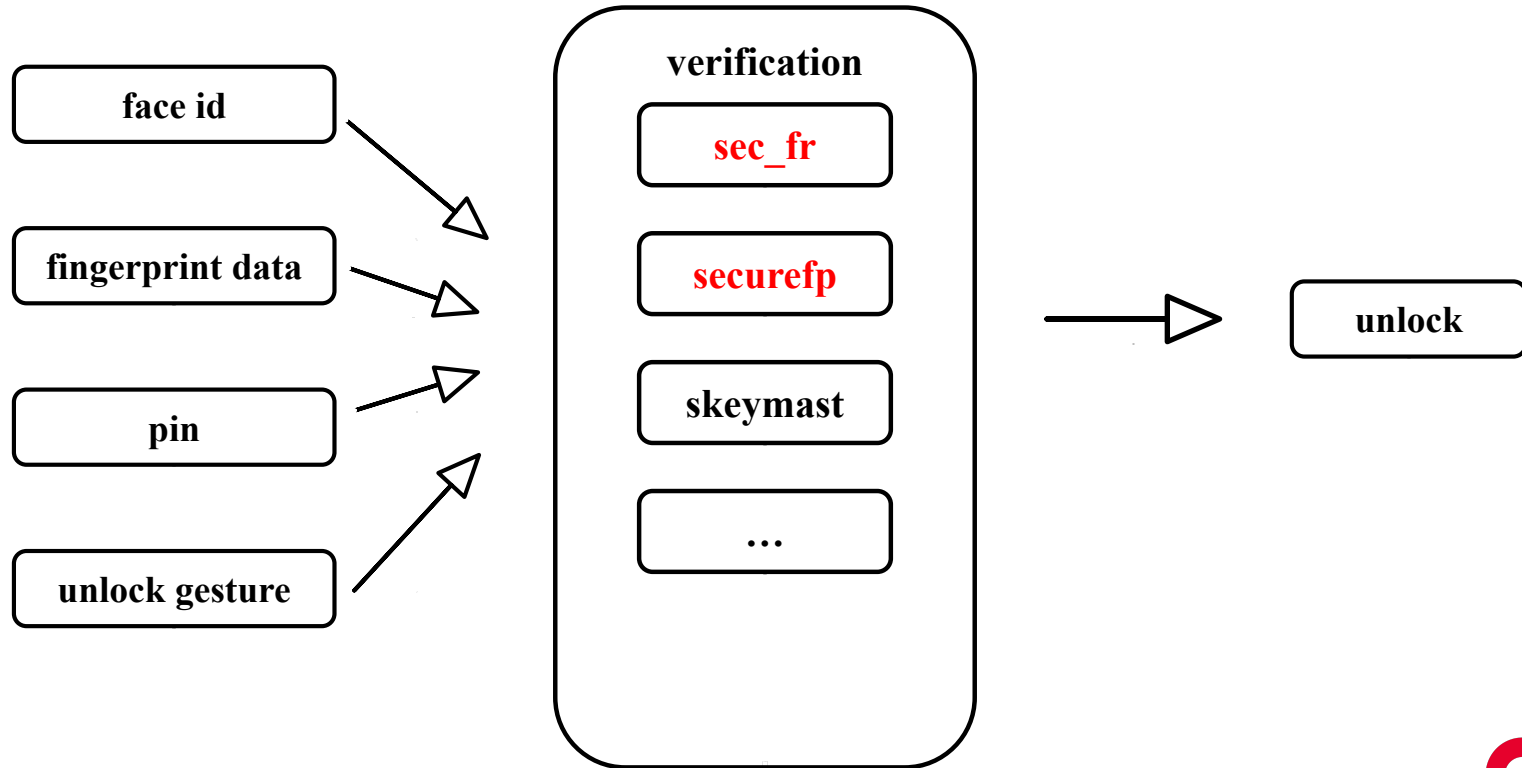




Device Unlock



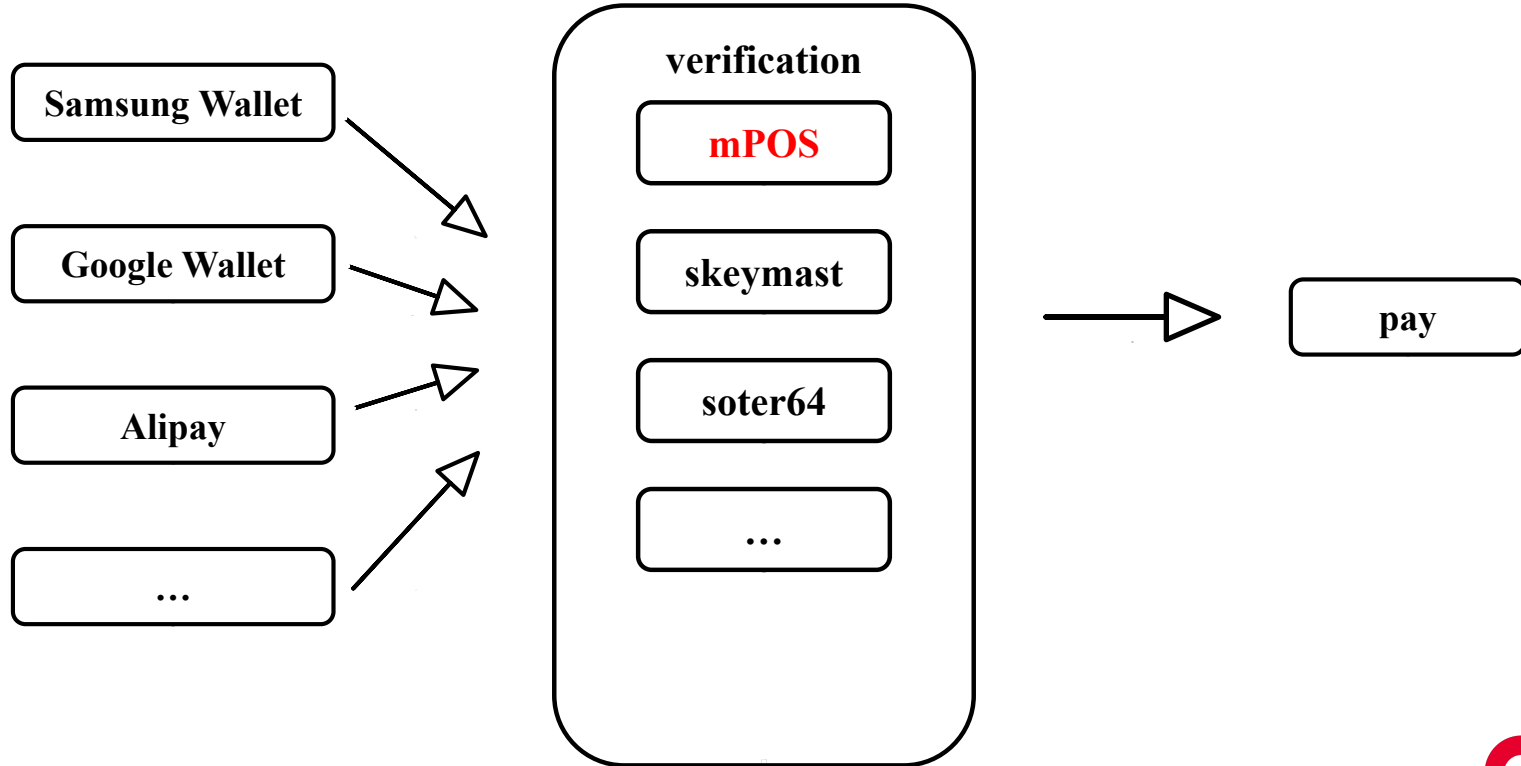
Device Unlock



Payment Security



Payment Security



Common modules



- sec_fr

The sec_fr module is a trusted execution module for facial recognition running in the TEE world. It processes facial recognition algorithms, template data, and authentication procedures within a secure environment, preventing the Rich Execution Environment (REE, e.g., Android user space) from stealing or tampering with biometric data.



- securefp

The Fingerprint module is a trusted application running in the TEE, designed to securely perform fingerprint enrollment, feature matching, and encrypted template storage.



- mPOS

mPOS is a trusted application that runs in a secure environment to execute mobile payment transactions, key management, and cryptographic operations, preventing payment information, keys, and transaction data from being stolen or tampered with by the Rich Execution Environment (e.g., Android application layer).

Common vulnerabilities



- `sec_fr`

The **sec_fr** module is a trusted execution module for facial recognition running in the TEE world. It processes facial recognition algorithms, template data, and authentication procedures within a secure environment, preventing the Rich Execution Environment (REE, e.g., Android user space) from stealing or tampering with biometric data.

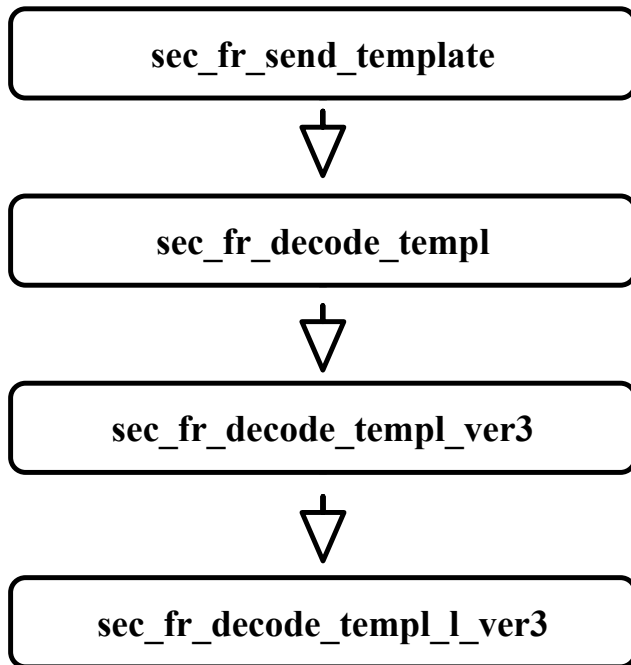
CVE-2025-20943

- The command with ID 0x6 is used to transmit a specific template, with the size of the input data limited to less than 0x8000C.

```
struct qsc_send_cmd_sendTemplate {
    unsigned int cmd_id;
    unsigned int version;
    unsigned int version_pad;
    unsigned int data_len;
<---- [1]
    char data[524274];
    char data1_end[2];
    unsigned int unknow_buf_len;
<---- [2]
    unsigned int data1;
    unsigned int data1_pad;
    unsigned long pad[6];
};
```

```
void *__fastcall sec_fr_cmd_handler(
    void *input_buf,
    unsigned int input_buf_len,
    void *output_buf,
    unsigned int output_buf_len)
{
    ...
    switch ( *(_DWORD *)input_buf )
    {
        ...
        case 6:
            if ( input_buf_len < 0x8000C ||
                output_buf_len <= 0xB )
            {
                ...
                sec_fr_send_template(data);
                ...
            }
    }
```

CVE-2025-20943



CVE-2025-20943

```
struct qsc_send_cmd_sendTemplate {  
...  
    unsigned int data_len;  
    char data[524274];  
    char data1_end[2];  
    unsigned int unknow_buf_len;  
...  
};
```

```
__int64 __fastcall sec_fr_decode_tmpl(  
    struct global_template_info *a1,  
    unsigned __int8 *input_buf_0x80000,  
    unsigned int unknow_buf_len,  
    int face_id)  
{  
...  
    if ( unknow_buf_len - 160 < 0x7FF61 )  
    {  
        buf_0x80064 = PAL_MemoryAlloc(0x80064LL);  
...  
    }  
}
```

- Here **v14** is controllable and can be used to perform an out-of-bounds write of 0x64 bytes.

```
__int64 __fastcall sec_fr_decode_tmpl_l_ver3(  
    struct global_template_info *a1,  
    unsigned __int8 *buf,  
    int data_len,  
    const uint8_t *buf_2,  
    int face_id,  
    int a6)  
{  
...  
    v14 = (unsigned int)(data_len - 32);  
    memcpy((__int64)&buf_2[v14],  
        (__int64)a1->arg3_info, 0x64uLL);  
...  
}
```

CVE-2025-20943

- The command with ID **0x17** is used to initialize a secfr template, with the size of the input data limited to less than 0x4B9.

```
void __fastcall sec_fr_cmd_handler(  
    void *input_buf,  
    unsigned int input_buf_len,  
    void *output_buf,  
    unsigned int output_buf_len)  
{  
    ...  
    switch ( *(_DWORD *)input_buf )  
    {  
        ...  
        case 0x17:  
            if ( input_buf_len < 0x4B9 || output_buf_len <= 7 )  
            {...}  
            sec_fr_init_template(a1, a2, a3, a4);  
            ...  
        }  
    }
```

CVE-2025-20943

- `Global_data` is a global variable, which means the contents of `arg3_info` are under our control.
- Combined with the previous `sec_fr_decode_tmpl_1_ver3` function, we can perform an out-of-bounds write at an arbitrary distance, writing any 0x64 bytes of data.

```
void *__fastcall sec_fr_init_template(
    void *a1,
    void *a2,
    int a3,
    void *a4,
)
{
    ...
    if ( (unsigned int)(*(__DWORD *)a1->data_size - 1) > 0x63 )
    {
        PAL_Log(4LL, "invalid store path");
        goto LABEL_4;
    }
    memset((__int8
*)Global_data.template_info.arg3_info, 0, 0x64uLL);
    ...
    memcpy((__int64)Global_data.template_info.arg3_info,
    (__int64)a1->addr, *(unsigned int *)a1->data_size);
    ...
}
```

CVE-2025-20943

★ Qualcomm

```
__int64 __fastcall
sec_fr_decode_tmpl_l_ver3(...)
{
...
    v14 = (unsigned int)(data_len - 32);
    memcpy((__int64)&buf_2[v14],
        (__int64)a1->arg3_info, 0x64uLL);
...
}
```

★ Exynos

```
__int64 __fastcall
sec_fr_decode_tmpl_l_ver3(...)
{
...
    v13 = (unsigned int)(data_len - 32);
    v14 = a4 + v13;
    *(_OWORD *)(v14 + 16) = *(_OWORD *)(a1
+ 786852);
    *(_OWORD *)(v14 + 32) = v15;
    *(_OWORD *)v14 = v16;
...
}
```

CVE-2025-20948

- The command with ID 0x12 is used to process a camera image frame, with the size of the input data limited to less than 0x7C, and it returns 0x42B bytes of data.

★ Exynos

```
__int64 __fastcall sec_fr_cmd_handler(int
a1, __int64 a2)
...
switch ( a1 )
{
...
case 0x2:
    v10 = sec_fr_on_enroll_frame(&n[1]);
...
}
...
}
```

★ Qualcomm

```
void *__fastcall sec_fr_cmd_handler(
void *input_buf,
unsigned int input_buf_len,
void * output_buf,
unsigned int output_buf_len)
{
...
switch ( *(_DWORD *)input_buf )
{
...
case 0x12:
    if ( input_buf_len < 0x7C ||
output_buf_len <= 0x42B )
    {...}
    sec_fr_on_enroll_frame(frame);
...
}
```

CVE-2025-20948

- Since the structure size is limited, where is the image frame stored???
- send image addr to TEE???

```
struct qsc_send_cmd_onEnrollFrameCDSP {  
    unsigned int cmd_id;  
    unsigned int frameCnt;  
    struct data_set data_set;  
    struct onEnrollFrameCDSP_I_input_part1  
input_part[2];  
    unsigned int pad;  
};
```

```
struct onEnrollFrameCDSP_I_input_part1  
{  
    char image_addr[8];  
    unsigned int width;  
    unsigned int height;  
    unsigned int unknow_data1;  
    unsigned int unknow_data2;  
    unsigned int arg4_1;  
    unsigned int format;  
    unsigned int srcLen;  
    unsigned int gain;  
    char arg6[8];  
    unsigned int brightnessValue;  
};
```

CVE-2025-20948

★ Qualcomm

```
__int64 __fastcall
sec_fr_wrapper_enroll_detect(...)
{
    ...
    PAL_Log(1LL, "1. S sec_fr_enroll_detect (%d)",
    *faceStep);
    v5 = register_data_to_cdsp_mem_info(a1,
    a2->StatusParam, a2->DataSet);
    ...
    v9 = SetRGBFrame(frameCnt, ImageData,
    &RGBFrame, &v28);
    ...
}
```

★ Exynos

```
__int64 __fastcall
sec_fr_wrapper_enroll_detect(...)
{
    ...
    v4 = sub_7FBAF0(1, "1. S sec_fr_enroll_detect
    (%d)");
    v5 = (_DWORD **)((_QWORD *)a2 + 2);
    if ( **v5 != 11 )
    {
        v6 = check_sec_camera_session_id(v4);
        memset(a1 + 30, 0, 0x27E64uLL);
    }
    ...
    v9 = SetRGBFrame(frameCnt, ImageData,
    &RGBFrame, &v28);
    ...
}
```

CVE-2025-20948

★ Qualcomm

```
__int64 __fastcall
register_data_to_cdsp_mem_info(...)
{
...
v7 = PAL_check_sec_camera_session_id();
...
}
```



```
__int64 PAL_check_sec_camera_session_id()
{
return qseeSecCamIsActive();
}
```

★ Exynos

```
__int64 __fastcall
sec_fr_wrapper_enroll_detect(...)
{
...
if ( **v5 != 11 )
{
v6 = check_sec_camera_session_id(v4);
memset(a1 + 30, 0, 0x27E64uLL);
...
}
```



```
__int64 check_sec_camera_session_id()
{
return 0LL;
}
```

CVE-2025-20948

In the Qualcomm environment, there are two ways to transmit data to a TA:

1. Directly pass the data structure required by the TA using the QSEECOM_send_modified_cmd_64 function.
2. Share data with the TA via exposed DMA(?) ports, namely the following.

```
r0q:/dev/mem_buf_vm # ls
qcom,cp_app          qcom,cp_pixel        qcom,hlos
qcom,cp_bitstream    qcom,cp_sec_display  qcom,oemvm
qcom,cp_camera        qcom,cp_spss_hlos_shared qcom,self
qcom,cp_camera_preview qcom,cp_spss_sp      qcom,trusted_vm
qcom,cp_cdsp          qcom,cp_spss_sp_shared
qcom,cp_non_pixel     qcom,cp_touch
```

CVE-2025-20948

- Here, data depends on the size we request via `srcLen` and is obtained from DMA. Meanwhile, `unknow_data1` and `unknow_data2`, although subject to boundary checks, are still controllable.

```
__int64 __fastcall transformYUV2BGR_rotate(
...
    __int64 data,
    int width,
    int height,
    int unknow_data1,
    int unknow_data2,...)

{
...
    if ( unknow_data2 > 1280 || unknow_data1 > 1280 ||
        unknow_data2 < height )
        goto LABEL_15;
    offset = = unknow_data2 * unknow_data1;
    LOBYTE(v14) = *(_BYTE *)(data + offset + v82);
...
}
```

Common vulnerabilities



- securefp

The Fingerprint module is a trusted application running in the TEE, designed to securely perform fingerprint enrollment, feature matching, and encrypted template storage.

CVE-2025-21044

```
void tl_cmd_handler(void *cmd, unsigned
int cmdlen, void *rsp, unsigned int rsplen)
{
...
switch ( *(_DWORD *)cmd )
{
...
case 0xC:
    cmd = (void *)run_control_op(struct
tl_controlop_req_s *)cmd, cmdlen, (struct
tl_controlop_rsp_s *)rsp, rsplen);
...
}
...
}
```

tl_do_extra_control



do_extra_control



unknow_func



do_extra_control_d



- Samsung's code performs preliminary processing.

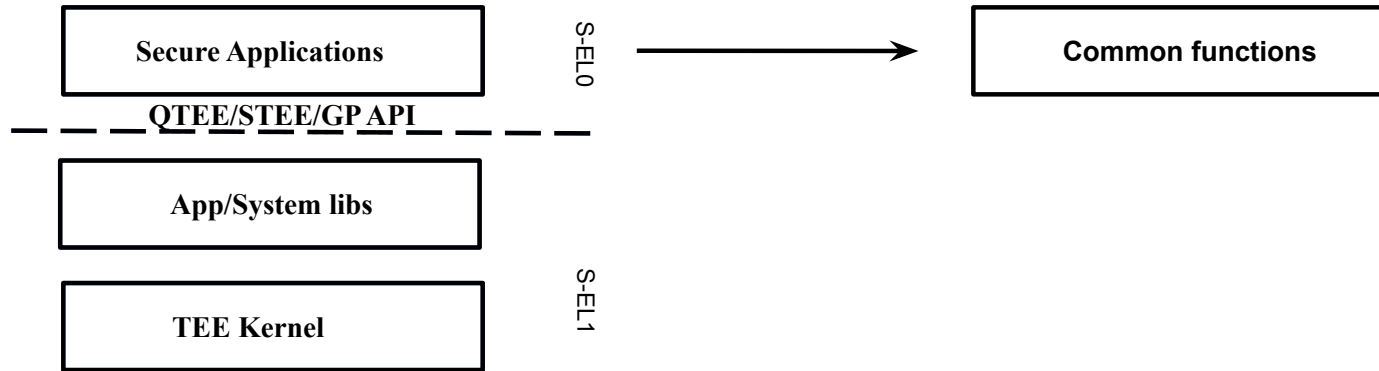
CVE-2025-21044

- `tl_do_extra_control` will call `decode_each_template`

```
int decode_each_template(char* single_template) {
    char* decrypted_template =
(char*)TEE_Malloc(MAX_TEMPLATE_SIZE);
    ...
    unsigned int inner_template_len;
    ARR_TO_INT32(&inner_template_len, single_template, 8);
    ...
    if (internal_cmp_hmac_vx(decrypted_template + 8,
inner_template_len) != 0) {
        return ERROR_HMAC_FAILED;
    }
    ...
    char* target_buffer = get_target_buffer();
    ...
    memcpy(target_buffer, decrypted_template + 312,
inner_template_len);
    ...
}
```

- An integer truncation occurs when the input data is too large.
- Inside `internal_cmp_hmac_vx`, a check $(a2 + 304) + 32 \leq a1$ is performed, which can trigger an integer overflow.
- `get_target_buffer` obtains a buffer much smaller than `inner_template_len`.

Common architectural vulnerabilities



Unique vulnerabilities



- mPOS

mPOS is a trusted application that runs in a secure environment to execute mobile payment transactions, key management, and cryptographic operations, preventing payment information, keys, and transaction data from being stolen or tampered with by the Rich Execution Environment (e.g., Android application layer).

CVE-2025-20904(Qualcomm)

```
int __fastcall process_cmd(int a1, unsigned __int8 *a2)
{
    ...
    switch ( *(_DWORD *)a1 )
    {
        ...
        case 5:
            ...
            v6 = a2[24] | (a2[25] << 8) | ((a2[26] | (a2[27] << 8)) << 16);
            if ( v6 <= 0x3E800 && !saveBackgroundResource((unsigned int *)a2 + 7, v6)
            )
                goto LABEL_20;
            ...
        }
    LABEL_20:
        ...
        if ( launchPinpad(1, 0) )
            v4 = 7;
        ...
    }
```

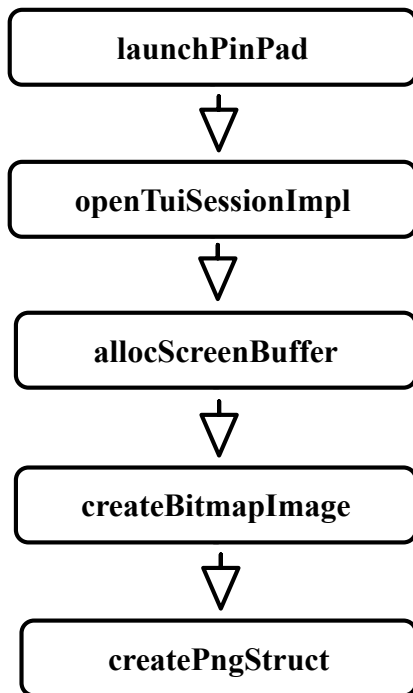
saveBackgroundResource



Initialize the background
storage space.

```
int __fastcall saveBackgroundResource(unsigned int *src, unsigned int
size)
{
    ....
    dst = (_BYTE *)((int (__fastcall *)(unsigned int))TZ_malloc)(size);
    v6 = gBgImageSize_0[0];
    *gBgImage_0[0] = (int)dst;
    *v6 = size;
    if ( dst )
    {
        memcpy(dst, src, size);
        result = 0;
    }
}
```

CVE-2025-20904(Qualcomm)

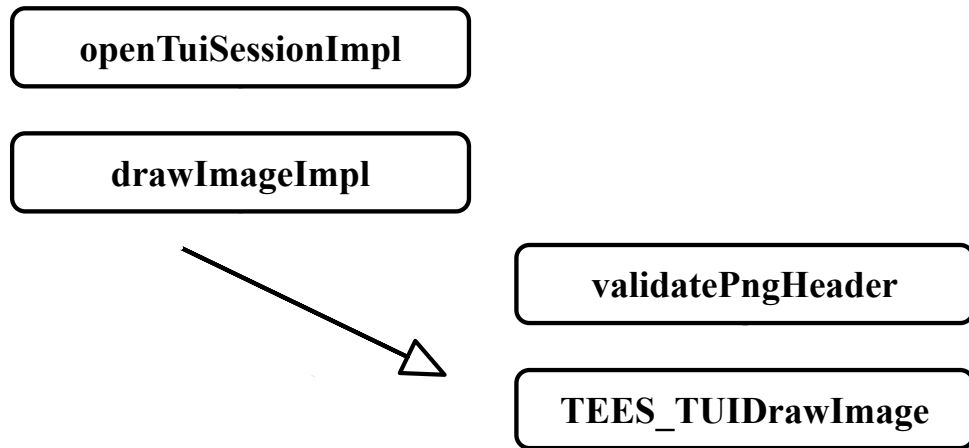


```
_DWORD *__fastcall createPngStruct(  
    int *png_ptr,  
    int *a2,  
    _DWORD *src,  
    _DWORD *width,  
    _DWORD *height,  
    _DWORD *color_type,  
    _DWORD *bit_depth)  
{  
    .....  
    png_set_read_fn(*png_ptr, (int)io_ptr, (int)off_1181B0);  
    png_set_sig_bytes(*png_ptr, 8);  
    png_read_info(*png_ptr, *info_ptr);  
    io_ptr = 0;  
    png_get_IHDR(*png_ptr, *info_ptr, width_1, height, bit_depth, color_type, 0, 0,  
0);  
}  
return io_ptr;  
}
```

A code block showing the implementation of the `createPngStruct` function. A black arrow originates from the `createPngStruct` box in the flowchart and points to the function signature of the code block.

CVE-2025-20904

- What about Exynos and MTK



- The OEM's TEE function library implementation is essentially a wrapper over an open-source PNG parsing library.

Unique vulnerabilities

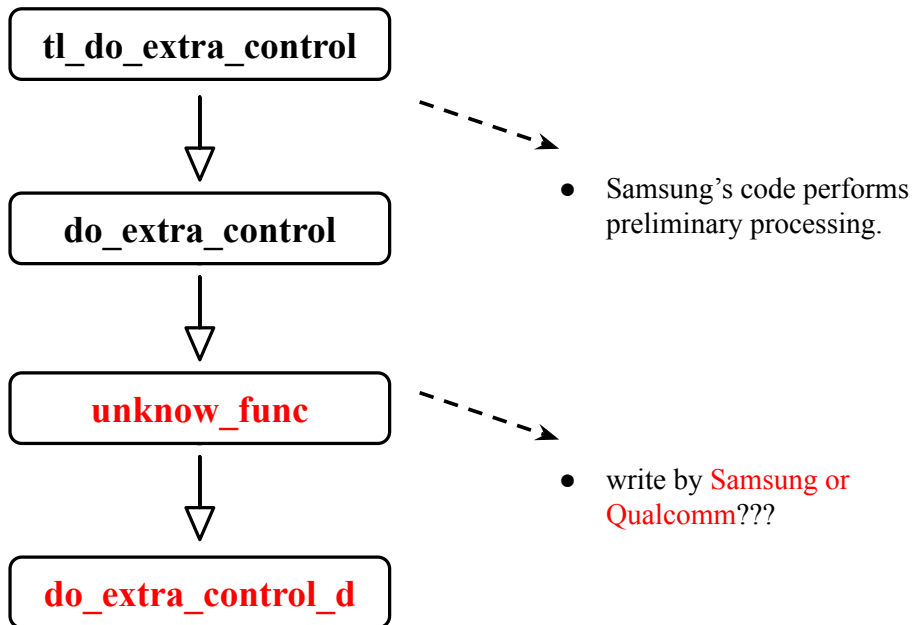


- securefp

The Fingerprint module is a trusted application running in the TEE, designed to securely perform fingerprint enrollment, feature matching, and encrypted template storage.

Fingerprint vulnerabilities

```
void tl_cmd_handler(void *cmd, unsigned
int cmdlen, void *rsp, unsigned int rsplen)
{
...
switch ( *(_DWORD *)cmd )
{
...
case 0xC:
    cmd = (void *)run_control_op(struct
tl_controlop_req_s *)cmd, cmdlen, (struct
tl_controlop_rsp_s *)rsp, rsplen);
...
}
...
}
```



Fingerprint vulnerabilities

```
__int64 __fastcall do_extra_control_d(
    void *global_data,
    char *input_buf,
    unsigned int input_buf_len,
    char *output_buf,
    unsigned int *output_buf_len)
{
    ...
    switch ( *(_DWORD *)global_data )
    {
        ...
        case 0x67:
            if ( output_buf && output_buf_len )
            {
                if ( (unsigned int)((__int64 (__fastcall
                *(char *))qfp_get_esp_data)(output_buf + 8)
                )
                ...
            }
            ...
    }
}
```

- When global_data is 0x67, neither output_buf nor output_buf_len is validated.



```
__int64 __fastcall
liveness_get_esp_data(unsigned __int64
output_buf)
{
    sg_memscpy(output_buf, 0x19344uLL,
(unsigned __int64)off_FF5CC370,
0x19344uLL);
}
```

Fingerprint vulnerabilities

★ Qualcomm (Samsung S22)

```
__int64 __fastcall do_extra_control_d(...)  
{  
...  
switch ( *(_DWORD *)global_data )  
{  
...  
case 0x67:  
if ( output_buf && output_buf_len )  
{  
if ( (unsigned int)((__int64 (__fastcall  
*)(char *))qfp_get_asp_data)(output_buf + 8)  
)  
...  
}  
...  
}  
}
```

```
__int64 __fastcall  
liveness_get_asp_data(unsigned __int64  
output_buf)  
{  
sg_memscpy(output_buf, 0x19344uLL,  
(unsigned __int64)off_FF5CC370,  
0x19344uLL);  
}
```

★ Exynos (Samsung S24)

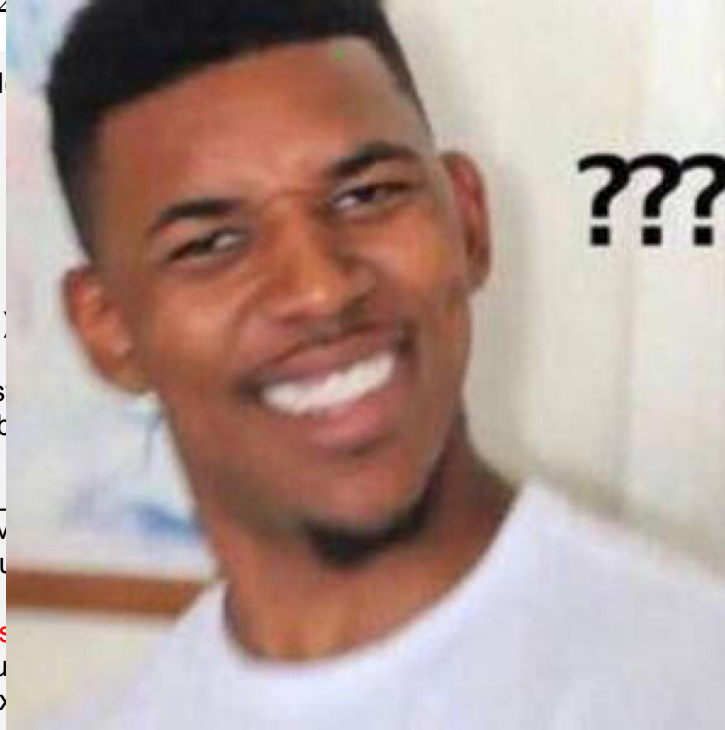
```
.rodata:0000000000025BDC aQfpGetAspData DCB "qfp_get_asp_data",0  
.rodata:0000000000025BDC ; DATA XREF: qfp_get_asp_data+3C1
```

```
__int64 __fastcall do_extra_control_d(...)  
{  
...  
switch ( *(_DWORD *)global_data )  
{  
...  
case 0x67:  
v12 = qfp_get_asp_data(a1, a4);  
...  
}  
}
```

Fingerprint vulnerabilities

★ Qualcomm (Samsung S22)

```
__int64 __fastcall do_extra_control_d
{
...
switch ( *(_DWORD *)global_data )
{
...
case 0x67:
    if ( output_buf && output_buf_len )
    {
        if ( (unsigned int)((__int64 (__fas
*)(char *))qfp_get_asp_data)(output_b
)
...
    }
...
}
```



★ Exynos (Samsung S24)

```
125BDC aQfpGetAspData DCB "qfp_get_asp_data",0
125BDC ; DATA XREF: qfp_get_asp_data+3C1
```

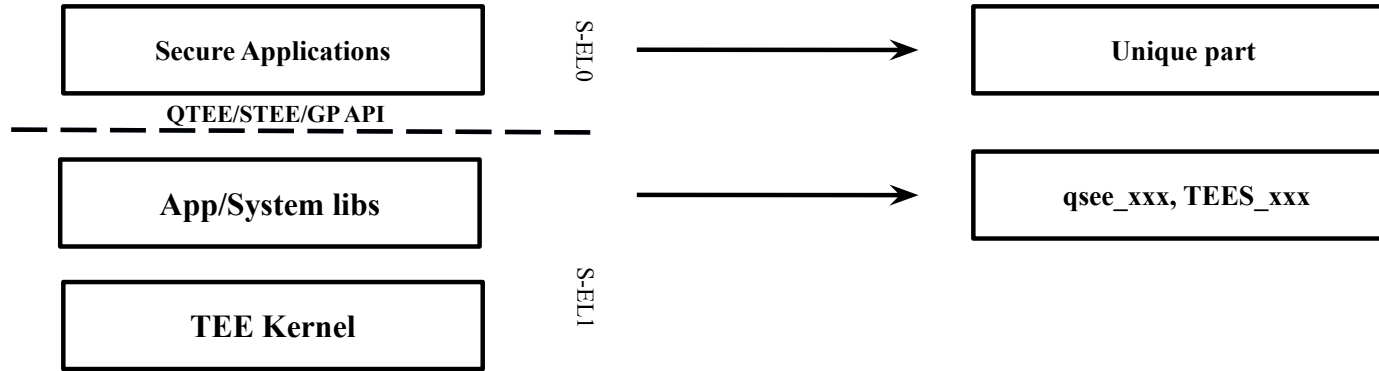
```
__fastcall do_extra_control_d(...)
```

```
*(_DWORD *)global_data )
```

```
<67:
```

```
qfp_get_asp_data(a1, a4);
```

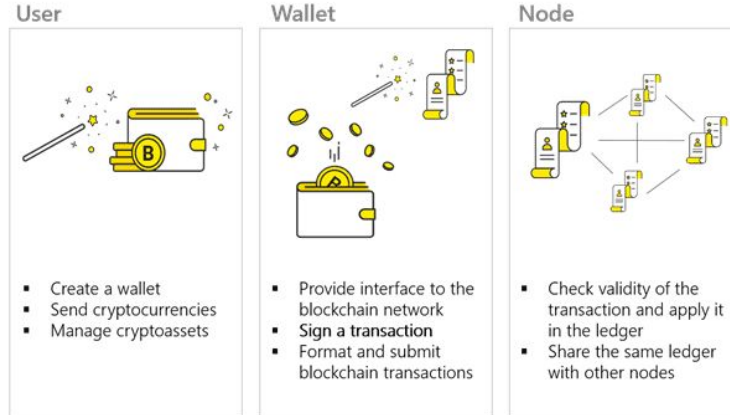
Architecture-specific vulnerabilities



Special modules

Special module

- Blockchain Wallet



- Knox Vault



Blockchain Wallet

- Key pair (public key/private key)
- Derived elements (wallet address/mnemonic phrase)
- “Private key manager”

12:45

< **New recovery phrase**

Write down your recovery phrase and keep it in a safe place. To prevent anyone else from accessing your cryptocurrency, don't share this phrase with anyone else or store it anywhere online (email, cloud, website, etc.).

 **Don't take a picture of this screen.**

9. voice
10. contact
11. certified
12. store

< 3/3

☒ I wrote it down safely

Next

12:45

< **Double check your phrase**

Enter the words below to make sure you've stored your recovery phrase correctly

Enter word 3.
super

Enter word 7.
cruise

Enter word 11.
gro

q w e r t y u i o p
a s d f g h j k l
z x c v b n m

Blockchain Wallet

- Key pair (public key/private key)
- Derived elements (wallet address/mnemonic phrase)
- “Private key manager”

Why TEE?



- Shared operating environment
 - Rich OS/REE
- Huge attack surface
 - Operating system vulnerabilities
 - Malicious APP attack
 - physical attack

Blockchain Wallet

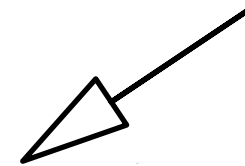
- Key pair (public key/private key)
- Derived elements (wallet address/mnemonic phrase)
- “Private key manager”

Why TEE?



- Shared operating environment
 - Rich OS/REE
- Huge attack surface
 - Operating system vulnerabilities
 - Malicious APP attack
 - physical attack

- Randomness
- Key storage isolation
- Confidential transaction signing process
- Trusted user interface



Blockchain Wallet

- TA is located in the “assets” directory of apk
- Differences in architectural implementation
 - TEE general interface
 - Functional interfaces (e.g. image parsing)
- bc_core
 - Key management
 - transaction signature
- bc_tui
 - pin code setting/verification
 - Transaction details/confirmation
 - mnemonic display
- bc_tvm
 - Trusted virtual machine

```
→ ta ls *
bf:
2100 8825 8835 8845 8855 980 9820 9820_BTS 990 9925 9945 9955
qc:
7125 7150 7225 7250 7325 8150 8150_FUSION 8150_FUSION_LA20
8150_FUSION_LA20_BTS 8250 8350 8450 8475 8550 8650 8750
→ ta ls qc/8750
bc_core.mp3 bc_tui.mp3 bc_tvm.mp3
→ ta ls bf/9955
00000000000000000000000042434e545549.mp3
0000000000000000000000004243636f7265.mp3
```

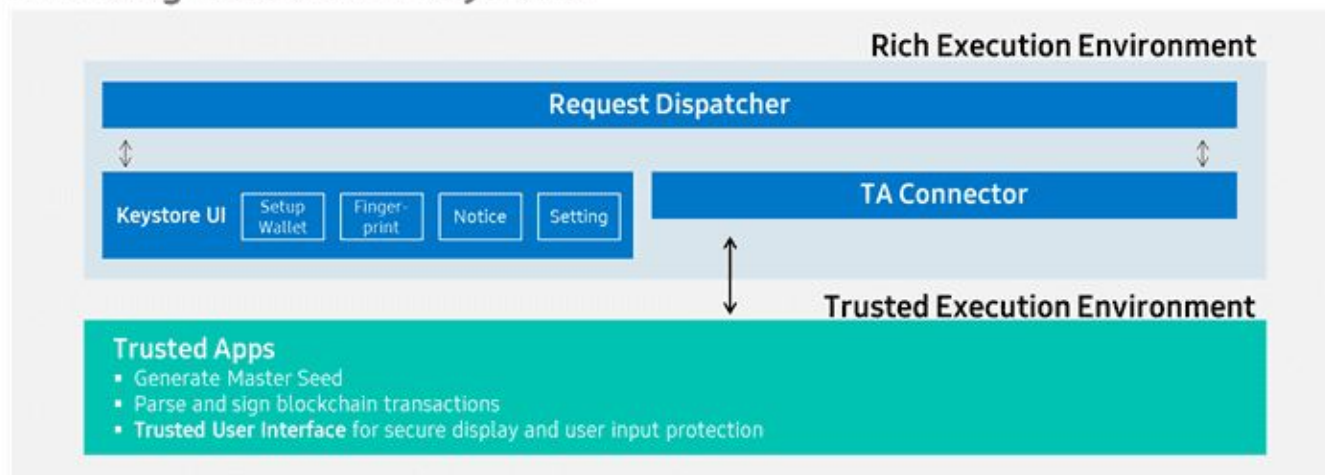
```
static {
    TA_INFO = ChipsetInfo.m6589d() ? new BcTvmTAInfo() : new BcTuiTAInfo();
    mResetPending = false;
    List<String> listAsList = Arrays.asList("8750", "8650", "8550", "8475", "8450", "8350", "7325");
    mQCTUI_1_5_SupportedBoardList = listAsList;
    HashSet<String> hashSet = new HashSet<>();
    mQCTUI_1_5_SupportedBoardSet = hashSet;
    hashSet.addAll(listAsList);
    bQC = ChipsetInfo.m6588c();
    bBF = ChipsetInfo.m6587b();
}
```

```
/* renamed from: d */
public static boolean m6589d() {
    String str = f10924b;
    return str.equals("8450") || str.equals("8475") || str.equals("8550") || str.equals("8650") || str.equals("8750");
}
```

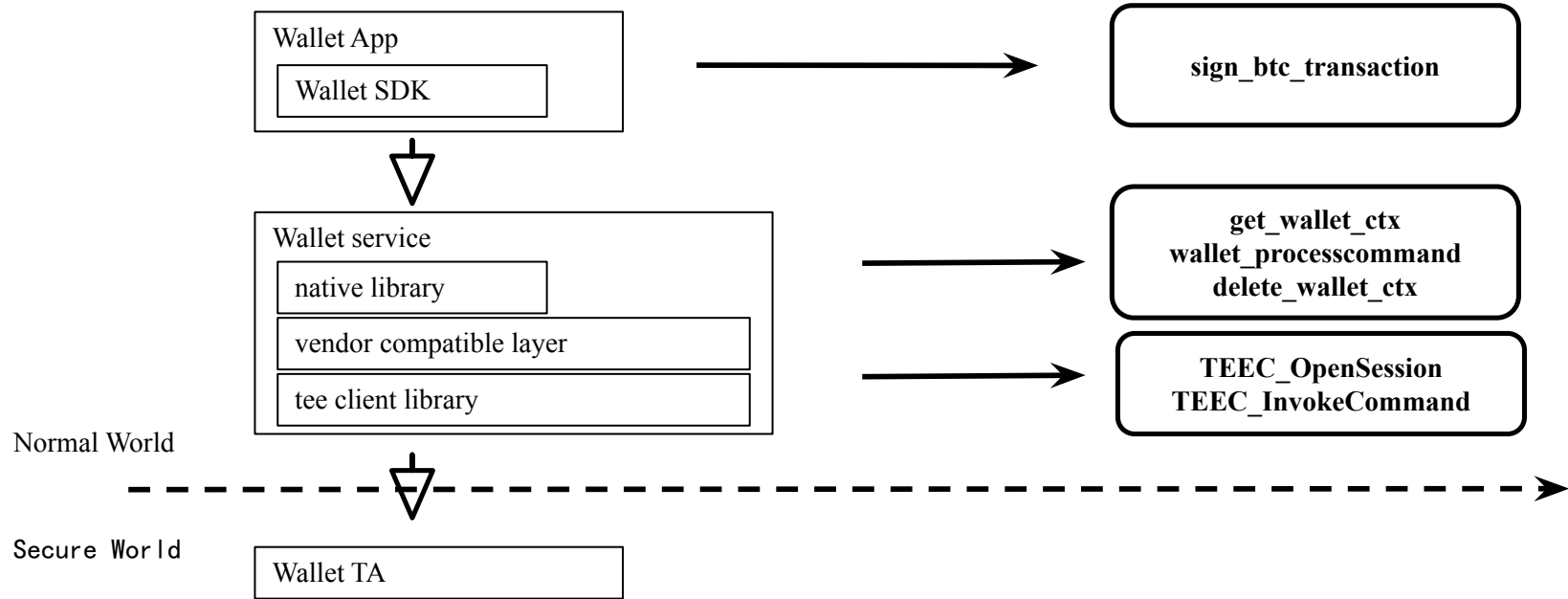
Blockchain Wallet



Samsung Blockchain Keystore



Blockchain Wallet



Blockchain Wallet

bc_core

- CVE-2025-21017 (Decryption related | OOB write)
- CVE-2025-21018 (Transaction Serialization/Deserialization Module | Deserialization)

bc_tui/bc_tvm

- CVE-2025-20901 (Image parsing Module | off-by-one)
- CVE-2025-20900 (Image parsing Module | Integer overflow)
- CVE-2025-21020 (Image parsing Module | OOB write)
- CVE-2025-21021 (Pin code processing | OOB write)



Some similar code
implementations

mPOS

CVE-2025-21021

```
uint32_t __cdecl drawImageImpl(uint32_t x, uint32_t y, uint8_t *pPngData, uint32_t length, uint32_t rotation)
{
    .....
    pngRaw.buffer = pPngData;
    pngRaw.size = length;
    pngRaw.offset = 8;
    .....
    ret = createPngStruct(&png_ptr, &info_ptr, &pngRaw, &width, &height, &color_type, &bit_depth);
    else
    {
        png_start_read_image();
        .....
        setPixelsClockwiseRotation(x, y, width, height, png_ptr, color_type);
    }
    else
    {
        ....
        setPixelsNoneRotation(x, y, width, height, png_ptr, color_type);
    }
}
```

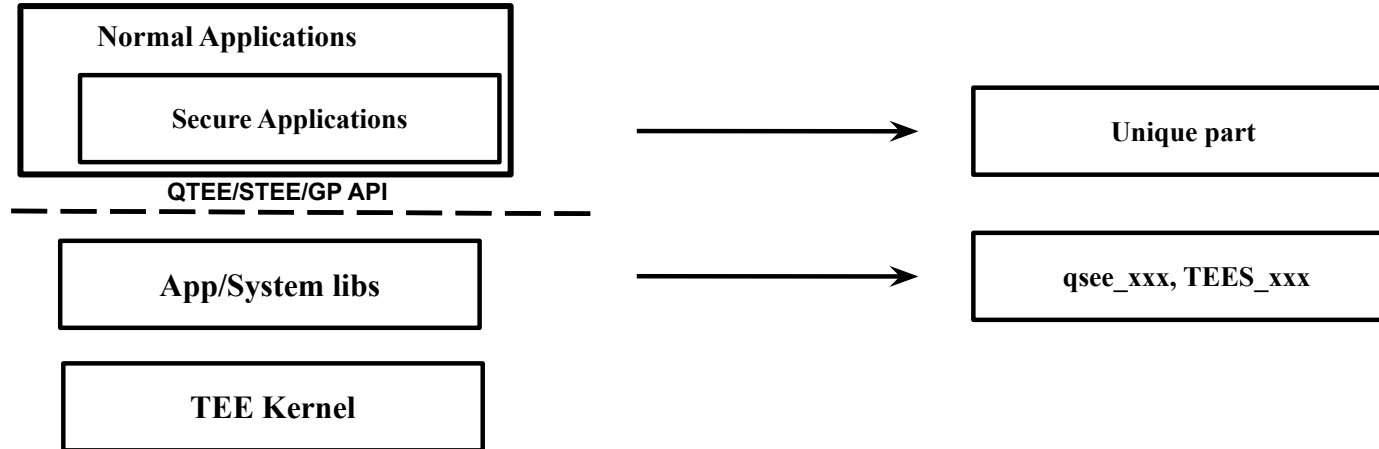
```
uint32_t __cdecl createPngStruct(...)
{
    .....
    if ( *info_ptr )
    {
        png_set_read_fn(*png_ptr, pngRaw,
readPngWithIo);
        png_set_sig_bytes(*png_ptr, 8LL);
        png_read_info();
        png_get_IHDR(....);
        return 0;
    }
    ....
}
```

CVE-2025-21021

```
uint32_t __cdecl drawImageImpl(uint32_t x, uint32_t y, uint8_t
{
    .....
    pngRaw.buffer = pPngData;
    pngRaw.size = length;
    pngRaw.offset = 8;
    .....
    ret = createPngStruct(&png_ptr, &info_ptr, &pngRaw, &width
else
{
    png_start_read_image();
    .....
    setPixelsClockwiseRotation(x, y, width, height, png_ptr, c
}
else
{
    ....
    setPixelsNoneRotation(x, y, width, height, png_ptr, color_t)
}
}

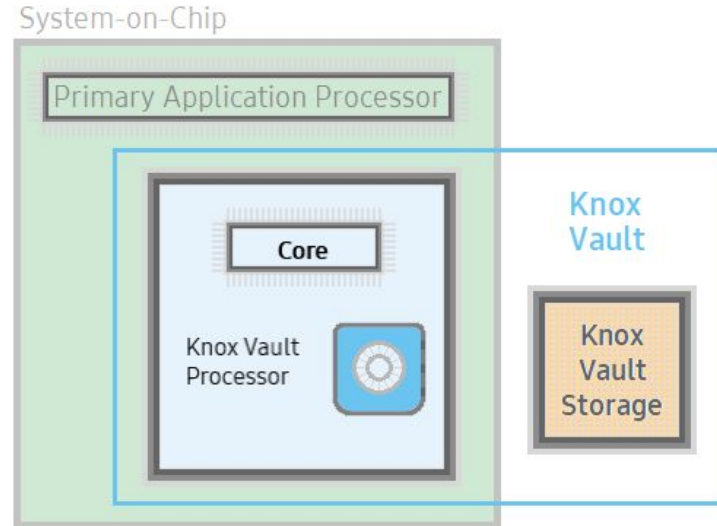
void __cdecl setPixelsNoneRotation(...)
{
    .....
    for ( row = 0; height > row; ++row )
    {
        png_read_row(png_ptr, row_buffer, 0LL);
        screenOffset = &g_screen[(row + y) * stride_x_BytePerPixel +
X_x_BytePerPixel];
        if ( (color_type & 4) == 0 )
            memset(screenOffset, 255, width_x_BytePerPixel);
        for ( i = 0; width > i; ++i )
        {
            .....
            memcpy(&screenOffset[itor_x_BytePerPixel],
&row_buffer[itor_x_BytePerPixel], 3uLL);
            .....
        }
    }
}
```

TA application in Normal App



Knox Vault

- Independent security chip
- Store sensitive information
- Core functions
 - Secure password authentication
 - Credential storage
 - Support for StrongBox
- Knox Vault Architecture
 - Subsystem (integrated in SoC)
 - **Storage (SoC independent)**
- Secure Communication (I2C)
- Double physical isolation
- Active defense (probes, FIB, fault injection, side channels, etc.)
- End-to-end, beyond traditional TEE



Knox Vault trustlet

- **Hardware Vault / Keystore**
 - Responsible for general key management, cryptographic operations, and credential storage.
- **Weaver Component**
 - Handles data storage that is tightly bound to user authentication (such as PINs or passwords) and provides protection against brute-force attacks.
- **Secure NVM (Non-Volatile Memory)**
 - Provides a general, protected persistent storage area and manages its lifecycle
- **Security Hub**

SVE-2024-2304(CVE-2025-20983, CVE-2025-20982): Out-of-bounds write in KnoxVault trustlet

Severity: High

Affected versions: Android 14, 15

Reported on: December 8, 2024

Disclosure status: Privately disclosed

Out-of-bounds write in KnoxVault trustlet prior to SMR Jul-2025 Release 1 allows local privileged attackers to write out-of-bounds memory.

The patch adds proper input validation.

Knox Vault trustlet

```
1 __int64 cmapi_drv_open()
2 {
3     unsigned int v0; // w19
4     _DWORD *errno_addr; // x0
5
6     v0 = 0;
7     crypto_manager_fd = open("/dev/crypto_manager", 3, 0LL); |
8     if ( crypto_manager_fd < 0 )
9     {
10         errno_addr = get_errno_addr();
11         printf("[CM] TA: ERROR: cm_drv_open: errno: %d\n", *errno_addr);
12         v0 = -65536;
13     }
14     printf("[CM] TA: driver open: success\n");
15     return v0;
16 }
```

Crypto Engine

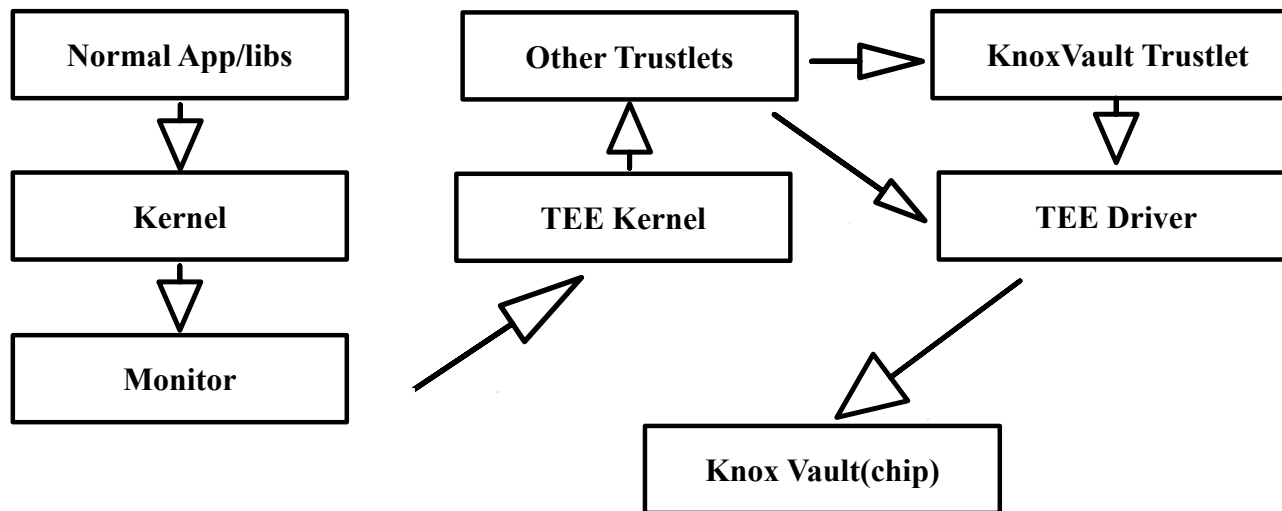
The Crypto Engine supports symmetric encryption to verify authentication codes after receiving packets from the Knox Vault Processor and also to enhance performance.

Crypto Engine

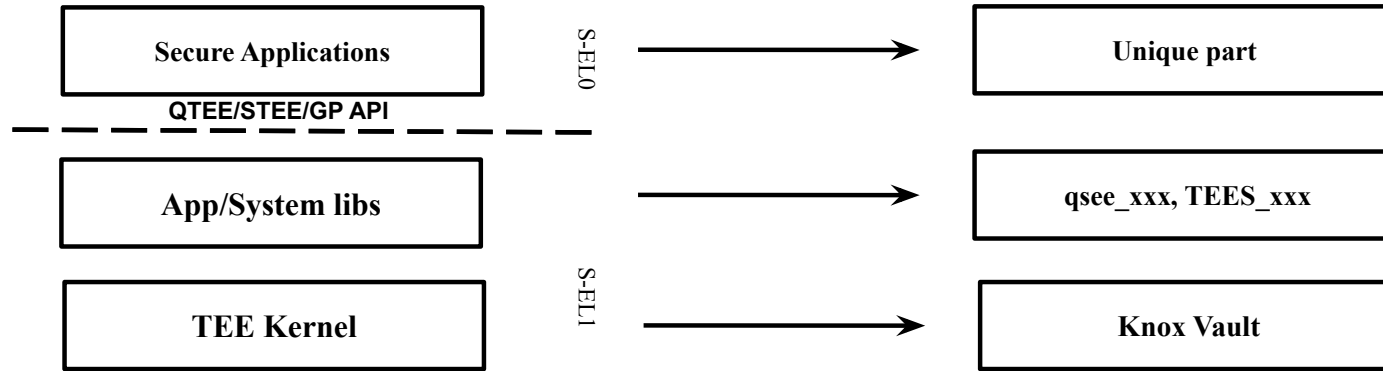
A hardware cryptographic module provides the following cryptographic functions:

- Advanced Encryption Standard (AES) encryption and decryption
- Deterministic random bit generator (DRBG) random number generation
- Secure hashing algorithm (SHA) hashing
- HMAC keyed hashing for message authentication code
- RSA and ECC key generation and services

CVE-2025-20983, CVE-2025-20982



Summary



Q & A

- Juntao Wu(@wugedbl)
- Hangyu Hua(@HBh25Y)
- Shuffle Team(@ShuffleTeam_AI): <https://shuffleteam.github.io/>