Unearthing the TrustedCore

A Critical Review on Huawei's Trusted Execution Environment

August 11, 2020

Marcel Busch, Johannes Westphal, Tilo Müller

Friedrich-Alexander-University Erlangen-Nürnberg, Germany

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Motivation

TEEs are the backbone of many security-critical services on Android devices.

What to expect?

- Share (general) insights from analysis of proprietary TEE, *TrustedCore*
- Elaborate on inner workings of selected components
- Show design and implementation flaws

Outline

Background

TrustedCore Architecture

Secure Loader

The Android Keystore System

Memory Corruptions & Exploitation

Conclusion

Background

Trusted Execution Environments (TEEs)

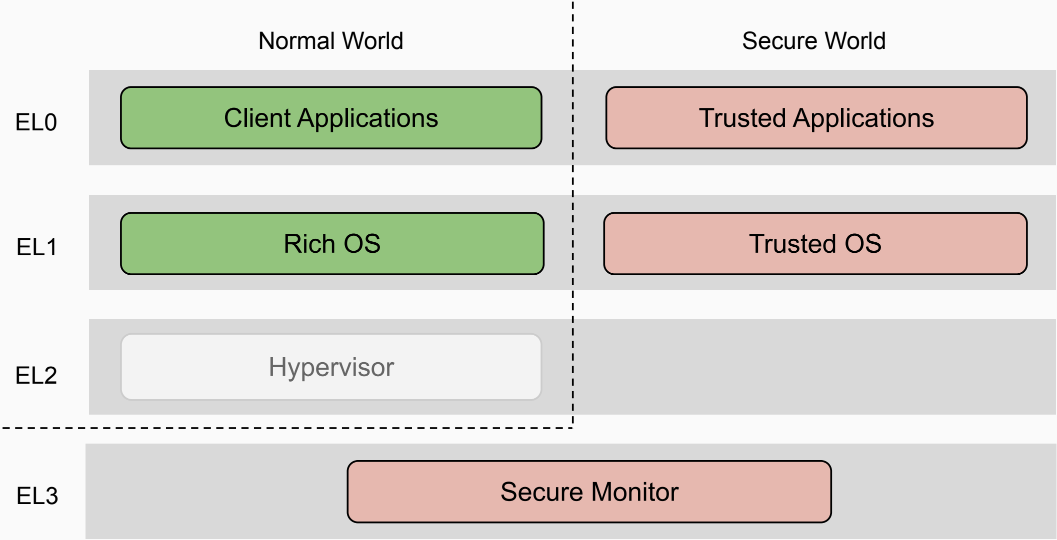
Isolated execution context providing

- Integrity and
- Confidentiality

TEE-enabling technologies

- AMD Platform Security Processor
- Intel Software Guard Extensions
- ARM TrustZone
- ...

ARM TrustZone on ARMv8-A Systems



TEEs in the Field (on Android)

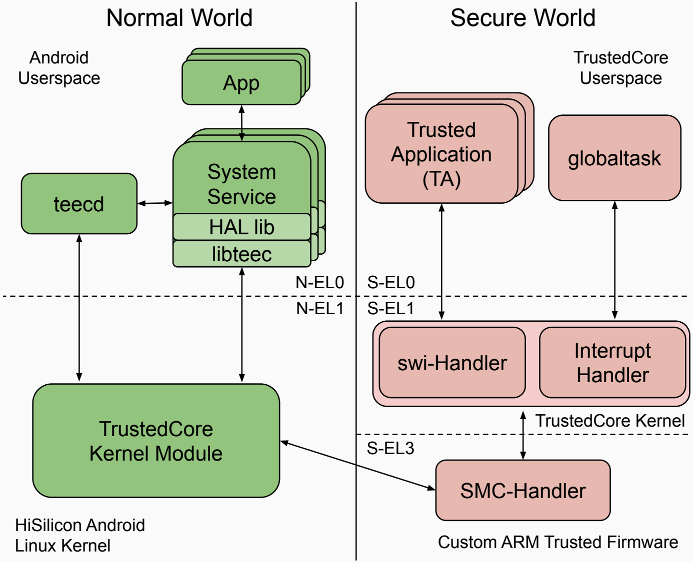
- Qualcomm Secure Execution Environment — 2016 [4, 3]
  - Pixel devices
  - Nexus devices
  - ...
- Kinibi by Trustonic — 2017 [5], 2018 [7, 8], 2019 [2]
  - Samsung Exynos devices up to Samsung Galaxy S9
  - ...
- TEEGris by Samsung — 2019 [1]
  - Samsung Exynos devices from Samsung Galaxy S10
  - ...
- TrustedCore by Huawei — 2015 [9], 2016 [10]
  - Up to Emotion UI 8 (e.g., Huawei P9, P10, P20)
- iTrustee by Huawei — N/A
  - From Emotion UI 9 (e.g., Huawei P30 and P40)

Applications

- PIN/pattern/password authentication
- Biometric authentication
  - Fingerprint
  - FaceID
- Digital rights management
- Mobile payment
- Full-disk encryption
- ...

TrustedCore Architecture

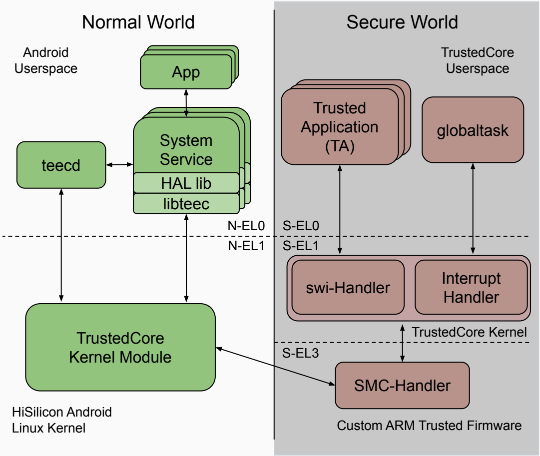
Overview



Normal World
Secure World
Android Userspace
App
System Service
HAL lib
libteec
teecd
TrustedCore Userspace
Trusted Application (TA)
globaltask
N-EL0
S-EL0
N-EL1
S-EL1
swi-Handler
Interrupt Handler
TrustedCore Kernel
S-EL3
TrustedCore Kernel Module
SMC-Handler
HiSilicon Android Linux Kernel
Custom ARM Trusted Firmware

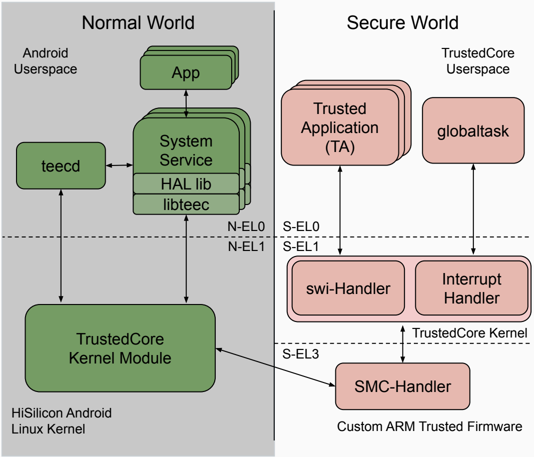
TrustedCore – Normal World

- N-EL0
  - Apps
  - System Services
  - `teecd`
- N-EL1
  - Linux Kernel Module



TrustedCore – Secure World

- S-EL0
  - Trusted Applications
  - globaltask
- S-EL1
  - TrustedCore Kernel
- S-EL3
  - Custom ARM TrustedFirmware

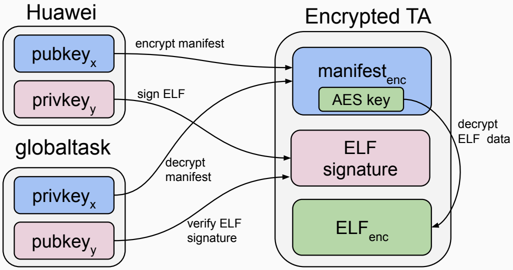


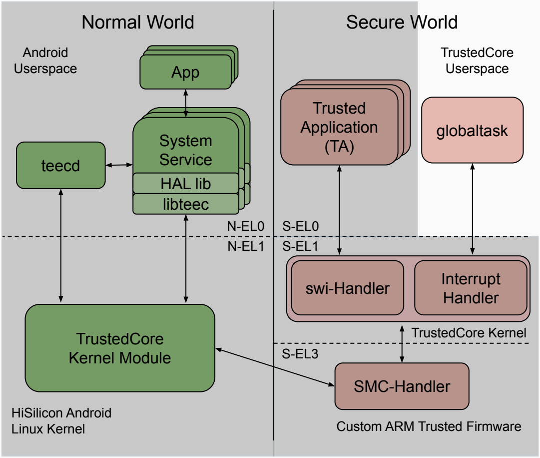
Secure Loader

Loading Encrypted Trusted Applications

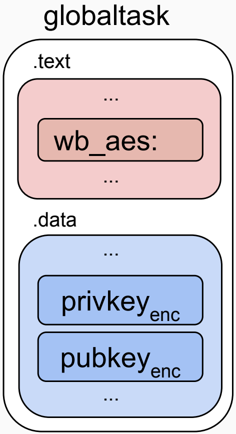
```
root@HWVNS-H:/ # ls /system/bin/*.sec
/system/bin/6c8cf255-ca98-439e-a98e-ade64022ecb6.sec
/system/bin/79b77788-9789-4a7a-a2be-b60155eef5f4.sec
/system/bin/868ccafb-794b-46c6-b5c4-9f1462de4e02.sec
/system/bin/883890ba-3ef8-4f0b-9c02-f5874acbf2ff.sec
/system/bin/9b17660b-8968-4eed-917e-dd32379bd548.sec
/system/bin/b4b71581-add2-e89f-d536-f35436dc7973.sec
/system/bin/fd1bbfb2-9a62-4b27-8fdb-a503529076af.sec
/system/bin/fpc_1021_ta.sec
/system/bin/fpc_1021_ta_venus.sec
/system/bin/fpc_1022_ta.sec
/system/bin/syna_109A0_ta.sec
```

Loading Encrypted Trusted Applications (cont.)





Protection of Crypto Keys



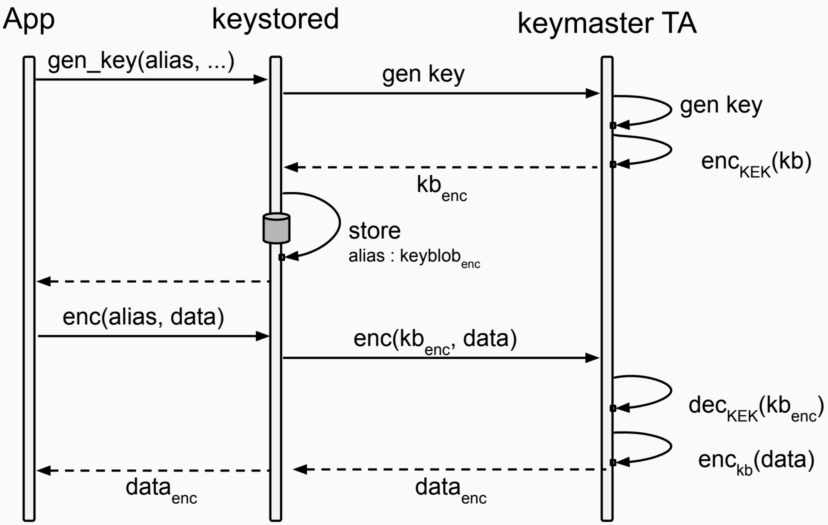

```
char globaltask[] = { ... }; // globaltask binary

int main(){
    char *pubkey_dec[0x1000] = { 0 };
    char *privkey_dec[0x1000] = { 0 };
    char* (*wb_aes) (char*, char*, unsigned int);

    mprotect(globaltask, sizeof(globaltask),
        PROT_READ|PROT_WRITE|PROT_EXEC);

    pubkey_enc = globaltask + <pubkeyenc_off>;
    privkey_enc = globaltask + <privkeyenc_off>;
    wb_aes  = globaltask + <wb_aes_off>;

    wb_aes(pubkey_enc, pubkey_dec, <pubkey_sz>);
    hexdump("privkey:", pubkey_dec, <pubkey_sz>);

    wb_aes(privkey_enc, privkey_dec, <privkey_sz>);
    hexdump("privkey:", pubkey_dec, <privkey_sz>);

    return 0;
}
```

Scope & Consequences

- Analysis of 133 firmware images distributed from July 2015 until April 2018
- 119 images using white-box crypto scheme
- Decryption of “confidential” TAs on models from 2016 (P9 Lite) until 2018 (P20 Lite)
- TCB size 16 times bigger than reported by Cerdeira et al. [6]

The Android Keystore System

Export-Protected Crypto Keys



kb - keyblob KEK - Key Encryption Key

The Key Encryption Key (KEK)

- blob contains encrypted key and hidden params
  - secret is a constant
- keyblob is protected by hmac
  - secret is a constant

```
struct keyblob {
  uint8_t hmac[32];
  uint8_t iv[16];
  uint8_t magic[4];
  uint32_t unknown;

  uint32_t keymaterial_offset;
  uint32_t keymaterial_size;
  uint32_t key_params1_count_offset;
  uint32_t key_params2_count_offset;
  uint32_t key_params1_data_offset;
  uint32_t key_params1_data_size;
  uint32_t hidden_params_count_offset;
  uint32_t hidden_params_data_offset;
  uint32_t hidden_params_data_size;
  uint32_t keyblob_size;
  uint8_t blob[]; // C99 FAM
}
```

Scope & Consequences

- 133 firmware images (from July 2015 until April 2018) use contant KEK
- Extract export-protected crypto keys
- Spoof `keyblobs`
- Off-device brute-forcing of full-disk encryption

Memory Corruptions & Exploitation

Memory Corruption in keymaster TA

- Stack-based buffer overflow in RSA key pair export routine

1. Craft `keyblob` with exploit payload using constant secrets
2. Import crafted `keyblob` into keymaster TA
3. Export crafted `keyblob` (triggers overflow)

Exploit Mitigations

$W \oplus X$	Stack Canaries	ASLR
●	◒	◒

- Stack canaries
  - Constant values
- ASLR
  - Low entropy
  - Reloaded to same address after crash

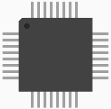
Escalating to S-EL1/S-EL3

- ~174 system calls available from S-EL0
- e.g., mapping of physical memory pages to TA virtual address space
- Flawed/ineffective range check for S-EL1 and S-EL3

Conclusion

Lessons Learned – Hardware-Protected Crypto Keys

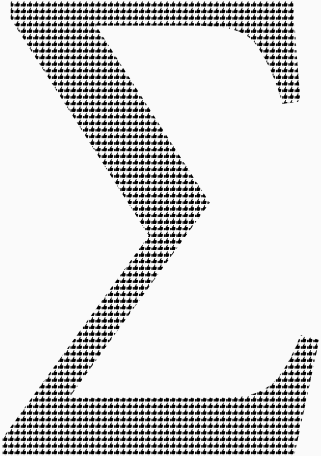
- ARM TrustZone == TEE construction kit
  - Confidentiality and integrity depend on hardware and software choices
- Severity of software-based protection designs
  - Leakage of KEK disables device-binding for an entire generation of phones
  - PIN/pattern/password can be brute-forced off-device



Lessons Learned – Attack Surface

- Trusted Computing Base is huge
- Trusted Computing Base attack surface is wide
- TAs are written in C/C++ and prone to memory corruption bugs
- Risk of exploitation not effectively mitigated

Summary



- Reviewed Huawei’s TrustedCore
- Examined and broke secure loader
- Examined and broke Android keystore system
- Found and exploited previously unknown memory corruption bug
- Escalated privileges to S-EL1/S-EL3
- Shared lessons learned

Questions?

eMail: `marcel.busch@fau.de`
Twitter: `@0ddc0de`

References (1)

T. Alexander.

Reverse-engineering samsung exynos 9820 bootloader and tz.

https://allsoftwaresucks.blogspot.com/2019/05/reverse-engineering-samsung-exynos-9820.html, 2019.

Accessed: 2019-08-30.

A. Alexandre, G. Joffrey, and P. Maxime.

A deep dive into samsung's trustzone (part 1).

https://blog.quarkslab.com/a-deep-dive-into-samsungs-trustzone-part-1.html, 2019.

Accessed: 2020-03-15.

References (2)

G. Beniamini.

Extracting qualcomm's keymaster keys - breaking android full disk encryption.

https://bits-please.blogspot.com/2016/06/extracting-qualcomms-keymaster-keys.html, 2016.

Accessed: 2019-12-28.

G. Beniamini.

Qsee privilege escalation vulnerability and exploit (cve-2015-6639).

https://bits-please.blogspot.com/2016/05/qsee-privilege-escalation-vulnerability.html, 2016.

Accessed: 2019-08-28.

References (3)

G. Beniamini.
Trust issues: Exploiting trustzone tees.
https://googleprojectzero.blogspot.com/2017/07/trust-issues-exploiting-trustzone-tees.html, 2017.
Accessed: 2019-08-28.

D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto.
Sok: Understanding the prevailing security vulnerabilities in trustzone-assisted tee systems.
In *Proceedings of the IEEE Symposium on Security and Privacy (S&P), San Francisco, CA, USA*, pages 18–20, 2020.

D. Komaromy.

Unbox your phone - part i.

https://medium.com/taszksec/unbox-your-phone-part-i-331bbf44c30c, 2018.

Accessed: 2019-08-28.

B. Lapid and A. Wool.

Navigating the samsung trustzone and cache-attacks on the keymaster trustlet.

In J. López, J. Zhou, and M. Soriano, editors, *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I*, volume 11098 of *Lecture Notes in Computer Science*, pages 175–196. Springer, 2018.

References (5)

D. Shen.

Attacking your Trusted Core.

https://www.blackhat.com/docs/us-15/materials/us-15-Shen-Attacking-Your-Trusted-Core-Exploiting-Trustzone-On-Android.pdf, 2015.

Accessed: 2019-11-28.

N. Stephens.

Behind the pwn of a trustzone.

https://www.slideshare.net/GeekPwnKeen/nick-stephenshow-does-someone-unlock-your-phone-with-nose, 2017.

Accessed: 2019-08-28.