Software Attacks on Hardware Wallets

riscure

Introduction

- Security Analyst at Riscure, Netherlands
- Focused on SW and HW device security
  - Smart cards
  - TEE
  - Secure devices
  - etc.

What is hardware wallet?

A device:

- Connects to smartphone, PC
- Stores and operates private keys
- Mainly used for cryptocurrency private keys
- Totally secure

https://www.ledgerwallet.com/products/ledger-nano-s

https://www.keepkey.com/wp-content/uploads/2014/08/12121301/shapeshift-large.jpg

https://trezor.io/start/

What is hardware wallet?

- Connects to smartphone, PC
- Stores and operates private keys
- Mostly used for cryptocurrency private keys
- **Totally secure?**

https://www.ledgerwallet.com/products/ledger-nano-s

https://www.keepkey.com/wp-content/uploads/2014/08/12121301/shapeshift-large.jpg

https://trezor.io/start/

Commits on Jul 28, 2016

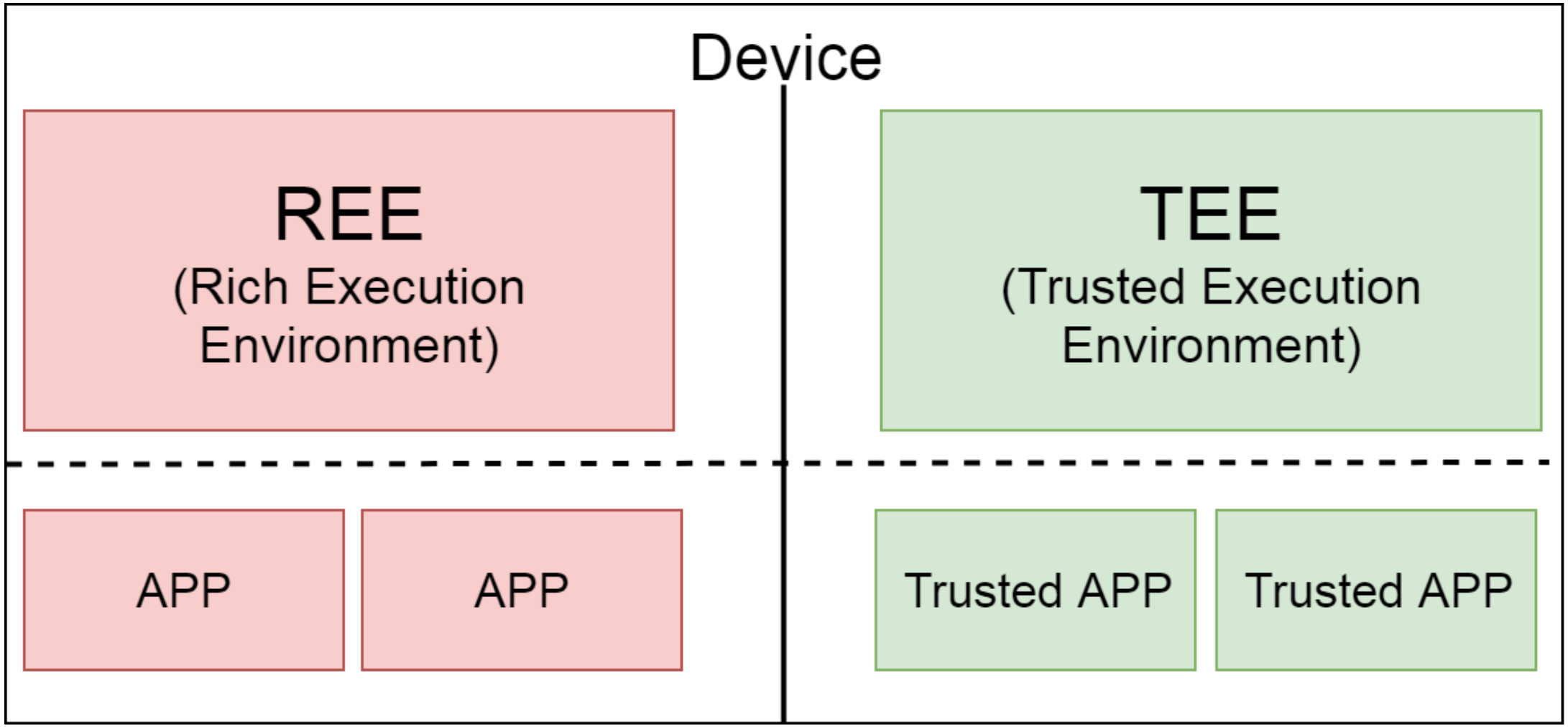
Initial commit

btchip committed on Jul 28, 2016

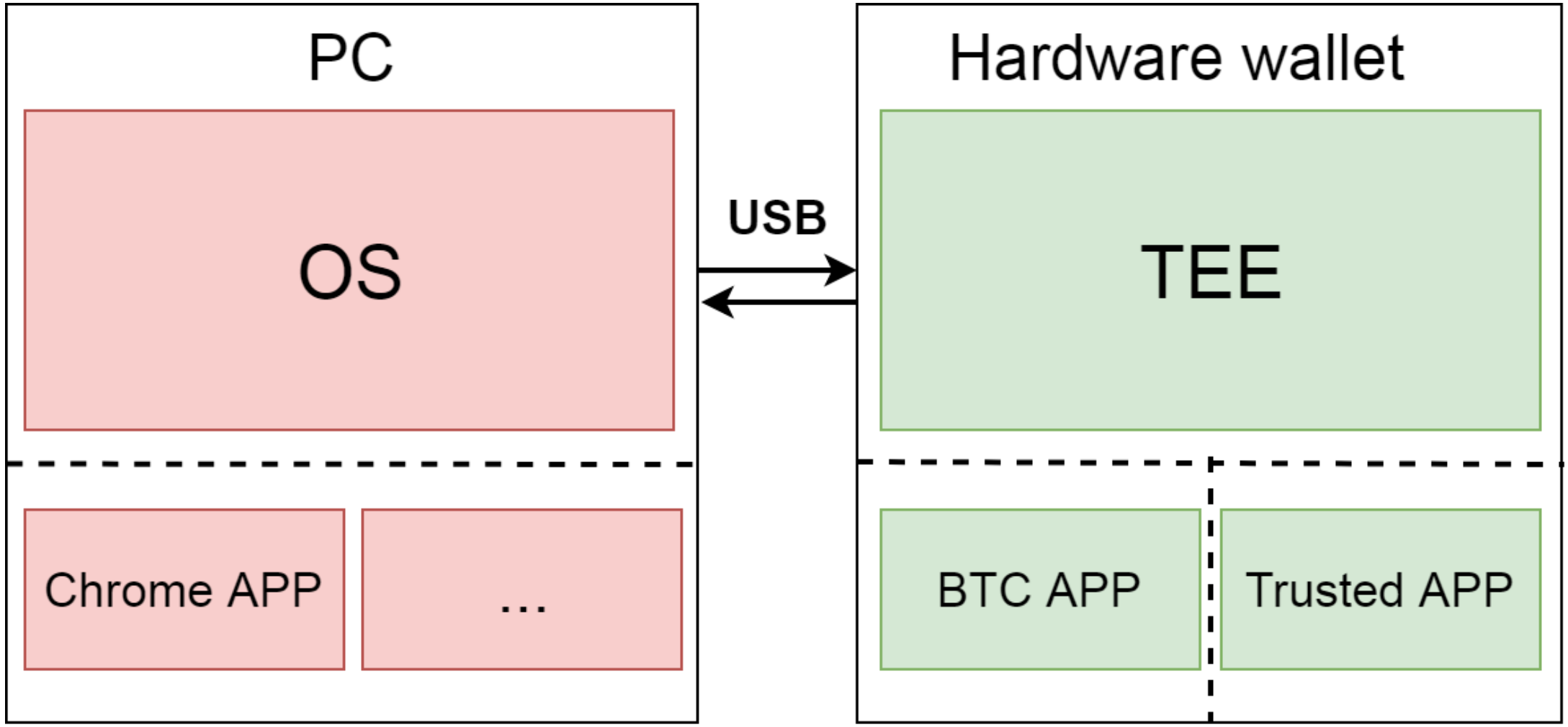
Why Ledger?

- New on the market
- Based on a certified Secure Element
- Proprietary TEE
- Multi-app support (Btc, Eth, …)
- **Support for custom applications in TEE**

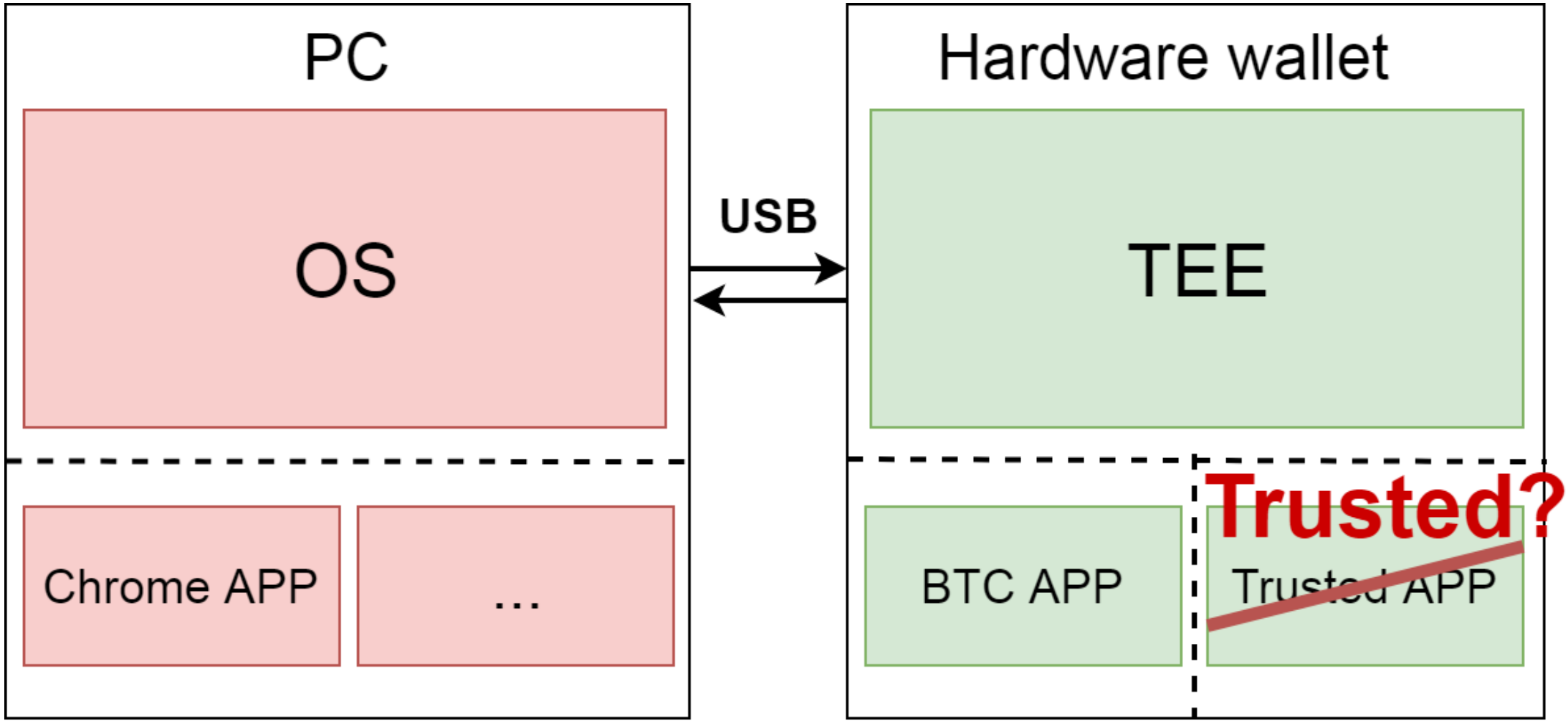
What is TEE?


Device
REE
(Rich Execution Environment)
TEE
(Trusted Execution Environment)
APP
APP
Trusted APP
Trusted APP

Why TEE?



Why TEE?


PC
OS
Chrome APP
...
USB
Hardware wallet
TEE
BTC APP
Trusted APP
Trusted?

Who is the attacker?

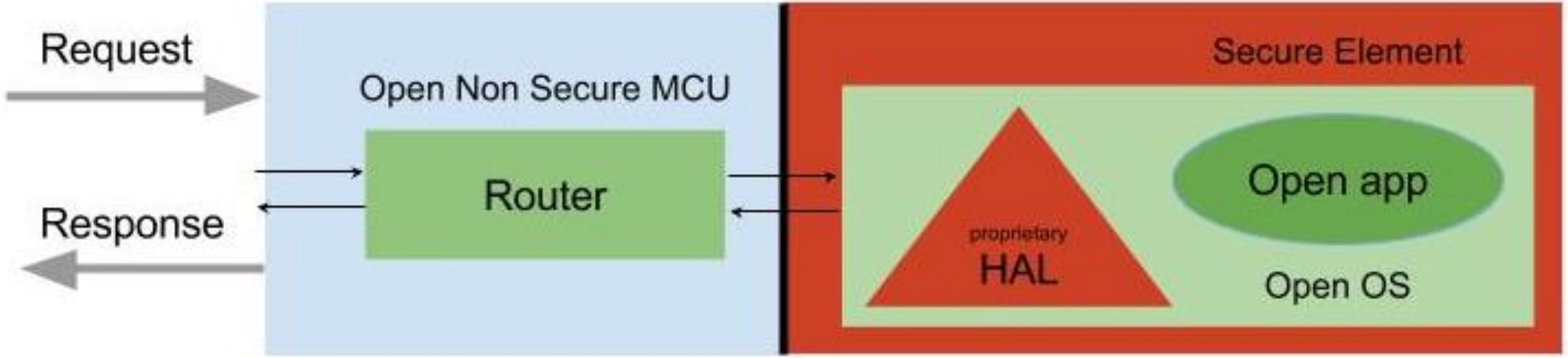
- Malware wallet (TA – TA, TA - TEE)
- Attacker with physical access to initialized device
- Attacker with access to the device before user initialization
  - Tampering before delivery
  - Second-hand devices*

Why anti-tampering?

"There is absolutely no way that an attacker could replace the firmware and make it pass attestation, without knowing the Ledger private key." (Manufacturer's blog)

What is the hardware?

https://www.ledger.fr/2016/06/09/secure-hardware-and-open-source/

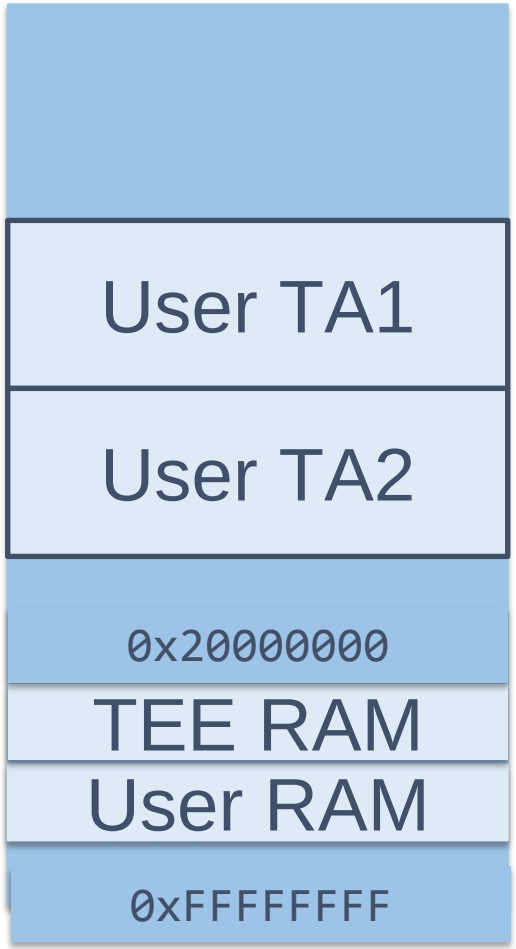


- STM32 MCU
  - Screen, buttons, USB…
  - Communication with the SE
- ST32 SE
  - BOLOS TEE
  - Wallets (Trusted Apps)
  - Secure Flash Memory

What is the BOLOS TEE?

- MPU ensures memory isolation
- Application has ~16 KB of Flash and ~1 KB of RAM
- Over 100 syscalls

```
SYSCALL void nvm_write(
        void WIDE *dst_adr PLENGTH(src_len),
        void WIDE *src_adr PLENGTH(src_len),
        unsigned int src_len);
```



V1. Dereferencing the null pointer

- Dereferencing memory outside user region mutes the device
- Any pointer outside the user region in a syscall returns SW6404
- … except null pointer
- Calling sha256() syscall with any length returns a hash
- Repeat 8k times, get 8k hashes, compute 256*8000, dump 8kB data

PTR	LEN	OUTPUT – HASH SHA256
0x00000000	0x00000001	6e340b9cffb37a989ca544e6bb780a2c78901d3fb33738768511a30617afa01d
0x00000000	0x00000002	2ee788372518190a6ab539cbb20331df1040f21846e3ba836c269aee907c894c
0x00000000	0x00000003	df236376becfe951a5a3dfa7c274ed26a75f1ccba7cf432772a9cc349017eaac

V3. Partial memory disclosure in cx_hash()

- cx_hash() syscall takes a pointer to the hash context
- Specs say the pointer must be to RAM

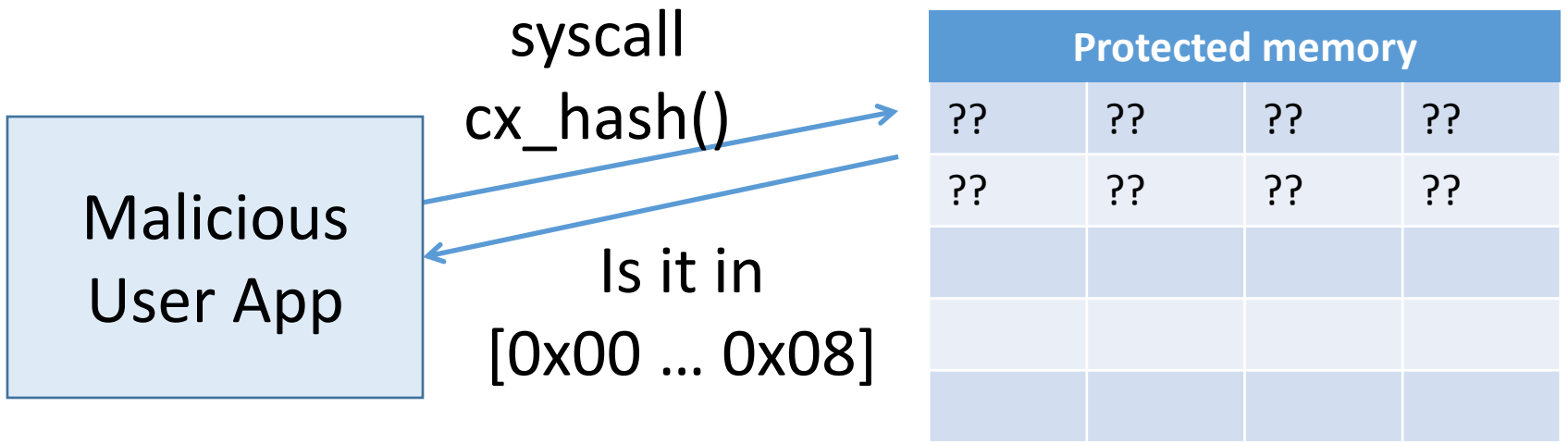
```
SYSCALL int cx_hash(
        cx_hash_t *hash PLENGTH(scc__cx_hash_ctx_size__hash),
        int mode, unsigned char WIDE *in PLENGTH(len),
        unsigned int len,
        unsigned char *out PLENGTH(scc__cx_hash_size__hash));
```

- Given different addresses there are different exceptions:
  - SW6404 – Security condition not satisfied
  - SW6402 – Condition of use not satisfied

V3. Partial memory disclosure in cx_hash()

- The hash context starts with header
- There are eight supported algorithms
  - cx_md_t [0x00 … 0x08]
- Given any memory address:

```
struct cx_hash_header_s {
        cx_md_t algo;
        unsigned int counter;
};
```



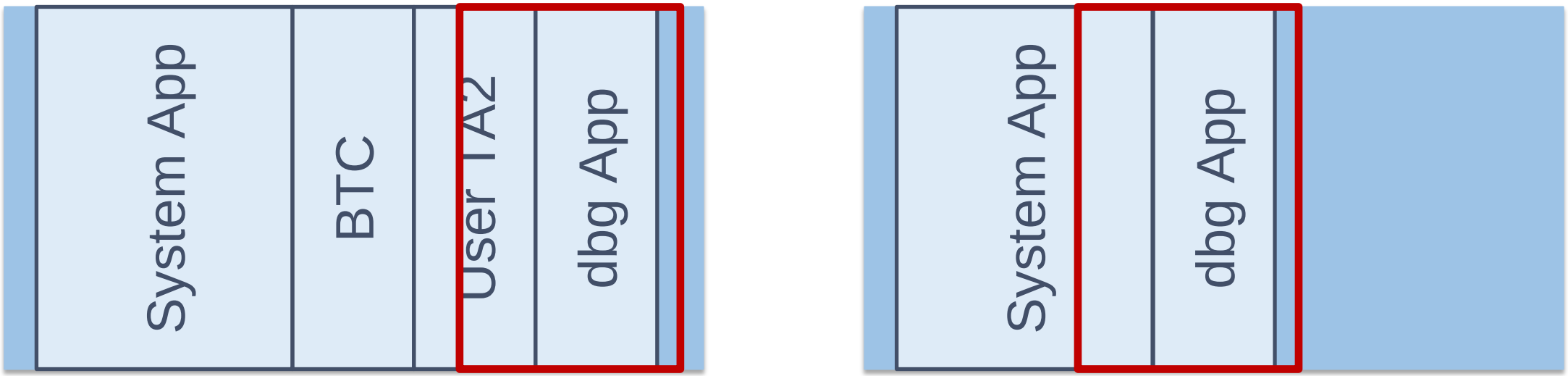
V5. Debug app installation flag

- There are a number of installation parameters of an application

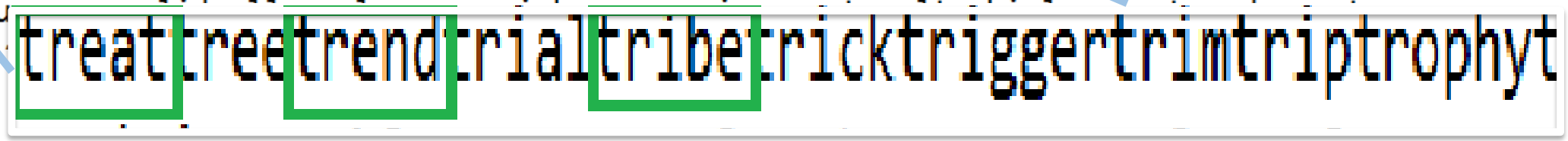
```
// This flag means the application is meant to be debugged and allows for dump
// or core ARM register in
// case of a fault detection
#define APPLICATION_FLAG_DEBUG 0x80
```

- Debug application can dump 0x4000 bytes of flash
- The memory region might include other applications

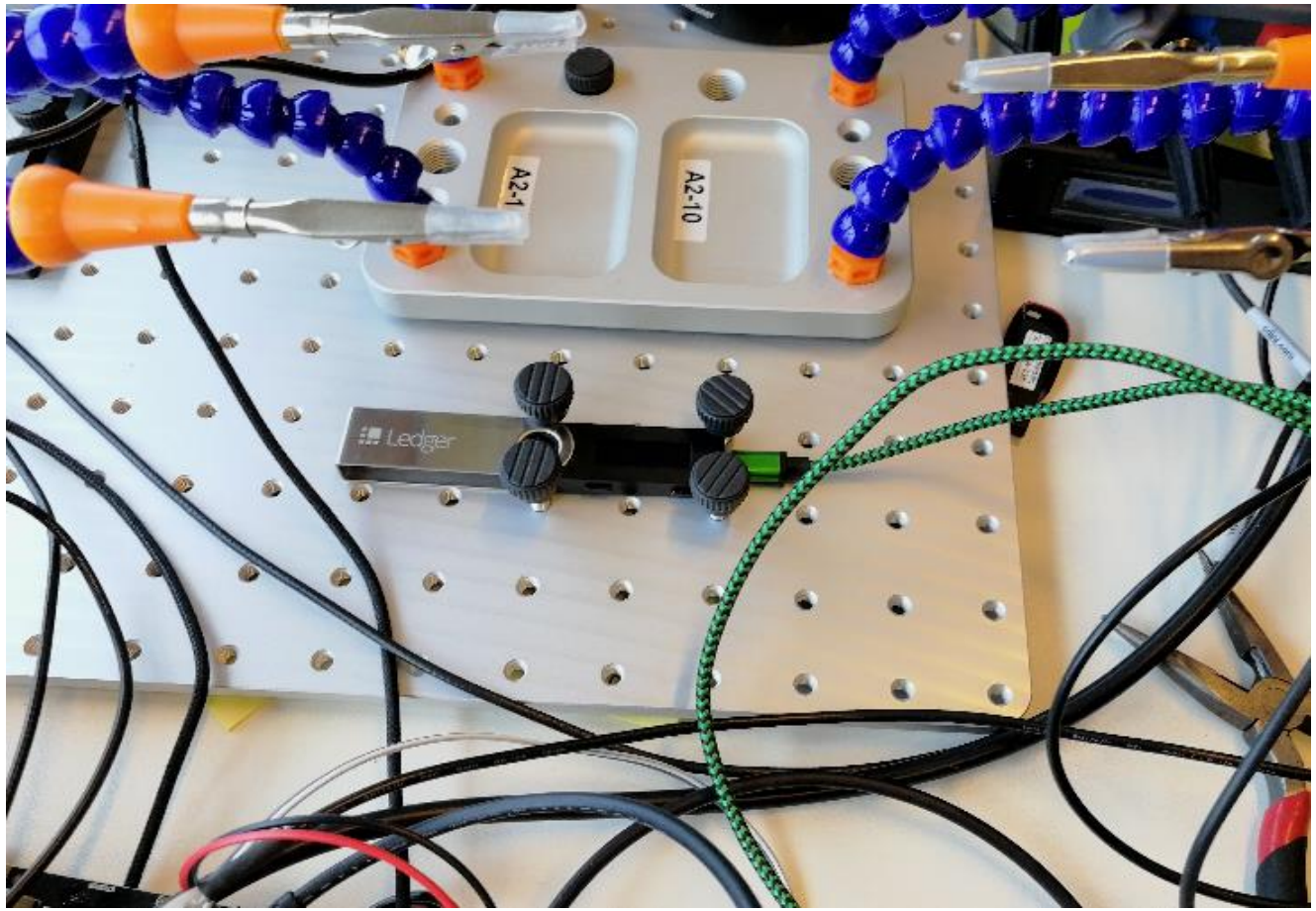
V5. Debug app installation flag



raytreattreetrendtrialtribetricktriggertrimtriptrophytroubletrucktruetrulytrumpettrusttruthtry
tubetuitiontumbletunatunnelturkeyturnturtletwelvetwentytwicetwintwisttwotypetypicaluglyumbrell
aunableunawareuncleuncoverunderundounfairunfoldunhappyuniformuniqueunituniverseunknownunlockun
tilunusualunveilupdateupgradeupholduponupperupseturbanurgeusageuseuseduselessusualutilit
yvacantvacuu



DEMO



V6. Flash is not cleared upon device reset

- The user application are not removed after device wipe
- … how about private keys?
- The wallet keys are meant to be derived from the seed at runtime

V6. Flash is not cleared upon device reset

LedgerHQ/blue-app-monero – monero_key.c

Showing the top three matches Last indexed 7 days ago

```
43      if (os_memcmp(pub, raw, 32)) {
44        THROW(SW_WRONG_DATA);
45        return SW_WRONG_DATA;
46      }
47      nvm_write(N_monero_pstate->a, sec, 32);
48      }
```

LedgerHQ/blue-app-openpgp-card – gpg_gen.c

Showing the top two matches Last indexed on Jun 12, 2017

```
153           cx_math_next_prime(pq+size,size);
154         }
155
156
157         cx_rsa_generate_pair(ksz, rsa_pub, rsa_priv, N_gpg_pstate->default
158
159         nvm_write(pkey, rsa_priv, pkey_size);
160         nvm_write(&keygpg->pub_key.rsa[0], rsa_pub->e, 4);
161         if (reset_cnt) {
```

V6. Flash is not cleared upon device reset

LedgerHQ/blue-app-btc – btchip_apdu_setup.c

Showing the top two matches Last indexed 3 days ago

```
43      // os_memmove(config.shortCoinId, PIC(G_coin_config->name_short),
44      // config.shortCoinIdLength);
45      nvm_write((void *)&N_btchip.bkp.config, &config, sizeof(config));
46      cx_rng(tmp, sizeof(tmp));
47      nvm_write((void *)&N_btchip.bkp.trustedinput_key, tmp, sizeof(tmp));
```

LedgerHQ/blue-app-u2f – u2f_config.c

Showing the top two matches Last indexed on Jan 22, 2017

```
38          os_perso_derive_node_bip32(CX_CURVE_256R1, keyPath, 1,
39                                     u2fConfig.hmacKey, u2fConfig.hmacKey + 32);
40  #endif
41          nvm_write(&N_u2f, &u2fConfig, sizeof(u2f_config_t));
…
53          if (os_memcmp(u2fConfig.hmacKey, N_u2f.hmacKey,
54                        sizeof(u2fConfig.hmacKey)) != 0) {
55              nvm_write(N_u2f.hmacKey, u2fConfig.hmacKey,
```

Success!

V6. Flash is not cleared upon device reset

```
0000000000000000000000000000000000000000000000000000000000000000
ffb18dac0100000001295ebe8b1fa7e86f6b9a3683b521fa71aa64b427471c5d
89995bd1e198b6d6a655dcdc64289b8b537fe351cf41898992a108928c370f4d
294add1edd696fe8418f1200000000000000000000000000000000000000000
```

```
0000000000000000000000000000000000000000000000000000000000000000
ffb18dac0200000001295ebe8b1fa7e86f6b9a3683b521fa71aa64b427471c5d
89995bd1e198b6d6a655dcdc64289b8b537fe351cf41898992a108928c370f4d
294add1edd696fe8418f1200000000000000000000000000000000000000000
```

uid counter init_flag padding hmacKey

Conclusions?

- There are 7 exploitable vulnerabilities identified in the wallet
- An attacker with physical access could compromise the isolation
- The private keys stored in any app were vulnerable
- Defense in depth helped runtime generated keys to be protected

What do we learn?

- To be secure devices require both protected HW and SW
- TEE should not expect any assumptions are met
- All the syscall parameters need to be restricted and checked
- Large API is a big attack surface
- Lifecycle state of HW wallets is prone to issues by design
- Third party evaluation is a must for a secure solution

Q&A

Sergei Volokitin

volokitin@riscure.com

riscure