Exploiting Hardware Wallet's Secure Element

Sergei Volokitin
What is a hardware wallet?

• Connects to smartphone / PC
• Stores and uses private keys
• Mainly used for cryptocurrency private keys
• Super secure
What is a hardware wallet?

- Connects to smartphone / PC
- Stores and uses private keys
- Mainly used for cryptocurrency private keys
- Super secure?

https://www.ledgerwallet.com/products/ledger-nano-s
https://trezor.io/start/
https://bitfi.com/bitfi-wallet
Who is the attacker?

Physical access

Malware on PC

Supply chain
What are the defenses?

• Secrets don’t leave device
• Tamper-resistant
• Keep secrets in a Secure Element
• Secure software updates
• Embedded private key for attestation
Ledger Wallet Bitcoin

To begin, connect and unlock your Ledger Wallet.

If your device supports multiple currencies, open the desired app.
Why Ledger?

- Fairly new/interesting
- Based on a **certified** Secure Element
- Multi-app support (Btc, Eth, ...)
- Support for custom applications on SE
Who is the attacker?

- Malware App
- Physical access
- Supply chain
Ledger Nano S hardware

- STM32 MCU
  - Screen, buttons, USB…
  - Communication with the SE

- ST31 SE
  - BOLOS
  - Wallets (Trusted Apps)
  - Secure Flash Memory
System calls

- MPU isolates memory
- Application has
  - ~16 KB of Flash
  - ~1 KB of RAM
- Over 100 syscalls
sha256() syscall
**sha256() syscall**

First 8 kB of the protected flash memory can be extracted!
Partial memory disclosure in cx_hash()

- cx_hash() syscall takes a pointer to a context structure
- Only values from 0 to 8 are valid algorithms!

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

```c
if (!known_algo(context->algo))
    return [invalid hash algorithm];
if (!access_allowed(context, needed_len(context)))
    return [security error];
```

You can tell whether ANY byte in memory is between 0-8 .. or between 9-255.
Memory disclosure in `cx_ecdsa_sign()`
cx_ecdsa_sign()
cx_ecdsa_sign()

- hash_len is used in the security check
- hashID is used to determine the length
- hash_len == 0 and hashID == CX_SHA512
  - 64 bytes of the next application in memory
- Security in depth:
  - System memory is at lower addresses
cx_ecdsa_sign()

- hash_len is used in the security check
- hashID is used to determine the length
- hash_len == 0 and hashID == CX_SHA512
- 64 bytes of the next application in memory

Security in depth:
- System memory is at lower addresses
Debug app installation flag

- There are per-application flags you can set, such as:

```cpp
// 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
```

- Application with debug flag can read ~16kB of flash belonging to another app!
# Debug app installation flag

<table>
<thead>
<tr>
<th>System Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTC App Flash</td>
</tr>
<tr>
<td>U2F App Flash</td>
</tr>
<tr>
<td>Debug App Flash</td>
</tr>
</tbody>
</table>
Debug app installation flag

- System Flash
- BTC App Flash
- U2F App Flash
- Debug App Flash
Debug app installation flag

- System Flash
- BTC App Flash
- Debug App Flash
Debug app installation flag

- System Flash
- Debug App Flash
Debug app installation flag

System Flash

Debug App Flash
Debug app installation flag
Device “Wipe”
Flash is not cleared upon device reset

• User applications are not removed after device wipe…

• Good news: the global seed is reset!

• Wallet keys should be derived from seed at **runtime**
  • *most* apps (e.g. Btc) do this…
Secret keys in flash

```c
LedgerHQ/blue-app-btc – btchip_apdu_setup.c

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));
```

```c
LedgerHQ/blue-app-monero – monero_key.c

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

```c
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) {
```
U2F app

```c
os_perso_derive_node_bip32(CX_CURVE_256R1, keyPath, 1,
    u2fConfig.hmacKey, u2fConfig.hmacKey + 32);

#ifdef

nvm_write(&N_u2f, &u2fConfig, sizeof(u2f_config_t));

... 

if (os_memcmp(u2fConfig.hmacKey, N_u2f.hmacKey,
    sizeof(u2fConfig.hmacKey)) != 0) {

    nvm_write(N_u2f.hmacKey, u2fConfig.hmacKey,
```
<table>
<thead>
<tr>
<th>uid</th>
<th>counter</th>
<th>init_flag</th>
<th>padding</th>
<th>hmacKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>ffb18dac0100000001295ebe8b1fa7e86f6b9a3683b521fa71aa64b427471c5d 89995bd1e198b6d6a655dcdc64289b8b537fe351cf41898992a108928c370f4d 294add1edd696fe8418f1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ffb18dac0100000001295ebe8b1fa7e86f6b9a3683b521fa71aa64b427471c5d 89995bd1e198b6d6a655dcdc64289b8b537fe351cf41898992a108928c370f4d 294add1edd696fe8418f1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Supply chain attack

• The issue:
  • The applets are not removed from flash upon wipe
  • CustomCA keys are not removed as well
  • An application signed with CustomCA keys does not show security warning
Supply chain attack (tamper-evident)
Supply chain attack

CustomCA keys

Malicious BTC app + CustomCA signature

“Wipe”
Summary

• Several vulnerabilities identified in Ledger Nano S
  • Most severe consequence: U2F keys were not safe

• Ledger fixed all these vulnerabilities in a timely manner
  • Bounty programs work!

• Closed source SE firmware => No community code reviews
What did we learn?

• Don’t trust Trusted Applications

• Hardware wallets still have software flaws

• Reset means reset – clear all that data!
Challenge your security