Revealing *One Key* Security Insight for Hardware Wallets

nwmonster & slipper
Offside Labs

OFFSIDE

Offside Labs

- Background
  - Elite hackers from web2 to web3
- Achievements
  - Pioneered jailbreaks for iPhone & PlayStation
  - Dominated top-tier hacking competitions:
    - Pwn2Own, Tianfu Cup, DEF CON CTF, Paradigm CTF
- Impact
  - Secured assets worth **$300M+**
  - Won **$9M+** in bug bounties
  - Led 2022's blockchain security breakthroughs

OFFSIDE

Hardware Wallets

Purpose of a Hardware Wallet

A hardware wallet is a tool for safeguarding digital currencies.

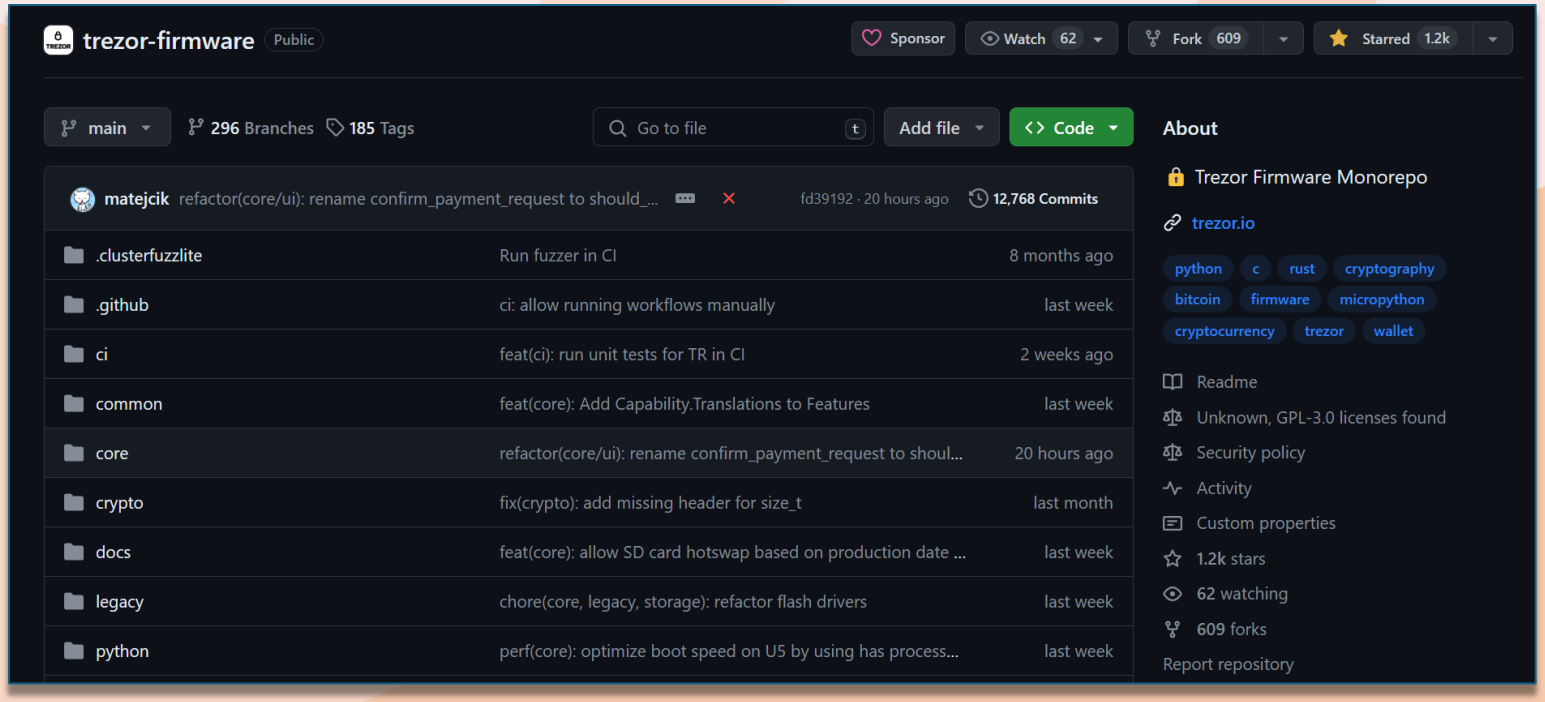
It reduces the risk of online threats and keeps your assets secure.

Use it to store your private keys offline and protect your cryptos.

OFFSIDE

Trezor

In 2012, our visionary founders created the first hardware wallet. Over the past decade, relentless innovation and dedication have driven us to incredible heights, now serving over 1 million users worldwide. Our success is powered by a team of over 150 exceptional employees.



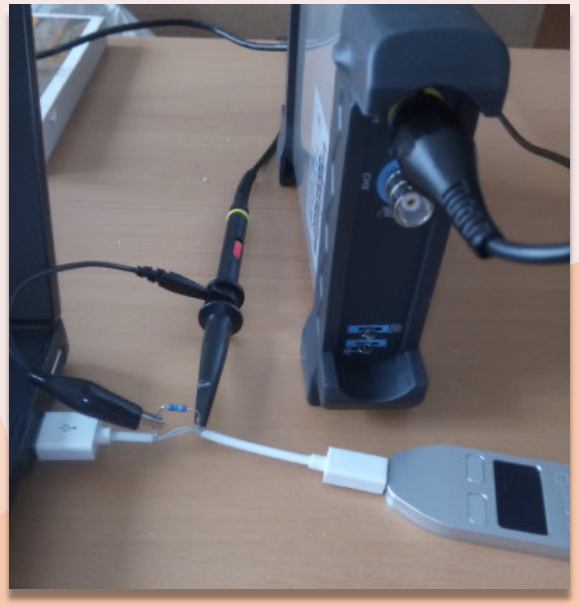
https://github.com/trezor/trezor-firmware

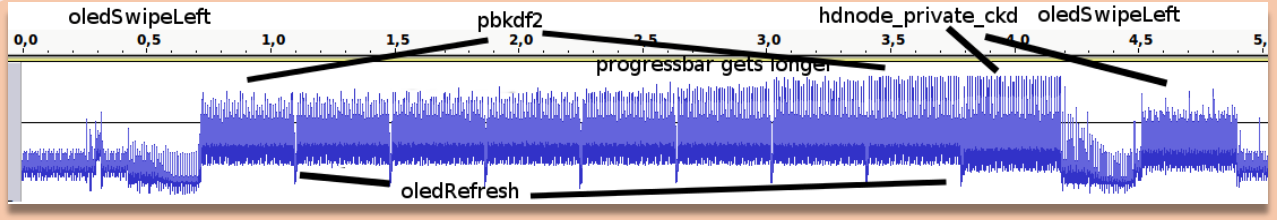
Hardware Wallets Security

Extracting the Private Key from a TREZOR

... with a 70 $ Oscilloscope

When the TREZOR is detected the PC will ask the TREZOR for the public key. If the TREZOR is not passphrase protected, it wakes up, computes the master private key from the seed and then the public key from the private key.





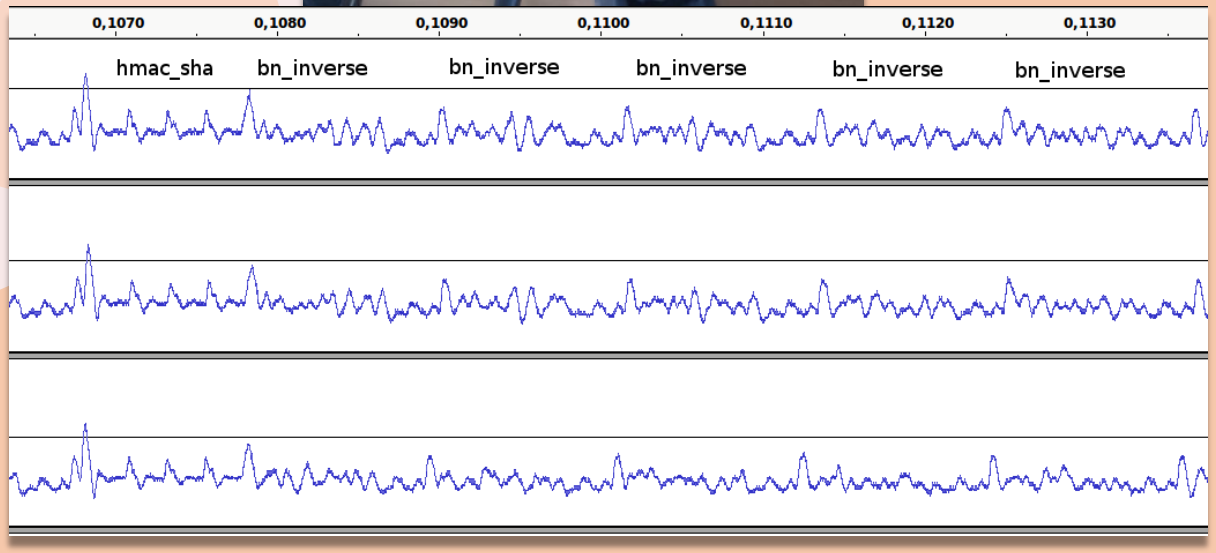
https://jochen-hoenicke.de/crypto/trezor-power-analysis/

Hardware Wallets Security

Extracting the Private Key from a TREZOR

... with a 70 $ Oscilloscope

I have recovered 128 bits of a private key as a proof of concept. It took me about two hours and it would take the same time to recover the second half. The main problem is that one needs a reference TREZOR and use it to retrieve the necessary fingerprints of the bn_inverse functions. This work has to be repeated for every private key.

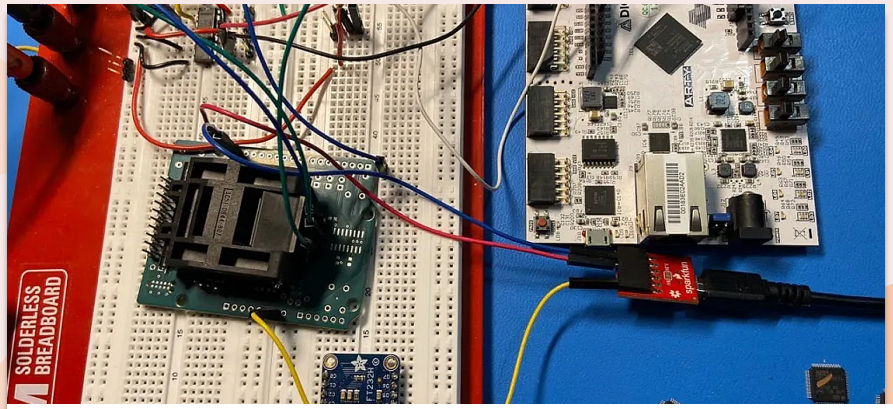


https://jochen-hoenicke.de/crypto/trezor-power-analysis/

Hardware Wallets Security

Kraken Identifies Critical Flaw in Trezor Hardware Wallets

The STM32s used in wallets like the Trezor One are set to RDP Level 2 at manufacturing time. This deativates all debugging features and disables the integrated BootROM bootloader. With voltage glitching it is possible to corrupt the RDP value being read from the Option Bytes, as shown in the Wallet.Fail research.



RDP Level	Option Byte Value	Behavior
0	0xAA	Full access to SRAM and Flash
1	Any value except 0xAA and 0xCC	Read access to SRAM, no access to Flash
2	0xCC	No access to SRAM or Flash

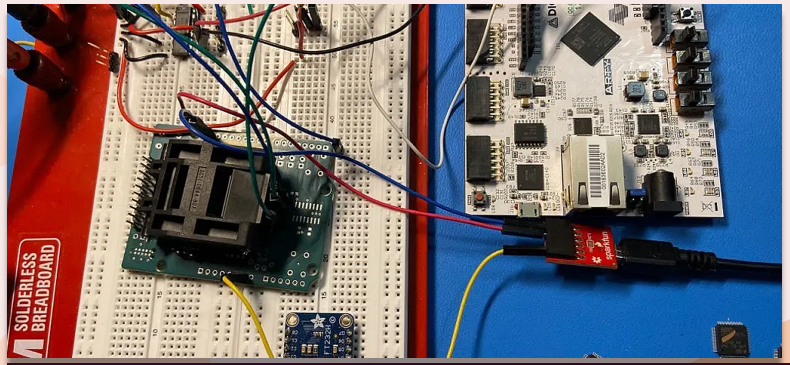
Command(1)	Command code	Command description
Get(2)	0x00	Gets the version and the allowed commands supported by the current version of the bootloader.
Get Version & Read Protection Status(2)	0x01	Gets the bootloader version and the Read Protection status of the Flash memory.
Get ID(2)	0x02	Gets the chip ID.
Read Memory(3)	0x11	Reads up to 256 bytes of memory starting from an address specified by the application.
Go(3)	0x21	Jumps to user application code located in the internal Flash memory or in the SRAM.
Write Memory(3)	0x31	Writes up to 256 bytes to the RAM or Flash memory starting from an address specified by the application.
Erase(3)(4)	0x43	Erases from one to all the Flash memory pages.

https://blog.kraken.com/product/security/kraken-identifies-critical-flaw-in-trezor-hardware-wallets
https://media.ccc.de/v/35c3-9563-wallet_fail

Hardware Wallets Security

Kraken Identifies Critical Flaw in Trezor Hardware Wallets

Once the salt, EDEK(Encrypted Data Encryption Key), ESAK(Encrypted Storage Authentication Key) and PVC (PIN verification code)have been retrieved, an additional hardware salt needs to be retrieved from the OTP bytes. This can be done by re-enabling the SWD interface or by re-enabling the Bootrom bootloader and issuing the corresponding Read Memory Commands.



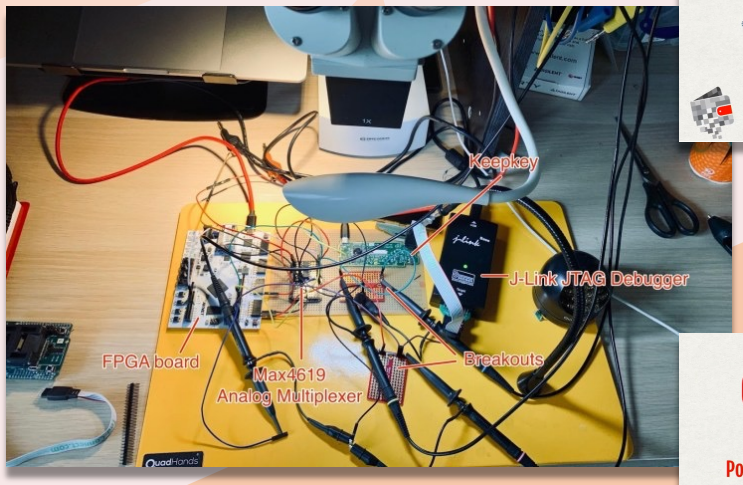

```
→ trezorcracker git:(master) ✗ time ./trezorcracker.py example1.img
Hardware salt:  b'85759b3811ff7dc47b03792ac85317be51431a3f9e01dcafce317ed736a391b0'
Assembled salt: b'85759b3811ff7dc47b03792ac85317be51431a3f9e01dcafce317ed736a391b00e53c68f'
Currently trying: 100
Currently trying: 200
Currently trying: 300
Currently trying: 400
Currently trying: 500
Currently trying: 600
Currently trying: 700
Currently trying: 800
Currently trying: 900
Currently trying: 1000
Currently trying: 1100
Currently trying: 1200
SUCCESS
PIN is:
1234
./trezorcracker.py example1.img  11,60s user 0,03s system 100% cpu 11,629 total
→ trezorcracker git:(master) ✗
```

https://blog.kraken.com/product/security/kraken-identifies-critical-flaw-in-trezor-hardware-wallets

Hardware Wallets Security

Inside Kraken Security Labs: Flaw Found in Keepkey Crypto Hardware Wallet

Most importantly, this research demonstrates that the security of a wallet like the KeepKey should not solely be based on the security of the STM32F205 microcontroller.

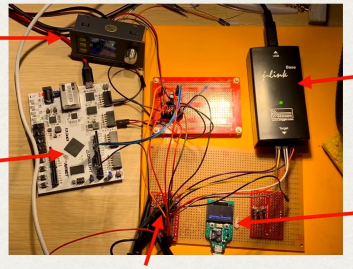


Prior STM32 Security Research

- STM32F0: JTAG bug + RDP2 -> RDP1 Downgrade
  "*Shedding too much Light on a Microcontroller's Firmware Protection*"
  https://www.usenix.org/system/files/conference/woot17/woot17-paper-obermaier.pdf
- STM32F1/STM32F3: F1 Bootloader bypass + RDP2 -> RDP1 Downgrade
  TBA - follow us on twitter for a link to this paper when it get's published
- How to apply it all? Read:
  "*Verifying Code Readout Protection Claims*"
  http://circuitcellar.com/cc-blog/verifying-code-readout-protection-claims/

WALLET.FAIL

Glitching the Trezor One



WALLET.FAIL

https://blog.kraken.com/product/security/flaw-found-in-keepkey-crypto-hardware-wallet
https://blog.kraken.com/product/security/flaw-found-in-keepkey-crypto-hardware-wallet-part-2
https://media.ccc.de/v/35c3-9563-wallet_fail
https://www.blackhat.com/us-19/briefings/schedule/#chipfail---glitching-the-silicon-of-the-connected-world-15192

Hardware Wallets Security

Extracting TREZOR Secrets from SRAM

Now that the exploit has been discovered, what you need to know is:
Firmware 1.5.2 is not vulnerable to this exploit or any variation of it
Physical access is required
If you are using a BIP-39 passphrase, you are safe (the mnemonic is useless without the passphrase and the TREZOR does not store the passphrase)
https://saleemrashid.com/2017/08/17/extracting-trezor-secrets-sram/

Hacking the ultra-secure hardware cryptowallet

Two sets of vulnerabilities could lead to theft of secrets if an attacker has physical access to a device, or is able to compromise a supply chain. Additional group of vulnerabilities does not lead to an immediate compromise of protected data, but reveals weaknesses in the device's security design that can potentially be exploited further. Ledger, the maker of the cryptowallet, was informed about the findings and addressed the vulnerabilities.
https://www.riscure.com/hacking-ultra-secure-hardware-cryptowallet/

Hardware Wallets Security

KeepKey Stack Buffer Overflow Vulnerability (CVE-2021-31616)

```
// Detect THORChain swap
if (ethereum_isThorchainSwap(msg)) {
  if (token == NULL && data_total > 0 && data_needs_confirm) {
    char swap_data[256] = {'\0'};
    uint8_t swap_data_len = ethereum_extractThorchainSwapData(msg, swap_data);
```

```
@@ -93,10 +93,10 @@ uint8_t ethereum_extractThorchainData(const EthereumSignTx *msg,
   // offset = deposit function hash + address + address + uint256
   uint16_t offset = 4 + (5 * 32);
   int16_t len = msg->data_length - offset;
-  if (msg->has_data_length && len > 0) {
+  if (msg->has_data_length && len > 0 && len < 256) {
    memcpy(buffer, msg->data_initial_chunk.bytes + offset, len);
    // String length must be < 255 characters
-    return len < 256 ? (uint8_t)len : 0;
+    return (uint8_t)len;
   }
   return 0;
 }
```

```
                    PTR_PTR__stack_chk_guard_08086b68          XREF[2]:     address_write_prefix_bytes:08086...
                                                                            address_write_prefix_bytes:08086...
08086b68 3c dc 0c 08     addr       PTR__stack_chk_guard_080cdc3c            = 2001f7f8

                    **********************************************************
                    *                          FUNCTION                      *
                    **********************************************************
                    undefined address_check_prefix()
                      assume LRset = 0x0
                      assume TMode = 0x1
     undefined          r0:1            <RETURN>
     undefined4         Stack[-0xc]:4  local_c                    XREF[2]:     08086b74(W),
                                                                               08086b8a(R)
                    address_check_prefix                       XREF[5]:     Entry Point(*),
                                                                            compile_output:0806c8d2(c),
                                                                            compile_output:0806c936(c),
                                                                            tx_output_weight:0806d596(c),
                                                                            tx_output_weight:0806d5a6(c)
08086b6c 13 b5          push       { r0, r1, r4, lr }
08086b6e 15 4b          ldr        r3,[PTR_PTR__stack_chk_guard_08086bc4]   = 080cdc40
08086b70 ff 29          cmp        r1,#0xff
08086b72 1b 68          ldr        r3,[r3,#0x0]=>->__stack_chk_guard        = 2001f7f8
08086b74 01 93          str        r3=>__stack_chk_guard,[sp,#local_c]
08086b76 4f f0 00 03    mov.w      r3,#0x0
08086b7a 02 78          ldrb       r2,[r0,#0x0]
08086b7c 0a d8          bhi        LAB_08086b94
08086b7e 91 42          cmp        r1,r2

                    LAB_08086b80                               XREF[1]:     08086ba2(j)
08086b80 10 4b          ldr        r3,[PTR_PTR__stack_chk_guard_08086bc4]   = 080cdc40
08086b82 14 bf          ite        ne
08086b84 00 20          mov.ne     r0,#0x0
08086b86 01 20          mov.eq     r0,#0x1
08086b88 1a 68          ldr        r2,[r3,#0x0]=>->__stack_chk_guard        = 2001f7f8
08086b8a 01 9b          ldr        r3=>__stack_chk_guard,[sp,#local_c]
08086b8c 5a 40          eor        r2,r3
08086b8e 16 d0          beq        LAB_08086bbe
08086b90 ef f7 54 ff    bl         __stack_chk_fail                         undefined __stack_chk_fail()
                    -- Flow Override: CALL_RETURN (CALL_TERMINATOR)
```

https://blog.inhq.net/posts/keepkey-CVE-2021-31616/
https://blog.inhq.net/posts/faulty-stack-canary-arm-systems/

OneKey

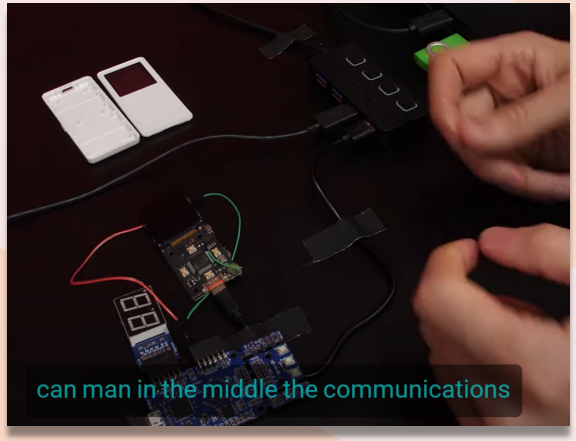
In 2019, we released our initial version of OneKey firmware, which is based on Trezor's open source work, we have been continuously extending, improving, and patching it since then. Trezor is a great forerunner, without their inspiration and open source work, we might not have been able to develop the products we have today.

src	Trezor	OneKey
\legacy	Model One Safe 3	Mini Classic
\	Model T	Touch

https://blog.onekey.so/write-original-firmware-55aedd7446c5
https://github.com/trezor/trezor-firmware
https://github.com/OneKeyHQ/onekey-firmware

Hardware Wallets Security

How We Hacked a Hardware Crypto Wallet and Saved The World



https://github.com/OneKeyHQ/firmware/releases/tag/mini%2Fv2.10.1
https://www.youtube.com/watch?v=b8OrakRJmHE

Our(OneKey) Response to Recent Security Fix Reports

We want to highlight the fact that these attacks cannot be exploited remotely. It would require disassembly of the device and physical access through a dedicated FPGA device in the lab to be possible to execute.

https://blog.onekey.so/our-response-to-recent-security-fix-reports-13914fea8afd

OneKey Mini vs Trezor Model One

OneKey Mini	Trezor Model One
$58	$59
Open Source	Open Source
1.44 Inch IPS Display (Color)	0.96 Inch OLED
USB-C	Micro USB
...	...
Secure Element protected	
Support Chinese	

legacy > firmware > font_ex.c

```c
static void font_data_read(void *buf, uint32_t offset, uint32_t len) {
  flash_read_enc(buf, FONT_DATA_ADDR + offset, len);
}

void font_init(void) {
  uint8_t end_flag[4] = {0};
  font_data_read(&font_header, 0, FONT_HEADER_LEN);

  if (memcmp(font_header.magic, "U3TP", 4) == 0) {
    font_data_read(&end_flag, font_header.total_len - 4, 4);
    if (memcmp(end_flag, "ENDU", 4) == 0) {
      has_font = true;
    }
  }
}

bool font_imported(void) {
```

Root Cause #0

Adding Chinese support requires an additional flash storage

legacy > firmware > protob > messages.proto

facotry Aa ab .* 2 of 3

```
81   enum MessageType {
495      // Nexa
503      //onekey
504      MessageType_DeviceInfoSettings = 10001 [(wire_in) = true];
505      MessageType_GetDeviceInfo = 10002 [(wire_in) = true];
506      MessageType_DeviceInfo = 10003 [(wire_out) = true];
507      MessageType_ReadSEPublicKey = 10004 [(wire_in) = true];
508      MessageType_SEPublicKey = 10005 [(wire_out) = true];
509      MessageType_WriteSEPublicCert = 10006 [(wire_in) = true];
510      MessageType_ReadSEPublicCert = 10007 [(wire_in) = true];
511      MessageType_SEPublicCert = 10008 [(wire_out) = true];
512      MessageType_SpiFlashWrite = 10009 [(wire_in) = true, (facotry) = true];
513      MessageType_SpiFlashRead = 10010 [(wire_in) = true, (facotry) = true];
514      MessageType_SpiFlashData = 10011 [(wire_out) = true];
515      MessageType_SESignMessage = 10012 [(wire_in) = true];
516      MessageType_SEMessageSignature = 10013 [(wire_out) = true];
517      MessageType_NFTWriteInfo = 10014 [(wire_in) = true];
520  }
```

Root Cause #0

The font file needs to be installed before leaving the factory.

Root Cause #0*

facotry

files to include

legacy

files to exclude

12 results in 6 files - Open in editor

messages-management.proto legacy/bootloader/

...initstates = 513; // list of facotry state

messages.proto legacy/bootloader/protob

optional bool facotry = 50501;

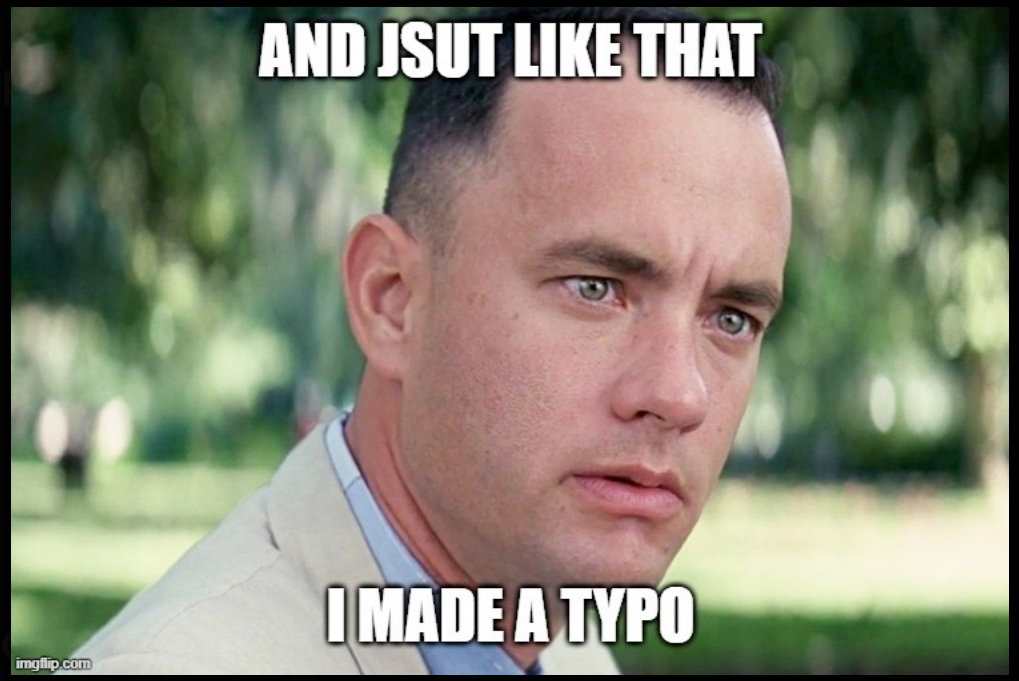
...10009 [(wire_in) = true, (facotry) = true];

...10010 [(wire_in) = true, (facotry) = true];

messages-management.proto legacy/firmware/pr

...initstates = 513; // list of facotry state

mini 0 0 0



legacy > firmware > C fsm_msg_common.h

```
1132  void fsm_msgSpiFlashRead(const SpiFlashRead *msg) {
1133    (void)msg;
1134  #if ONEKEY_MINI
1135    RESP_INIT(SpiFlashData);
1136
1137    if (flash_read_enc(resp->data.bytes, msg->address, msg->len)) {
1138      resp->data.size = msg->len;
1139      msg_write(MessageType_MessageType_SpiFlashData, resp);
1140    } else {
1141      fsm_sendFailure(FailureType_Failure_ProcessError, _("Read faile
1142    }
1143  #else
1144    fsm_sendFailure(FailureType_Failure_UnexpectedMessage, _("Unknown
1145  #endif
1147  }
```

Root Cause #1

Lack of length validation for *msg->len*

legacy > firmware > protob > messages_map.py

```python
44  def handle_message(fh, fl, skipped, message, extension):
69      if direction == "i":
70          process_func = f"(void (*)(const void *))fsm_msg{short_name}"
71      else:
72          process_func = "0"
73
74      cmd_flags = "CommandFlags_"
75      if getattr(options, "factory", None):
76          cmd_flags += "Factory_Only"
77      else:
78          cmd_flags += "Default"
79
80      fh.write(
81          TEMPLATE.format(
82              type=f"'{interface}',",
83              dir=f"'{direction}',",
84              cmd_flags="%s," % cmd_flags,
```

Root Cause #2

Factory_Only is never enforced

legacy > firmware > C fsm_msg_common.h

```
1117  void fsm_msgSpiFlashWrite(const Sp
1118      (void)msg;
1119  #if ONEKEY_MINI
1120      if (flash_write_enc((uint8_t *)ms                    >add
1121                                  msg->data.si
1122          fsm_sendSuccess(_("Write succe
1123      } else {
1124          fsm_sendFailure(FailureType_Fa                    _("W
1125      }
1126  #else
1127      fsm_sendFailure(FailureTyp ailure_UnexpectedMessage, _
1128  #endif
1130  }
```

Exploitation 20% 🤔

Control data by *msgSpiFlashWrite*

```
                    uint8_t_ARRAY_20009dac
                    uint8_t_ARRAY_20009db0
                    uint8_t_ARRAY_20009db4
                    uint8_t_ARRAY_20009db8
                    msg_resp
20009d58                uint8_t[... ??

                    unlock_path
```

msg_resp@20009d58

```
2000a958                uint32_t   ??

                    backup.1
                    enter_sleep::backup

2000a95c                _Bool      ??

                    device_sleep_state
```

```
2001659c                TimerDse... ??
  2001659c                TimerDsec
```

timer_array@2001659c

```
2001659c                char[32]

  2001659c [0]              ??,  ??,  ??,  ??,
  200165a0 [4]              ??,  ??,  ??,  ??,
  200165a4 [8]              ??,  ??,  ??,  ??,
  200165a8 [12]             ??,  ??,  ??,  ??,
  200165ac [16]             ??,  ??,  ??,  ??,
  200165b0 [20]             ??,  ??,  ??,  ??,
                            ??,  ??,  ??,  ??,
                            ??,  ??,  ??,  ??
200165bc                uint32_t
```

Exploitation 50% 😊

Control overflow to global variable *timer_array*

```
timer_array[0].current
timer_array[0].cycle
```

legacy > C timer.c

```c
#define TIMER_NUM 4
typedef struct {
```

legacy > C timer.c

```c
void sys_tick_handler(void) {
    int i;
    system_millis++;
    system_millis_sleep_start++;
    timer_out_decrease();
    for (i = 0; i < TIMER_NUM; i++) {
        if (timer_array[i].fp) {
            if ((system_millis - timer_array[i].current) > timer_array[i].cycle)
                timer_array[i].current = system_millis;
                timer_array[i].fp();
            }
        }
    }
}
```

Exploitation 80% 😜

```
2001659c        char[32]        ??        name
```

Hijacking the *fp* in *TimerDsec*

```
2001659c [0]        ??,  ??,  ??,  ??,
```

timer_array[0].current
timer_array[0].cycle

legacy > C timer.c

```c
#define TIMER_NUM 4
typedef struct {
```

legacy > C timer.c

```c
void sys_tick_handler(void) {
  int i;
  system_millis++;
  system_millis_sleep_start++;
  timer_out_decrease();
  for (i = 0; i < TIMER_NUM; i++) {
    if (timer_array[i].fp) {
      if ((system_millis - timer_array[i].current) > timer_array[i].cycle)
        timer_array[i].current = system_millis;
        timer_array[i].fp();
      }
    }
  }
}
```

~~Exploitation 80%~~ 🥴

2001659c char[32] ?? name

Random Crashes

2001659c [0] ??, ??, ??, ??,

legacy > firmware > C flash_enc.c

```c
59  bool flash_read_enc(uint8_t *buffer, uint32_t address, uint32_t
64    if (w25qxx_read_bytes(buffer, address, len)) {
67      } else {
88        if (len) {
89          w25qxx_read_bytes(buffer + pre_remain, address + pre_rema
90          aes_ecb_decrypt(buffer + pre_remain, buffer + pre_remain,
91                          &dec_ctx);
92        }
93        if (next_remain) {
94          w25qxx_read_bytes(dec_buffer, address + pre_remain + len
95                            AES_BLO
96          aes_ecb_decrypt(dec_buffer, dec_buffer, AES_BLOCK_SIZE, &
97          memcpy(buffer + pre_remain + len, dec_buffer, next_remai
98        }
99      }
```

Source of Chaos

aes_ecb_decrypt is too slow!

legacy > firmware > C flash_enc.c

```c
59  bool flash_read_enc(uint8_t *buffer, uint32_t address, uint32_t
64    if (w25qxx_read_bytes(buffer, address, len)) {
67      } else {
88        if (len) {
89          w25qxx_read_bytes(buffer + pre_remain, address + pre_rema
90          aes_ecb_decrypt(buffer + pre_remain, buffer + pre_remain,
91                          &dec_ctx);
92        }
93        if (next_remain) {
94          w25qxx_read_bytes(dec_buffer, address + pre_remain + len
95                            AES_BLO
96          aes_ecb_decrypt(dec_buffer, dec_buffer, AES_BLOCK_SIZE, &
97          memcpy(buffer + pre_remain + len, dec_buffer, next_remain
98        }
99      }
```

Avoid the Chaos

Trigger overflow in the *next_remain*

core > embed > trezorhal > C mpu.c

```c
92    void mpu_config_firmware(void) {
136     MPU->RASR = MPU_RASR_ENABLE_Msk | MPU_RASR_ATTR_FLASH |
140     // SRAM (0x20000000 - 0x2002F                          8 at end,
141     // read-write, execute never)
142     MPU->RNR = MPU_REGION_NUMBER5
143     MPU->RBAR = SRAM_BASE;
144     MPU->RASR = MPU_RASR_ENABLE_M
145                 LL_MPU_REGION_SIZ                           ESS |
146                 MPU_RASR_XN_Msk |
147
148     // Peripherals (0x40000000 - 0x5FFFFFFF, read-write, execute never)
149     // External RAM (0x60000000 - 0x7FFFFFFF, read-write, execute never)
150     MPU->RNR = MPU_REGION_NUMBER6;
151     MPU->RBAR = PERIPH_BASE;
152     MPU->RASR = MPU_RASR_ENABLE_Msk | MPU_RASR_ATTR_PERIPH |
                                                  L_ACCESS |
154                 MPU_RASR_XN_Msk;
```

legacy > C setup.c

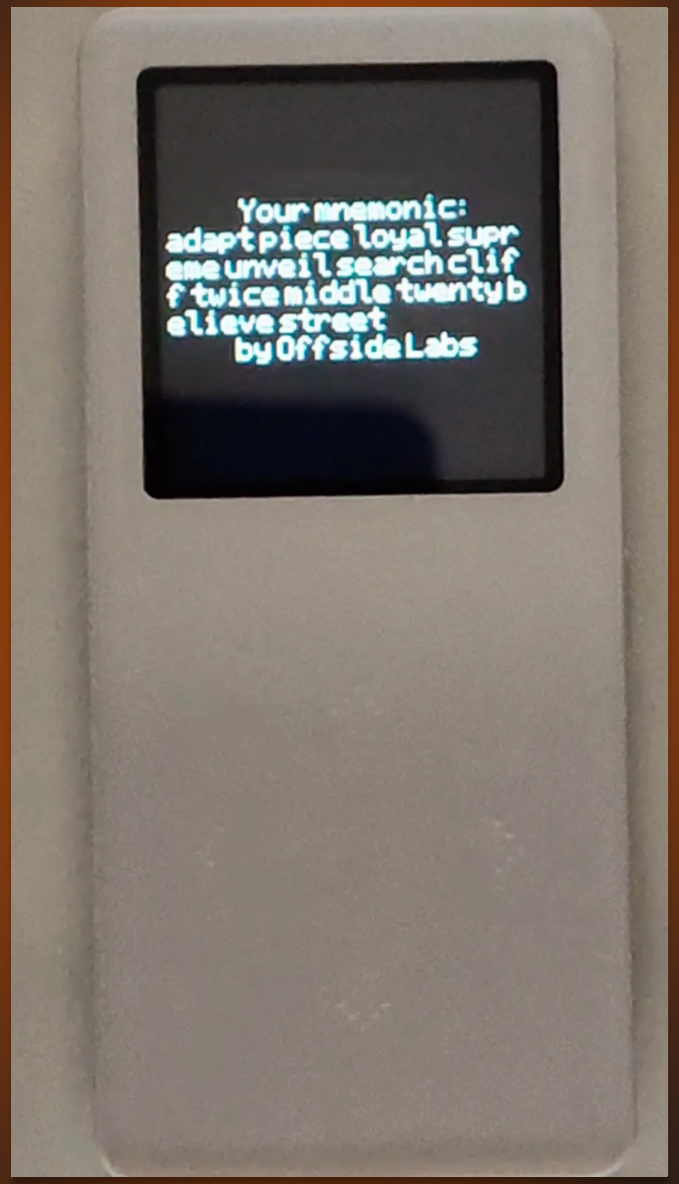
```c
274   void mpu_config_off(void) {
275     // Disable MPU
276     MPU_CTRL = 0;
277
278     __asm__ volatile("dsb");
279     __asm__ volatile("isb");
280   }
```

Exploitation 95% 😃

mpu_config??? Just turn OFF!!!

OneKey Mini

Extract Seed Phrase
without Password



Timeline

2023-11-09: Purchased OneKey Mini.

2023-11-15: Began reviewing the source code of the OneKey Mini and discovered the bug.

2023-11-16: Confirmed the severity of the bug by controlling the PC register.

2023-11-22: Completed the first version of the demo video.

2023-12-05: Reported the vulnerability to OneKey.

2023-12-20: Patch was merged.

2023-12-25: Received bounty for the vulnerability report.

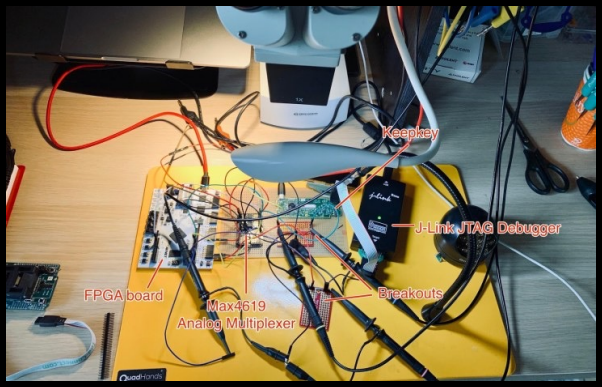
2024-01: Mandatory updates were rolled out.

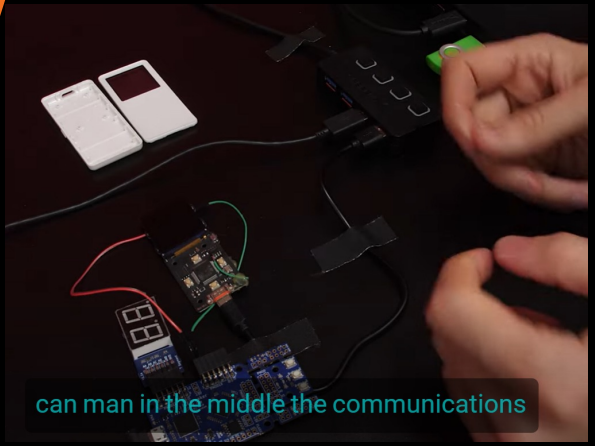
fix factory command invocation permissions. #499

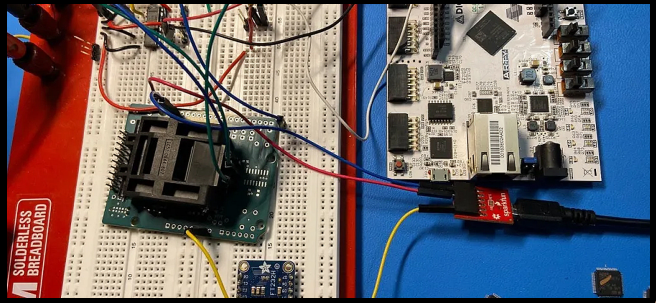
Merged lihuanhuan merged 3 commits into `OneKeyHQ:mini` from `lihuanhuan:mini` on Dec 20, 2023

Hardware Security

- Secure Element
- Anti-tamper
- Clock glitch detection
- Voltage glitch detection
- …



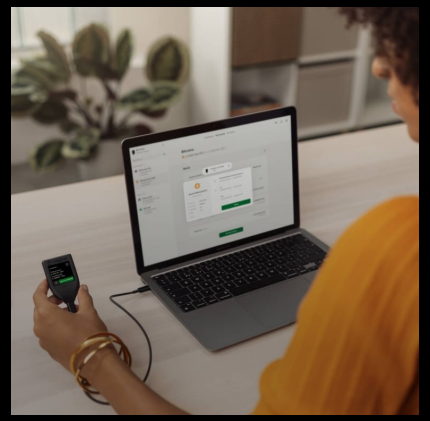


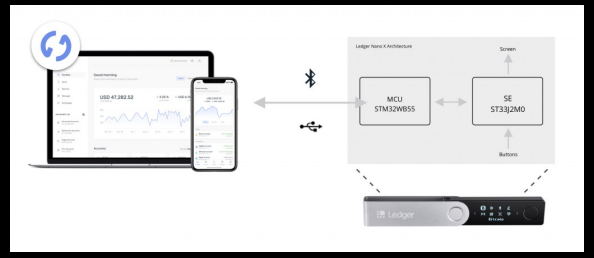


OFFSIDE

AirGap

- One-way communication through QR codes
- No physical connections
- No wireless communications
- …





OFFSIDE

OneKey ~~Mini~~ Pro

Air-gapped signing with camera

Scan the QR code of the transfer message you want to sign, the display will show the completed signature as QR code on the screen. Signatures can be completed on a computer without connecting a data cable in an untrusted environment, making it more secure.

Upgrade your *Mini*, or Use the *OneKey Pro*

Buy

Help

Q & A

OFFSIDE