When Wireless Malware Stays On After Turning Off iPhones

Jiska Classen
Secure Mobile Networking Lab - SEEMOO
TU Darmstadt, Germany

TROOPERS

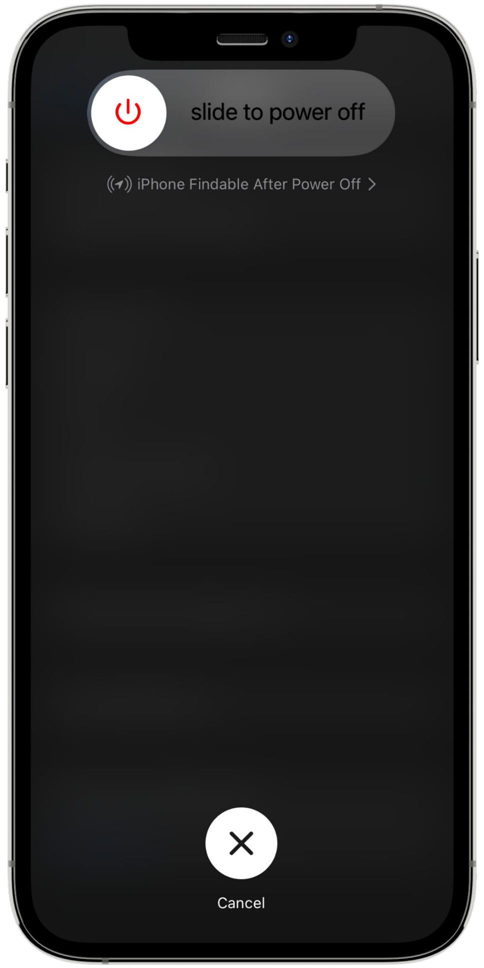
TECHNISCHE UNIVERSITÄT DARMSTADT

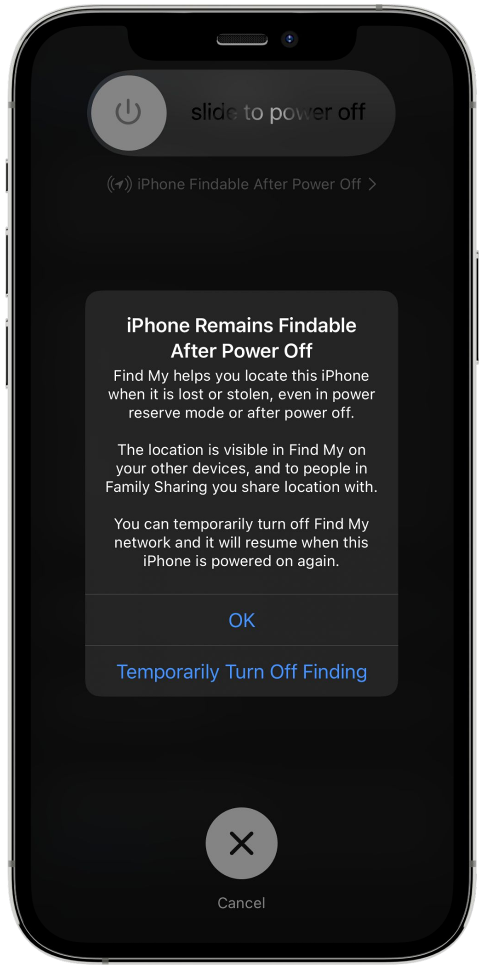
ATHENE National Research Center for Applied Cybersecurity

emergenCITY

Find My
After Power Off

New “Security” Feature (iOS 15)



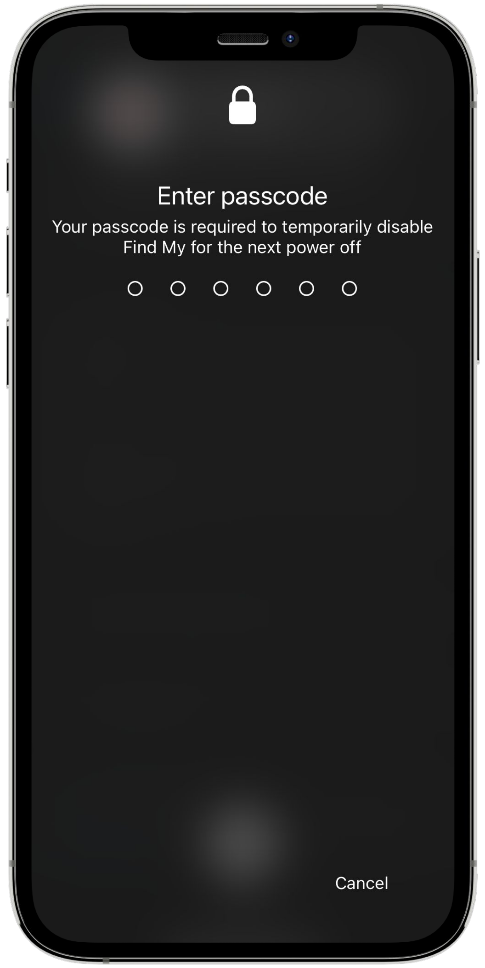


iPhone Remains Findable After Power Off

Find My helps you locate this iPhone when it is lost or stolen, even in power reserve mode or after power off.

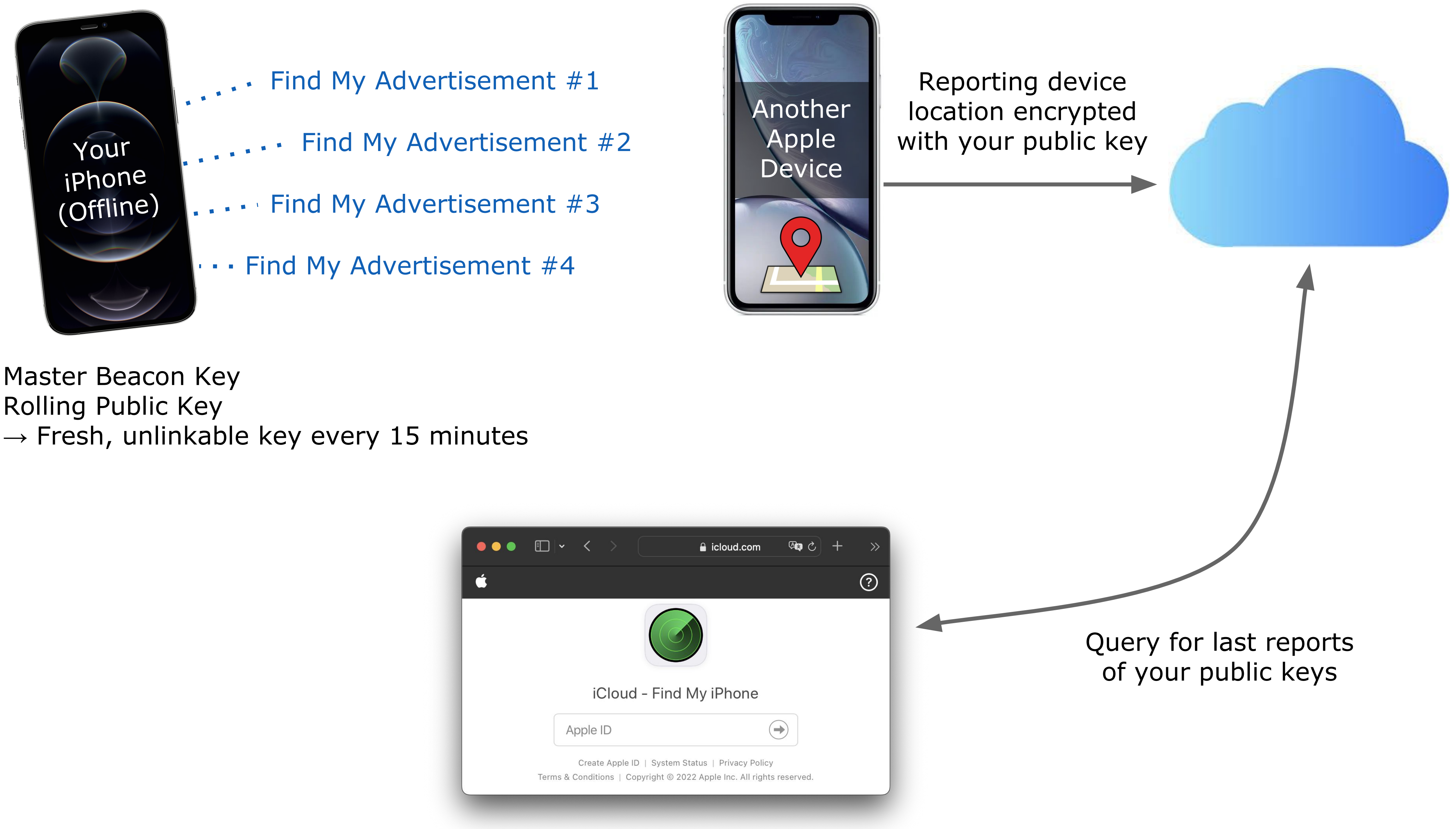
The location is visible in Find My on your other devices, and to people in Family Sharing you share location with.

You can temporarily turn off Find My network and it will resume when this iPhone is powered on again.


Enter passcode

Your passcode is required to temporarily disable Find My for the next power off

Is it a good anti-theft protection?



iCloud - Find My iPhone

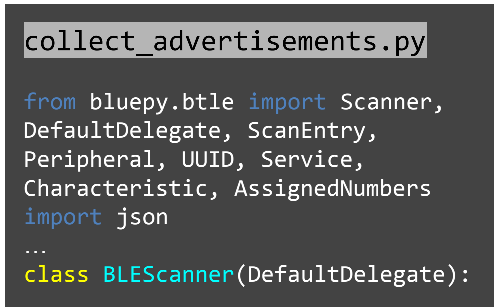
Apple ID

Create Apple ID | System Status | Privacy Policy
Terms & Conditions | Copyright © 2022 Apple Inc. All rights reserved.

- Master Beacon Key
- Rolling Public Key
  → Fresh, unlinkable key every 15 minutes

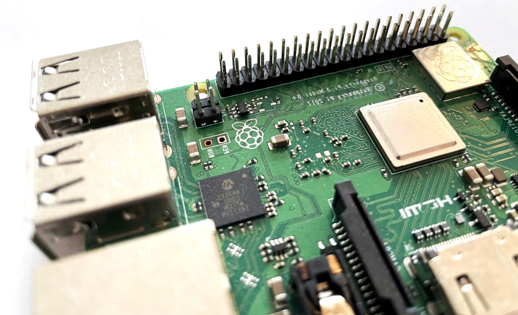
Observations

- Collect BLE advertisements (script by @Sn0wfreeze)
  - User-initiated shutdown.
  - Low battery automated shutdown.
  - Reboot, unlock, …
- Unexpected findings!
  - Advertisements roll every 15min, as with normal Find My.
  - Advertisements stop after 5h on low battery.
  - Advertisements stop after 24h on user-initiated shutdown.
  - Reboot won't re-enable advertisements after 24h without unlock/Internet.
  - Find My dialogue might be shown even if activating Find My fails.



collect_advertisements.py

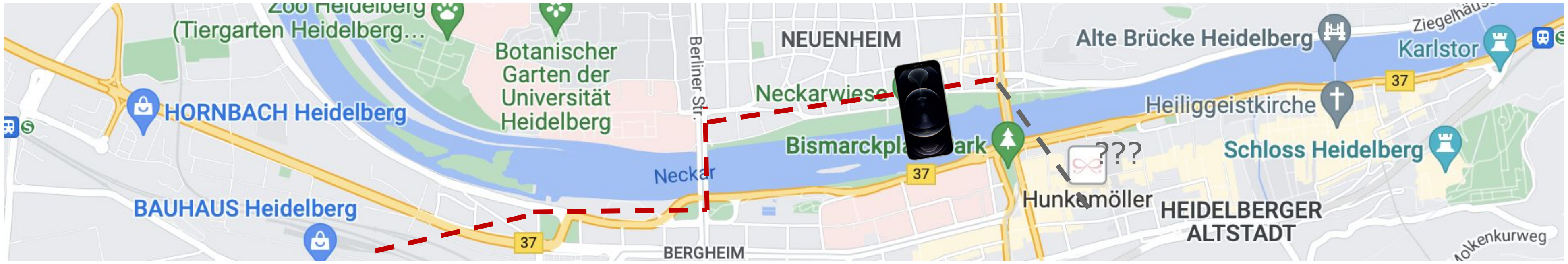
```
from bluepy.btle import Scanner,
DefaultDelegate, ScanEntry,
Peripheral, UUID, Service,
Characteristic, AssignedNumbers
import json
…
class BLEScanner(DefaultDelegate):
```



Technical Limitations

- Secure storage of Find My *Master Beacon Key* vs. promise of anti-theft…

- Master Beacon Key allows generating all past and future advertisements sent by a device.
  → Allows access to the device's location reports.



- On shutdown, 96 Find My advertisements for 15min (=24h) are generated.
  - Sent to the Bluetooth chip.
  - Stored in a database accessible before first unlock.

Is it a good anti-theft protection?

No!

Low-Power Mode

LPM, LEPM, Power Reserve, …

Initial Reverse Engineering

- Get latest iOS IPSW.
- Extract firmware…

```
% strings BCM4387C2_19.3.384.3994_PCIE_Hazelnut_CLPC_OS_USI_20211011.bin | grep Hazelnut
tier2/Olympic/PCIE/Hazelnut_CLPC_OS/USI/bld/A_4387C2_ROM/tier2/patch/bcs/scheduler.o.patch2.c
tier2/Olympic/PCIE/Hazelnut_CLPC_OS/USI/bld/A_4387C2_ROM/tier2/patch/bcs/isr.o.patch2.c
…
tier2/Olympic/PCIE/Hazelnut_CLPC_OS/USI/bld/A_4387C2_ROM/tier2/patch/bcs/mpaf_layer_patch.o.patch2.c
…
tier2/Olympic/PCIE/Hazelnut_CLPC_OS/USI/bld/A_4387C2_ROM/tier2/patch/mpaf/apps/lpm/lpm_app.o.patch2.c
tier2/Olympic/PCIE/Hazelnut_CLPC_OS/USI/bld/A_4387C2_ROM/tier2/patch/mpaf/apps/lpm/lpm_app_gatt.o.patch2.c
tier2/Olympic/PCIE/Hazelnut_CLPC_OS/USI/bld/A_4387C2_ROM/tier2/patch/mpaf/apps/lpm/lpm_app_fsm.o.patch2.c
…
```

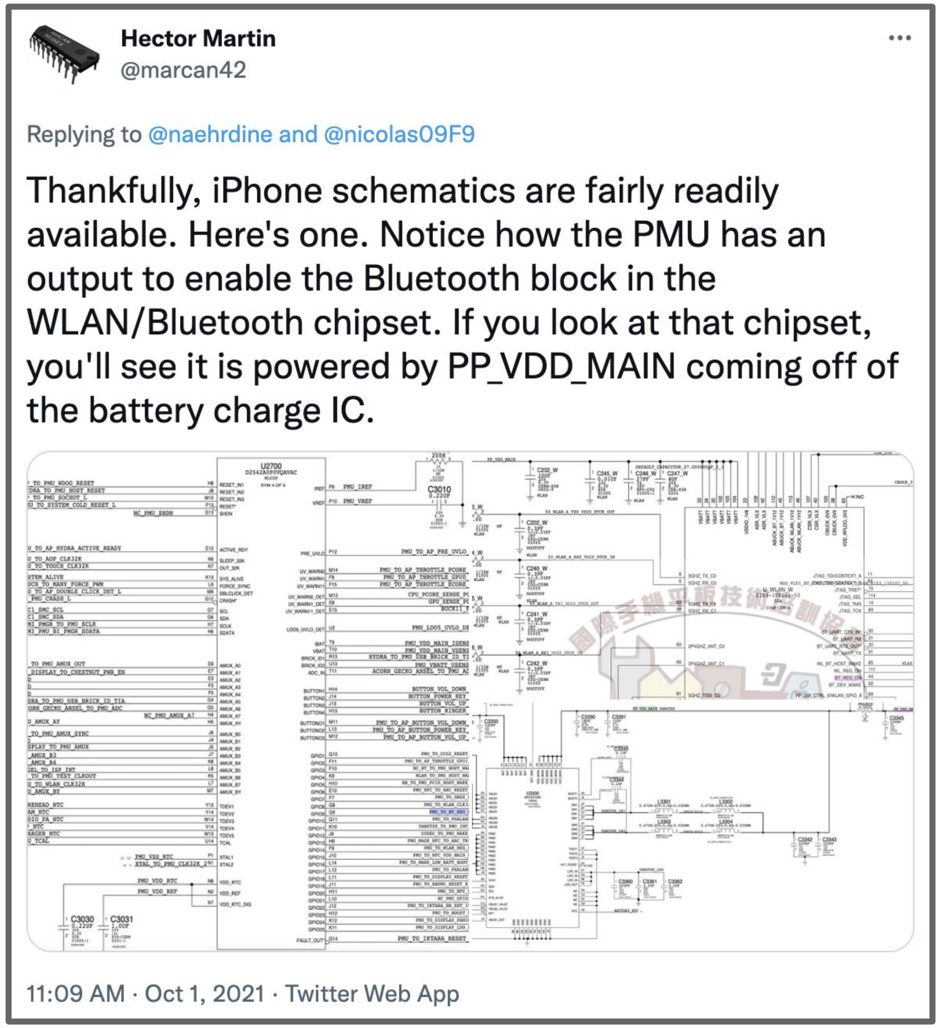
- **MPAF** is the name of a standalone thread, also used for IoT device development in the *Cypress Wiced SDK 6.2*.
- Not surprising at all.
- Reversing finished 🎉

How is the Bluetooth chip powered?

- No idea but Always-on Processor (AoP) firmware has some power module for Bluetooth…

- Publish blog post.

- If you're wrong on the Internet, people will correct you.



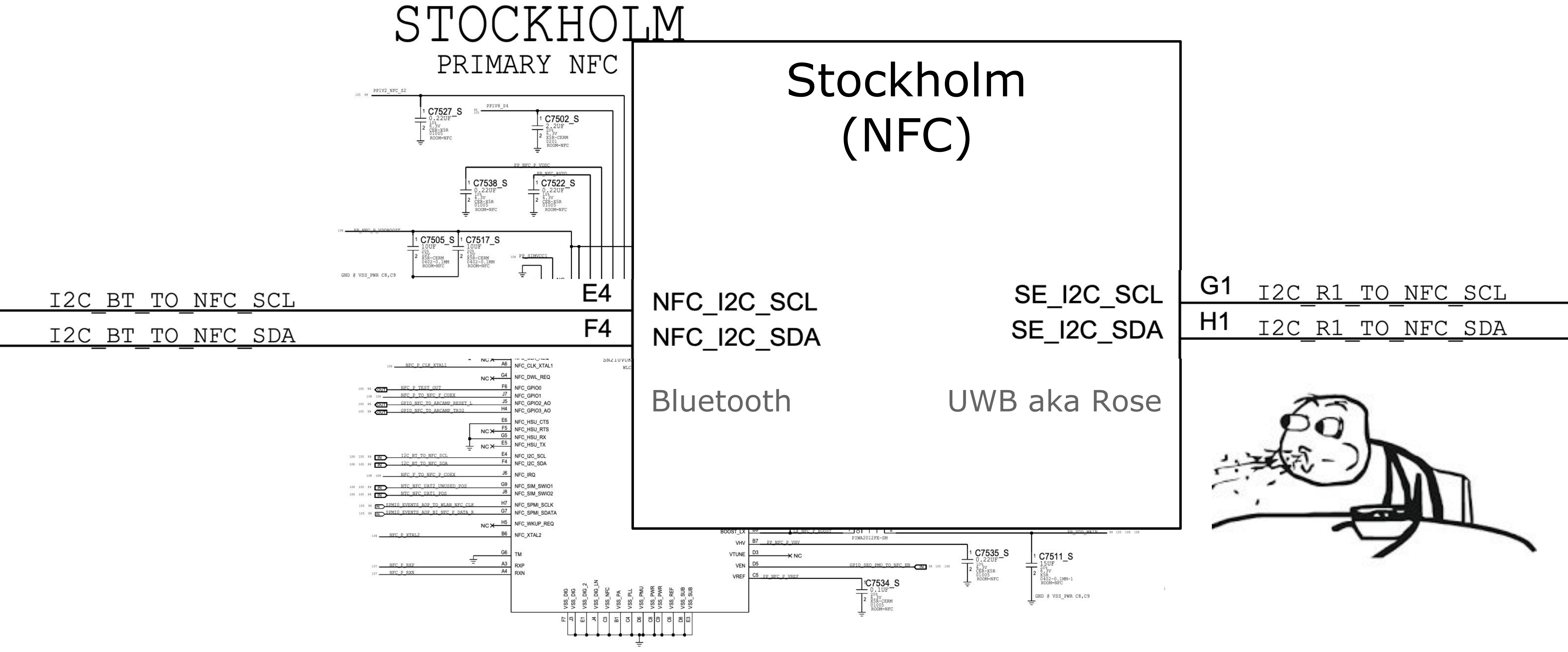
Hector Martin
@marcan42

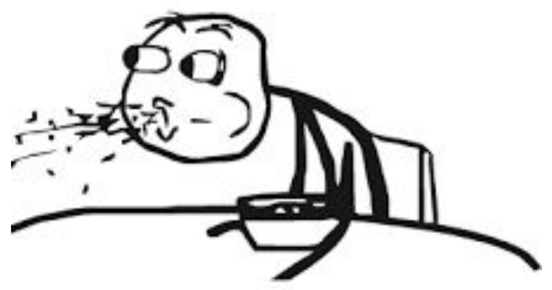
Replying to @naehrdine and @nicolas09F9

Thankfully, iPhone schematics are fairly readily available. Here's one. Notice how the PMU has an output to enable the Bluetooth block in the WLAN/Bluetooth chipset. If you look at that chipset, you'll see it is powered by PP_VDD_MAIN coming off of the battery charge IC.

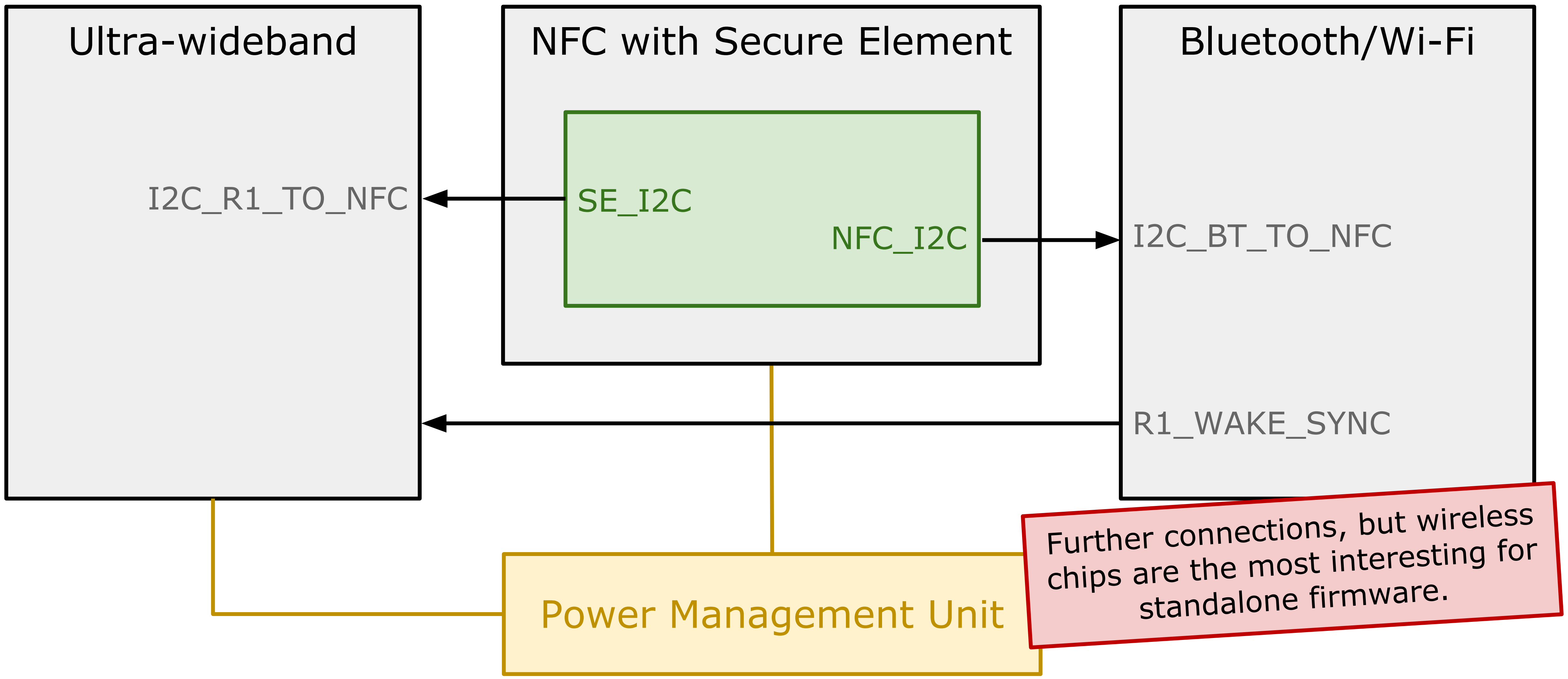
11:09 AM · Oct 1, 2021 · Twitter Web App

Schematics


STOCKHOLM
PRIMARY NFC
Stockholm
(NFC)
I2C_BT_TO_NFC_SCL
I2C_BT_TO_NFC_SDA
E4
F4
NFC_I2C_SCL
NFC_I2C_SDA
SE_I2C_SCL
SE_I2C_SDA
G1
H1
I2C_R1_TO_NFC_SCL
I2C_R1_TO_NFC_SDA
Bluetooth
UWB aka Rose



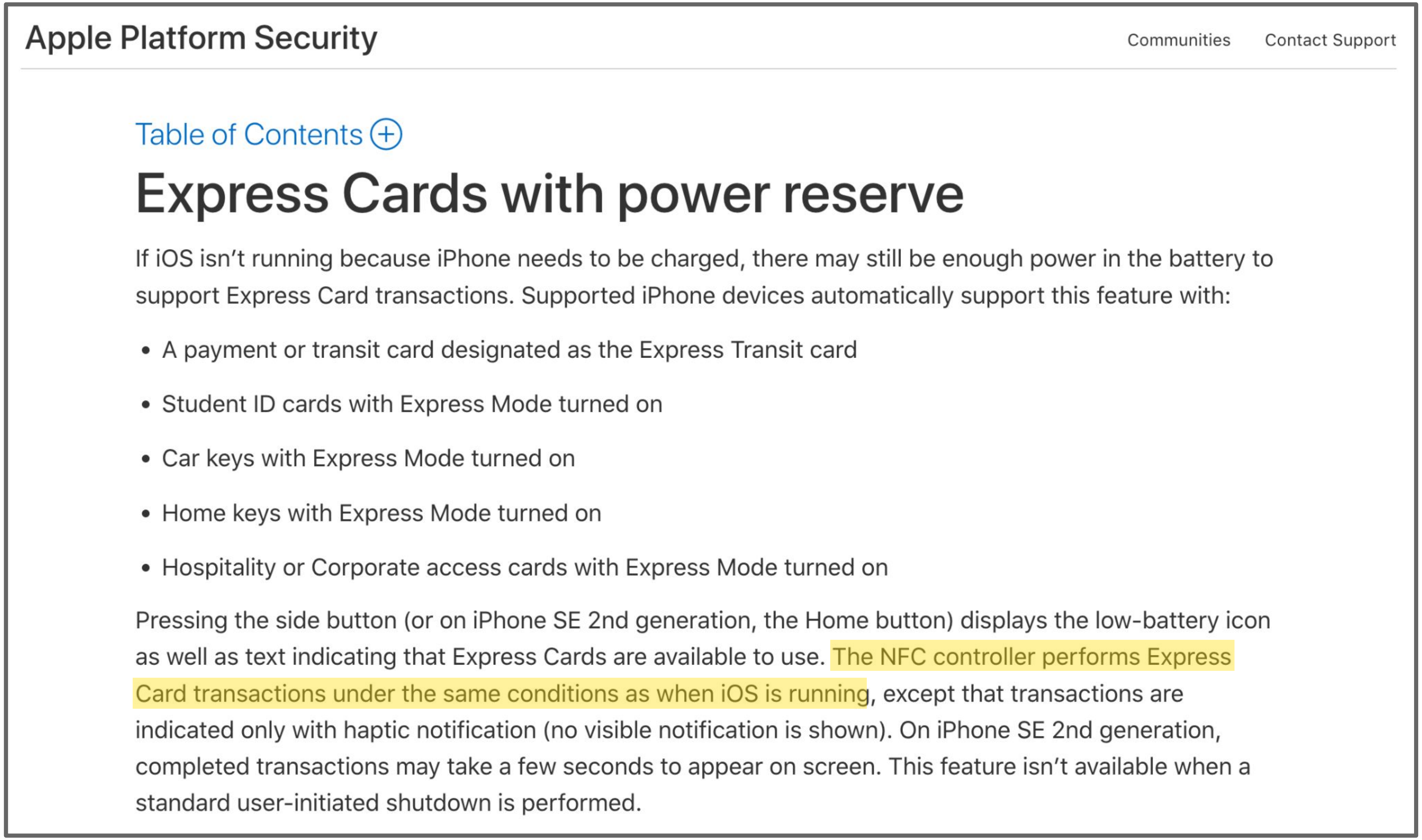
Hardware Changes (iPhone 11 and Newer)



Further connections, but wireless chips are the most interesting for standalone firmware.

iPhone Insomnia

What is all this doing while the iPhone is “off”?


Apple Platform Security

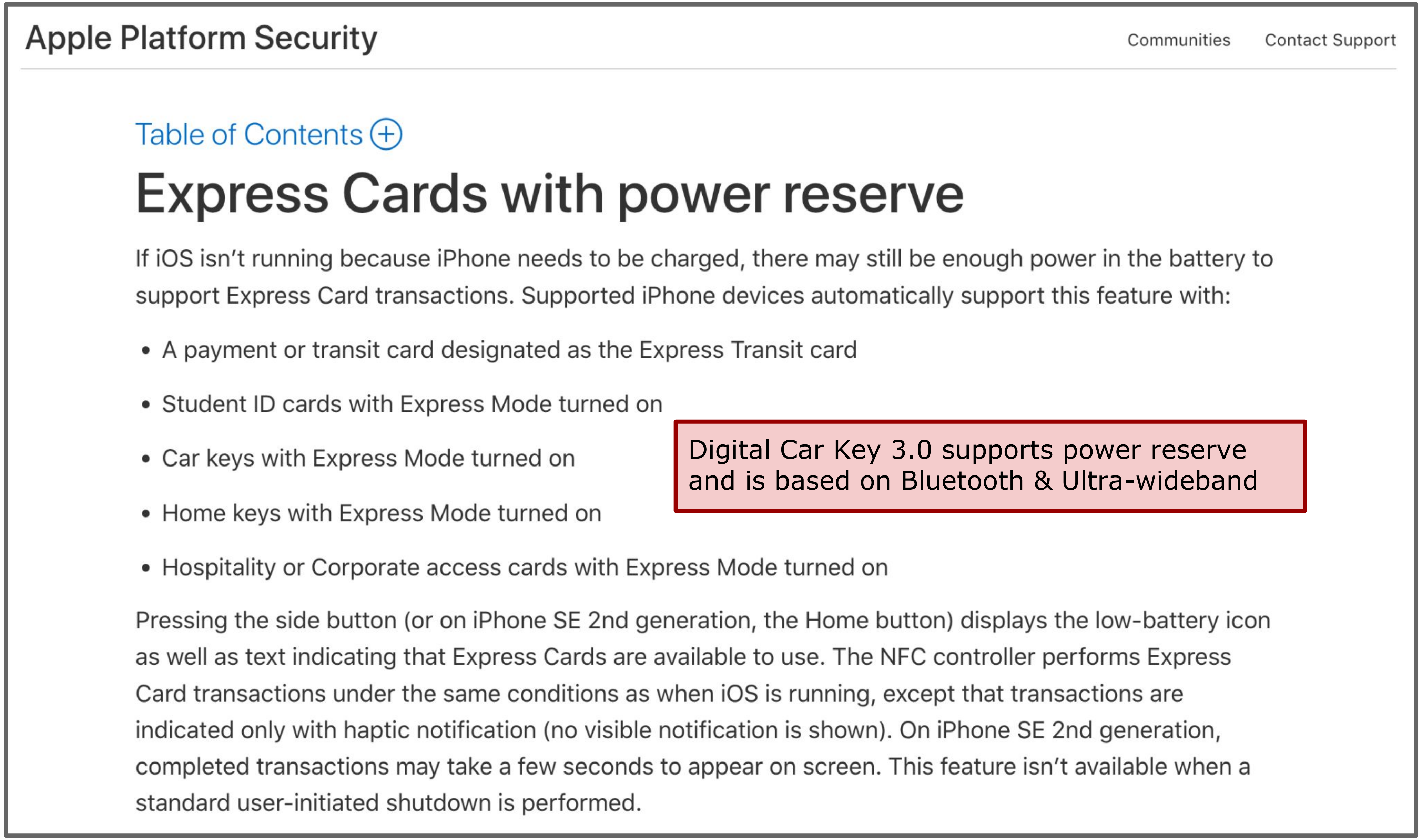
Express Cards with power reserve

If iOS isn't running because iPhone needs to be charged, there may still be enough power in the battery to support Express Card transactions. Supported iPhone devices automatically support this feature with:

- A payment or transit card designated as the Express Transit card
- Student ID cards with Express Mode turned on
- Car keys with Express Mode turned on
- Home keys with Express Mode turned on
- Hospitality or Corporate access cards with Express Mode turned on

Pressing the side button (or on iPhone SE 2nd generation, the Home button) displays the low-battery icon as well as text indicating that Express Cards are available to use. The NFC controller performs Express Card transactions under the same conditions as when iOS is running, except that transactions are indicated only with haptic notification (no visible notification is shown). On iPhone SE 2nd generation, completed transactions may take a few seconds to appear on screen. This feature isn't available when a standard user-initiated shutdown is performed.

https://support.apple.com/guide/security/express-cards-with-power-reserve-sec90cd29d1f/web



Apple Platform Security

Communities Contact Support

Table of Contents ⊕

Express Cards with power reserve

If iOS isn’t running because iPhone needs to be charged, there may still be enough power in the battery to support Express Card transactions. Supported iPhone devices automatically support this feature with:

- A payment or transit card designated as the Express Transit card
- Student ID cards with Express Mode turned on
- Car keys with Express Mode turned on
- Home keys with Express Mode turned on
- Hospitality or Corporate access cards with Express Mode turned on

Pressing the side button (or on iPhone SE 2nd generation, the Home button) displays the low-battery icon as well as text indicating that Express Cards are available to use. The NFC controller performs Express Card transactions under the same conditions as when iOS is running, except that transactions are indicated only with haptic notification (no visible notification is shown). On iPhone SE 2nd generation, completed transactions may take a few seconds to appear on screen. This feature isn’t available when a standard user-initiated shutdown is performed.

Digital Car Key 3.0 supports power reserve and is based on Bluetooth & Ultra-wideband

https://support.apple.com/guide/security/express-cards-with-power-reserve-sec90cd29d1f/web
https://developer.apple.com/videos/play/wwdc2021/10084/ (at 6:08)

“Find My After Power Off” is likely a byproduct of Digital Car Key 3.0.

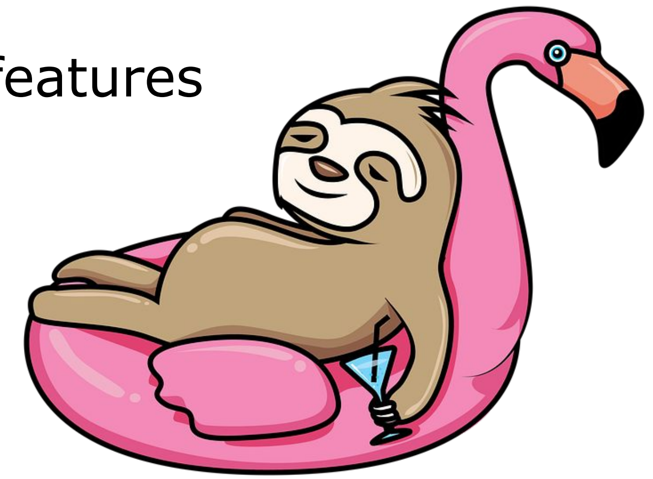
Entering Low-Power Mode

☐ User-initiated

- Find My (24h)
- Find My dialogue is shown
- User can opt out during each shutdown
- Pressing power button turns on phone

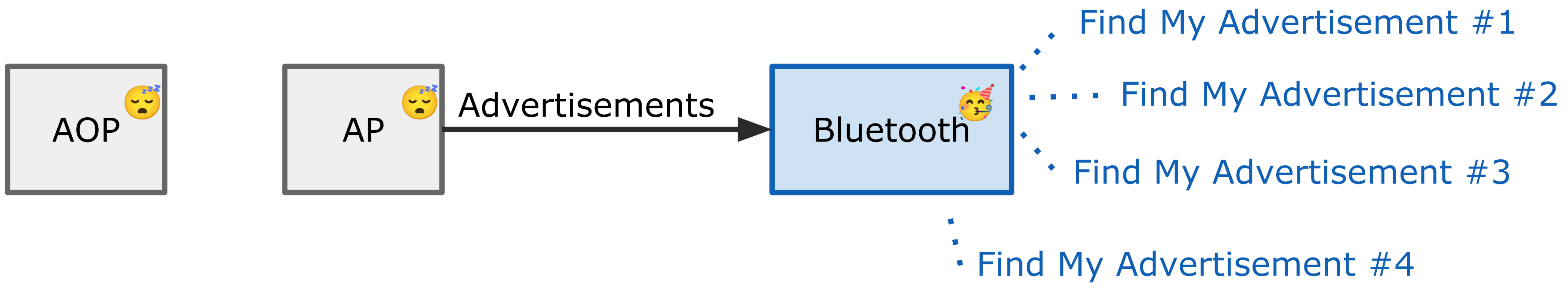
☐ Automated (aka “Power Reserve”)

- Find My + Express Cards and Keys (5h)
- No dialogue, opt out via disabling features in settings
- Pressing power button shows empty battery and enabled features



Basic Principle

- iOS initializes firmware on LPM-enabled chips.
- iOS application processor (and always-on processor) shut down.
- Power Management Unit (PMU) powers chips to run standalone firmware.



Supported Devices

Series	NFC+SE	NFC LPM	Bluetooth/Wi-Fi	UWB	Find My LPM
iPhone Xʀ	NXP SN100	✓	BCM4347B1	—	—
iPhone X□	NXP SN100	✓	BCM4377B2	—	—
iPhone 11	NXP SN200	✓	BCM4378B1	r1p0	✓
iPhone SE 2020	NXP SN200	✓	BCM4378B1	—	—
iPhone 12	NXP SN210	✓	BCM4387C2	r1p1	✓
iPhone 13	NXP SN210	✓	BCM4387C2	r1p2	✓

Same Bluetooth chip but no Find My LPM support…

Analysis & Attack Vectors

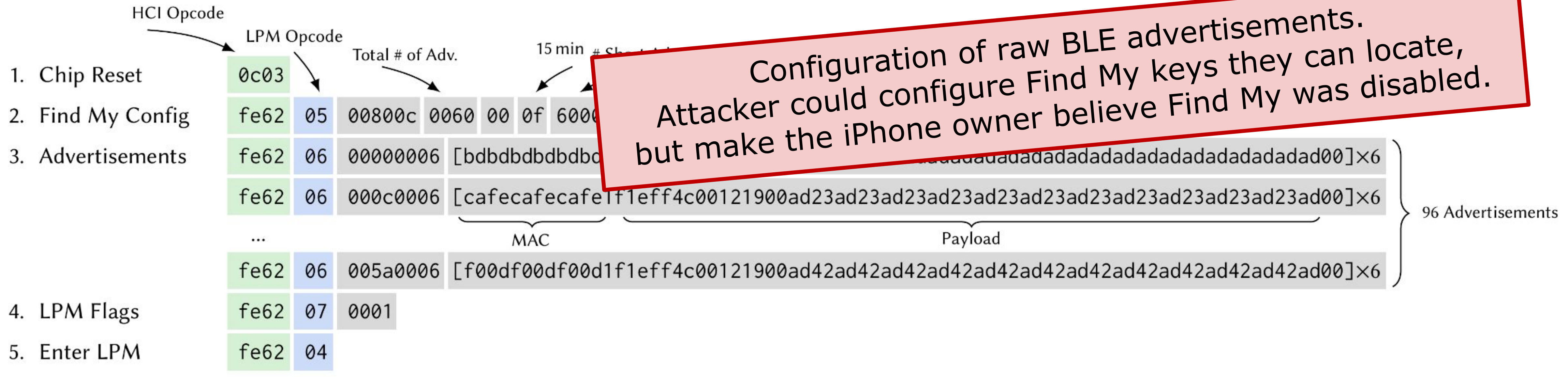
- **NFC/SE** firmware is encrypted and signed.
    - Previous hacking attempts on PN553, but SN100/200/210 is not vulnerable to the same attack.
      https://www.pentestpartners.com/security-blog/breaking-the-nfc-chips-in-tens-of-millions-of-smart-phones-and-a-few-pos-systems/

- **UWB** firmware is only signed.
    - We did lots of static analysis, some of our U1 driver interface reversing published at Black Hat 2021.
    - Florian Kosterhon even fuzzed the interface into chip direction, emulated the firmware for fuzzing, etc. but no vulnerabilities found.
    - UWB has no data transfer, requires activation via BLE, limited attack surface.

- **Bluetooth** firmware??
    - Lots of experience with that in our team!
    - @r0bre wrote an iPhone X☐+ firmware patcher for his thesis.

Bluetooth Firmware Analysis

LPM Initialization

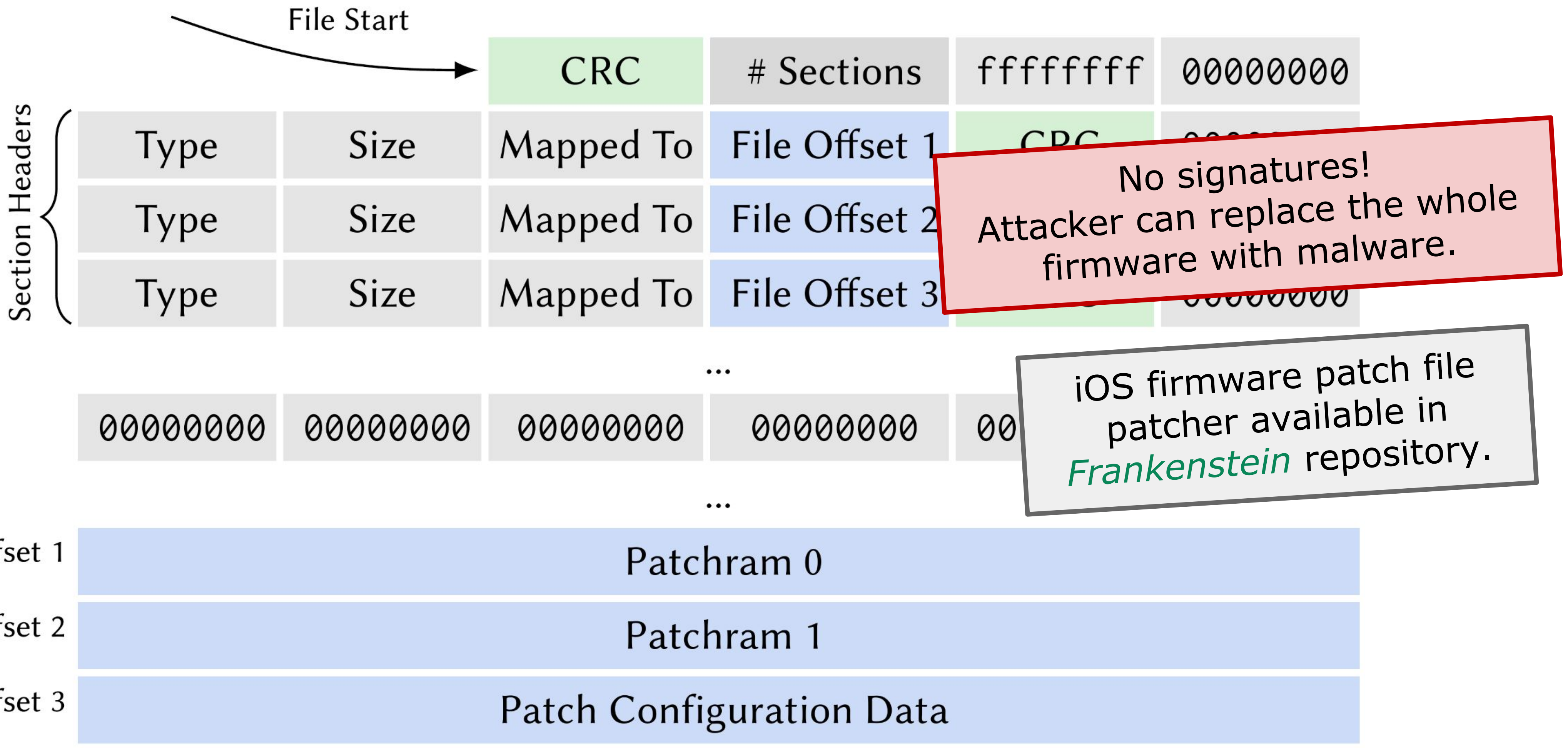


```
1. Chip Reset        0c03
2. Find My Config    fe62 05 00800c 0060 00 0f 600
3. Advertisements    fe62 06 00000006 [bdbdbdbdbdb                  adadadadadadadadadadadadadadadadad00]×6
                     fe62 06 000c0006 [cafecafecafe1f1eff4c00121900ad23ad23ad23ad23ad23ad23ad23ad23ad23ad23ad23ad00]×6
                     ...
                     fe62 06 005a0006 [f00df00df00d1f1eff4c00121900ad42ad42ad42ad42ad42ad42ad42ad42ad42ad42ad42ad00]×6
4. LPM Flags         fe62 07 0001
5. Enter LPM         fe62 04
```

Configuration of raw BLE advertisements.
Attacker could configure Find My keys they can locate,
but make the iPhone owner believe Find My was disabled.

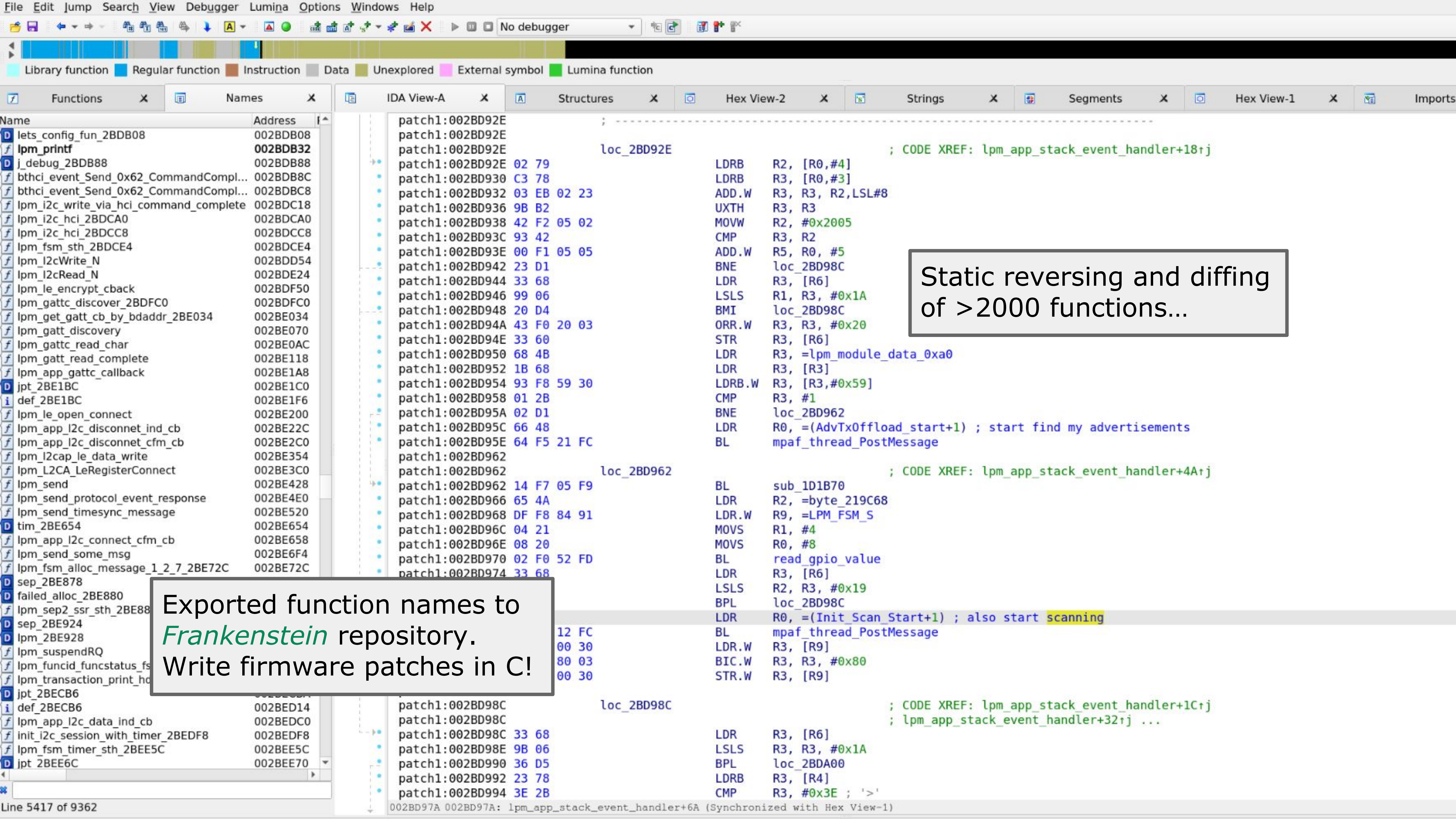
- Initialization commands observed from logs (non-jailbroken iOS 15).
- Static and dynamic firmware analysis to confirm semantics.

Bluetooth Firmware Patch Format



No signatures!
Attacker can replace the whole firmware with malware.

iOS firmware patch file patcher available in *Frankenstein* repository.



```
Name                                         Address
lets_config_fun_2BDB08                       002BDB08
lpm_printf                                   002BDB32
j_debug_2BDB88                               002BDB88
bthci_event_Send_0x62_CommandCompl...        002BDB8C
bthci_event_Send_0x62_CommandCompl...        002BDBC8
lpm_i2c_write_via_hci_command_complete       002BDC18
lpm_i2c_hci_2BDCA0                           002BDCA0
lpm_i2c_hci_2BDCC8                           002BDCC8
lpm_fsm_sth_2BDCE4                           002BDCE4
lpm_I2cWrite_N                               002BDD54
lpm_I2cRead_N                                002BDE24
lpm_le_encrypt_cback                         002BDF50
lpm_gattc_discover_2BDFC0                    002BDFC0
lpm_get_gatt_cb_by_bdaddr_2BE034             002BE034
lpm_gatt_discovery                           002BE070
lpm_gattc_read_char                          002BE0AC
lpm_gatt_read_complete                       002BE118
lpm_app_gattc_callback                       002BE1A8
jpt_2BE1BC                                   002BE1C0
def_2BE1BC                                   002BE1F6
lpm_le_open_connect                          002BE200
lpm_app_l2c_disconnet_ind_cb                 002BE22C
lpm_app_l2c_disconnet_cfm_cb                 002BE2C0
lpm_l2cap_le_data_write                      002BE354
lpm_L2CA_LeRegisterConnect                   002BE3C0
lpm_send                                     002BE428
lpm_send_protocol_event_response             002BE4E0
lpm_send_timesync_message                    002BE520
tim_2BE654                                   002BE654
lpm_app_l2c_connect_cfm_cb                   002BE658
lpm_send_some_msg                            002BE6F4
lpm_fsm_alloc_message_1_2_7_2BE72C           002BE72C
sep_2BE878
failed_alloc_2BE880
lpm_sep2_ssr_sth_2BE88
sep_2BE924
lpm_2BE928
lpm_suspendRQ
lpm_funcid_funcstatus_fs
lpm_transaction_print_ho
jpt_2BECB6
def_2BECB6                                   002BED14
lpm_app_l2c_data_ind_cb                      002BEDC0
init_i2c_session_with_timer_2BEDF8           002BEDF8
lpm_fsm_timer_sth_2BEE5C                     002BEE5C
jpt_2BEE6C                                   002BEE70
Line 5417 of 9362
```

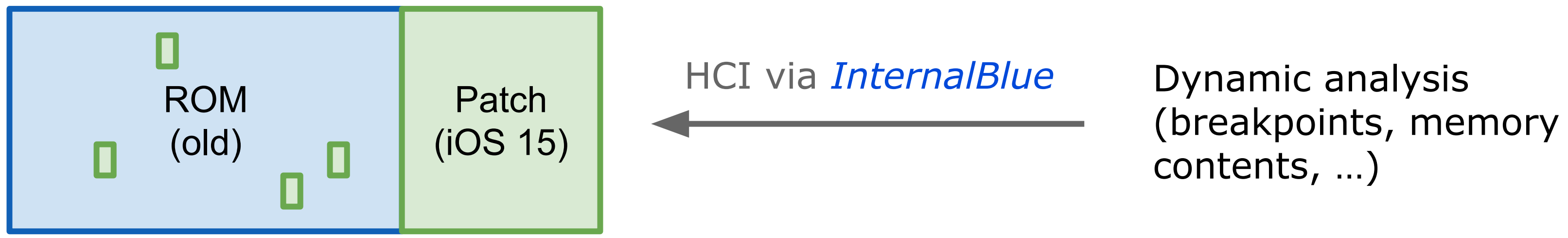
```
patch1:002BD92E             ; ---------------------------------------------------------------------------
patch1:002BD92E
patch1:002BD92E             loc_2BD92E                              ; CODE XREF: lpm_app_stack_event_handler+18↑j
patch1:002BD92E 02 79                       LDRB    R2, [R0,#4]
patch1:002BD930 C3 78                       LDRB    R3, [R0,#3]
patch1:002BD932 03 EB 02 23                 ADD.W   R3, R3, R2,LSL#8
patch1:002BD936 9B B2                       UXTH    R3, R3
patch1:002BD938 42 F2 05 02                 MOVW    R2, #0x2005
patch1:002BD93C 93 42                       CMP     R3, R2
patch1:002BD93E 00 F1 05 05                 ADD.W   R5, R0, #5
patch1:002BD942 23 D1                       BNE     loc_2BD98C
patch1:002BD944 33 68                       LDR     R3, [R6]
patch1:002BD946 99 06                       LSLS    R1, R3, #0x1A
patch1:002BD948 20 D4                       BMI     loc_2BD98C
patch1:002BD94A 43 F0 20 03                 ORR.W   R3, R3, #0x20
patch1:002BD94E 33 60                       STR     R3, [R6]
patch1:002BD950 68 4B                       LDR     R3, =lpm_module_data_0xa0
patch1:002BD952 1B 68                       LDR     R3, [R3]
patch1:002BD954 93 F8 59 30                 LDRB.W  R3, [R3,#0x59]
patch1:002BD958 01 2B                       CMP     R3, #1
patch1:002BD95A 02 D1                       BNE     loc_2BD962
patch1:002BD95C 66 48                       LDR     R0, =(AdvTxOffload_start+1) ; start find my advertisements
patch1:002BD95E 64 F5 21 FC                 BL      mpaf_thread_PostMessage
patch1:002BD962
patch1:002BD962             loc_2BD962                              ; CODE XREF: lpm_app_stack_event_handler+4A↑j
patch1:002BD962 14 F7 05 F9                 BL      sub_1D1B70
patch1:002BD966 65 4A                       LDR     R2, =byte_219C68
patch1:002BD968 DF F8 84 91                 LDR.W   R9, =LPM_FSM_S
patch1:002BD96C 04 21                       MOVS    R1, #4
patch1:002BD96E 08 20                       MOVS    R0, #8
patch1:002BD970 02 F0 52 FD                 BL      read_gpio_value
patch1:002BD974 33 68                       LDR     R3, [R6]
                                            LSLS    R2, R3, #0x19
                                            BPL     loc_2BD98C
                                            LDR     R0, =(Init_Scan_Start+1) ; also start scanning
                    12 FC                   BL      mpaf_thread_PostMessage
                    00 30                   LDR.W   R3, [R9]
                    80 03                   BIC.W   R3, R3, #0x80
                    00 30                   STR.W   R3, [R9]
patch1:002BD98C             loc_2BD98C                              ; CODE XREF: lpm_app_stack_event_handler+1C↑j
patch1:002BD98C                                                     ; lpm_app_stack_event_handler+32↑j ...
patch1:002BD98C 33 68                       LDR     R3, [R6]
patch1:002BD98E 9B 06                       LSLS    R3, R3, #0x1A
patch1:002BD990 36 D5                       BPL     loc_2BDA00
patch1:002BD992 23 78                       LDRB    R3, [R4]
patch1:002BD994 3E 2B                       CMP     R3, #0x3E ; '>'
002BD97A 002BD97A: lpm_app_stack_event_handler+6A (Synchronized with Hex View-1)
```

Static reversing and diffing of >2000 functions…

Exported function names to *Frankenstein* repository. Write firmware patches in C!

Firmware Analysis Workflow

- Dump Bluetooth ROM of iPhone 12 on any (jailbreakable) iOS version with *InternalBlue*.
- Apply iOS 15 firmware patches.
- Perform static analysis to locate functions of interest.
  Modify functions in patch file if needed (e.g., allow `Write_RAM`, `Launch_RAM`).
- Load patches to jailbroken iPhone for dynamic analysis.



Running Backported Firmware

- Run iOS 15 LPM firmware on jailbroken iPhone with iOS 14, load it with `BlueTool`.

```
power off
device -D
bcm -w /tmp/fw.bin
```

- Send the same LPM initialization commands to the firmware.
- Entering LPM (`fe 62 04`) terminates HCI communication with the host.
    - We can no longer debug what's going on.
    - `bluetoothd` will immediately restart the Bluetooth chip upon timeouts.
    - Would require lots of firmware patching for analysis…
- Dumping Bluetooth RAM with *InternalBlue* just before entering LPM.
    - See all structures in RAM.
    - Cross-reference their access in functions.
    - Analyze and name functions, e.g., the one rolling advertisements.

```
int bthci_cmd_0xfe62_set_advertisements(int input, int a2, byte* a3) {
…
 if ( lpm_module_data_0xa0 ) {
   switch (input + 12) { // get LPM opcode from HCI command
     case 1:
       r = bthci_cmd_lmp_0xfe62_0x01(input + 9);
       break;
     case 2:
       r = bthci_cmd_lmp_0xfe62_0x02(input + 9);
       break;
     case 3:
       r = bthci_cmd_lmp_0xfe62_0x03(input + 9);
       break;
     case 4:  // Enter LPM
       r = bthci_cmd_lmp_0xfe62_finally_activate_0x04(input + 9, a2, a3, v4);
       …
       break;
     case 5:  // Find My configuration
       r = bthci_cmd_lmp_0xfe62_0x05_set_advertisement_config(input + 9);
       break;
     case 6:  // Set advertisements
       r = bthci_cmd_lmp_0xfe62_0x06_add_advertisements(input + 9);
       break;
     case 7:  // Final step
       r = bthci_cmd_lmp_0xfe62_0x07_after_advertisements(input + 9, a2, a3, v4);
       break;
     default: // Error code: HCI command disallowed
       r = 18;
       break;
   }
…
```

Some LPM initialization functions not used by iOS Find My setup.

Name functions starting from here.

Enable `Write_RAM` for Dynamic Analysis

- iOS applies Bluetooth firmware patch to RAM.
- Firmware patch then disables `Write_RAM` command.
- Statically remove check, calculate new CRCs.
- Patched firmware available in *InternalBlue* repository.

```
patch1:002C57D2             disallow_fwupdate                       ; CODE XREF: bthci_cmd_HandleCommand+1D6↑j
patch1:002C57D2 2E 2A                        CMP     R2, #0x2E      ; VSC_HandleDownload_Minidriver
patch1:002C57D4 00 F0 C3 81                  BEQ.W   command_disallowed
patch1:002C57D8 4C 2A                        CMP     R2, #0x4C      ; VSC_Write_RAM
patch1:002C57D8                                                     ; -> disallows writing to RAM via HCI
patch1:002C57DA 00 F0 C0 81                  BEQ.W   command_disallowed
patch1:002C57DE 0B E2                        B       call_default_command_handler
```

Interesting Functions in LPM Firmware

- MPAF thread calls into multiple BLE functions known from leaked symbols (CYW20735 etc. from *Wiced Studio 6.2*).
  - BLE advertisements for Find My.
  - Scanning for other devices, GATT service, … likely all used for Digital Car Key.
- Analyzing Digital Car Key 3.0?
  - Dynamic analysis would require to also backport NFC + UWB firmware.
  - At the same time also requires the NFC SE applet and various user-space daemon updates introduced in iOS 15.
  - …waiting for an iOS 15 jailbreak.

Impact of Firmware Modification

- Malicious Bluetooth firmware could be installed. 🐛
  - Only protection is CRC, no signature checks.
  - Code execution on system → code execution after “power off”.

- Use jailbroken iOS 14.x iPhones for dynamic Bluetooth security analysis. 🔍
  - Modify firmware, e.g., install your own patches for testing.
  - We bring back *Frankenstein* and *InternalBlue* support to the iPhone X☐, 11, 2020 SE, 12, and 13 Bluetooth chips!

https://github.com/seemoo-lab

Conclusions

You have twenty four hours to find your iPhone!

24h Limitation in Find My LPM Module

- No real limitation why LPM should only run 24h.

- Generate more advertisements?
  Quite some memory usage and Bluetooth memory is very limited.

- Master Beacon Key protections…
  Copy key to NFC Secure Element, request more advertisements when empty?
  (Not implemented as of now.)

Security Impact

- New *"Find My After Power Off"* feature markets LPM for anti-theft.
  Current implementation makes false promises and does not prevent theft. 🔓

- Turning off the main processor 😴 no longer turns off all chips.
  High-value targets can no longer trust iPhones that off means off. 🥳

- Direct connections between wireless technologies.
  Chips might extract each other's secrets or even execute code. 💥
  (Code execution has been shown for Bluetooth→Wi-Fi…)

Q&A

https://github.com/seemoo-lab/{frankenstein,internalblue}

Twitter: @naehrdine

jclassen@seemoo.de

🎓 https://arxiv.org/abs/2205.06114