Spectra

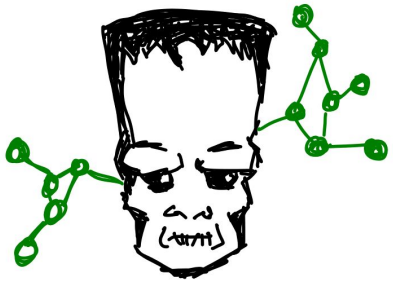
New Wireless Escalation Targets

Jiska Classen
Secure Mobile Networking Lab - SEEMOO
Technische Universität Darmstadt, Germany

Francesco Gringoli
Dept. of Information Engineering
University of Brescia, Italy

Motivation

When you got Bluetooth on-chip RCE...



FRANKENSTEIN

CVE-2019-11516
CVE-2019-13916
CVE-2019-18614

35C3 Talk: https://media.ccc.de/v/35c3-9498-dissecting_broadcom_bluetooth, Frankenstein Fuzzer: https://github.com/seemoo-lab/frankenstein

...but Wi-Fi has more privileges.

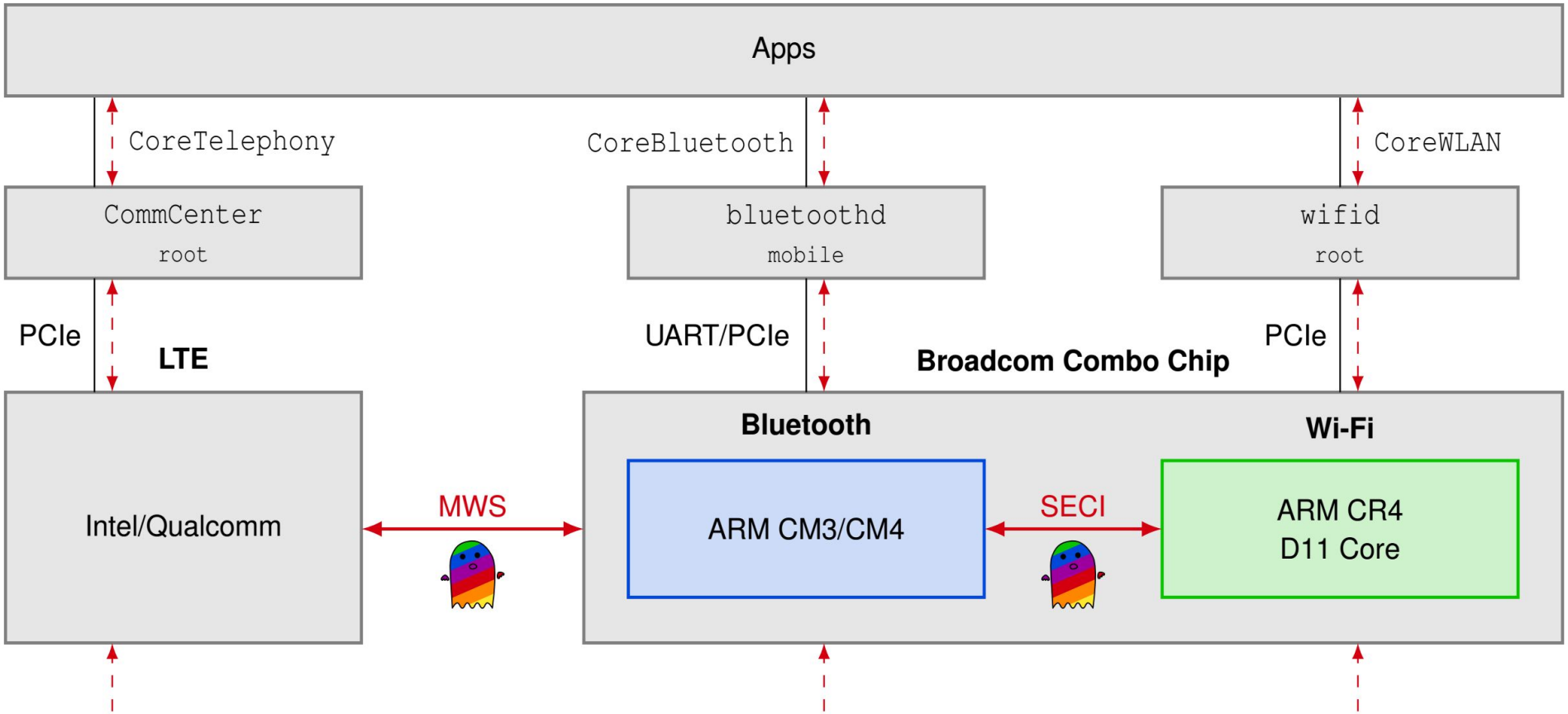
Let’s break inter-chip separation!

Spectra: ~~Speculative~~Spectrum Transmission

- Wi-Fi, Bluetooth, and even LTE share frequencies in the 2.4 GHz spectrum.
- They cause interference in small devices like smartphones.

- Wireless combo chip **performance optimization**: enhanced coexistence mechanisms.
- Observable side effects of transmission delays and coordination lead to **side channels**.
- Attackers require **code execution** privileges, but they can escalate between wireless cores without further checks by the operating system.

Wireless Architecture (iOS)



Spectra Impact

1. **Denial of Service**
   One wireless core denies transmission to the other core.

2. **Information Disclosure**
   One wireless core can infer data or actions of the other core.

3. **Code Execution**
   One wireless core can execute code within the other core.

Broadcom Coexistence

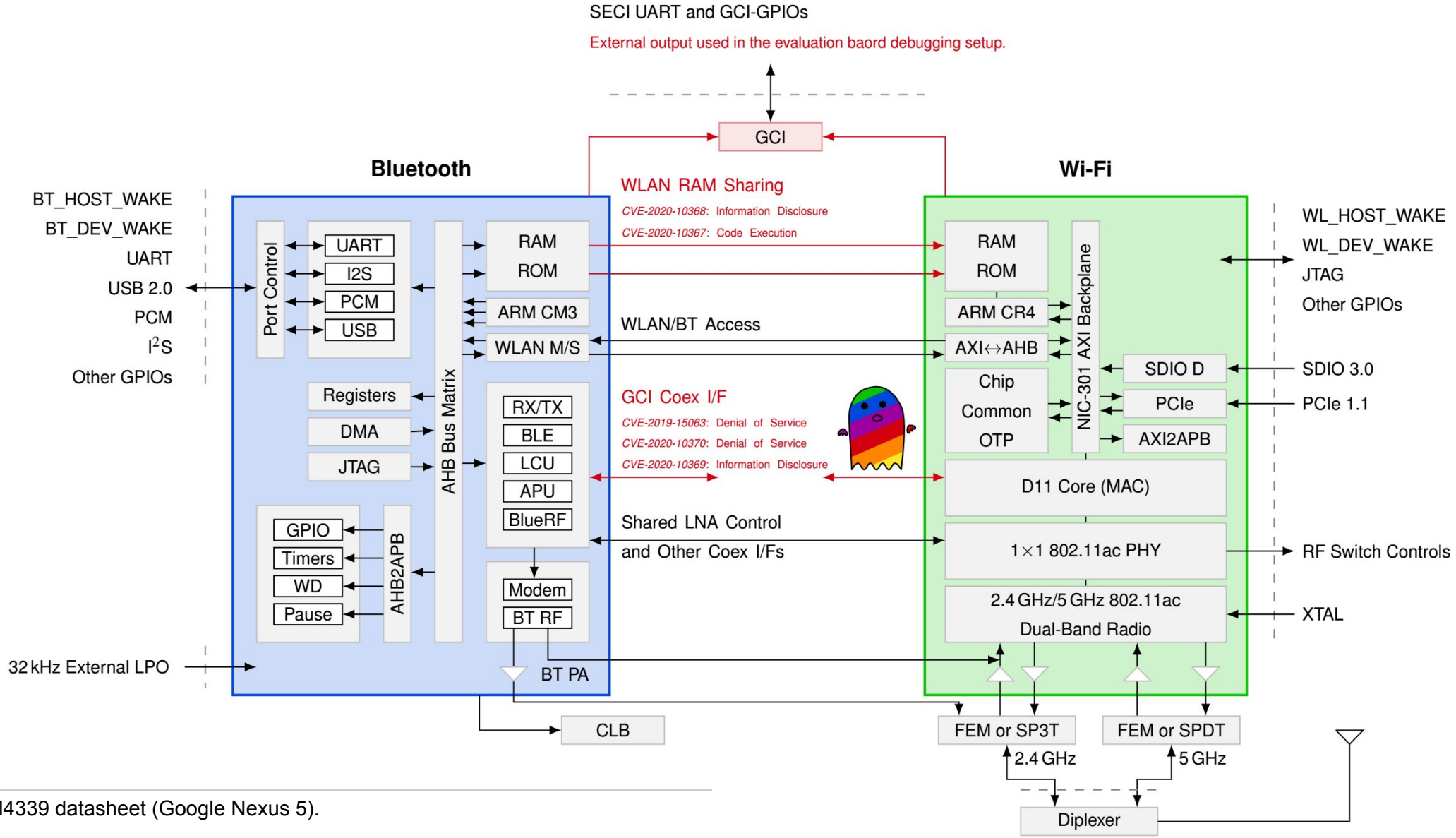
(and Cypress)

Broadcom: ~ 1 Billion Devices

- Apple
  - All iPhones, MacBooks, iMacs, older Apple Watches
- Samsung
  - Samsung Galaxy S and Note series in Europe
- Google
  - Only older devices, e.g., Nexus 5/6P
- Raspberry Pi
- IoT devices
  - Fitbit Ionic

…and no firmware checks. A perfect prototyping platform \o/

Coexistence: Escalation within the chip



From the BCM4339 datasheet (Google Nexus 5).

5 Product Overview

The Cypress CYW43XXX WLAN, CYW207XX BT, or WLAN/BT combo chips provide the highest level of integration for wearables and other consumer/industrial embedded applications. The device includes integrated PMU, IEEE 802.11 MAC/baseband, radio, and Bluetooth for each combo chip. It supports all rates specified in the IEEE 802.11a/b/g/n/ac specifications. It also supports optional antenna diversity for improved RF performance in difficult environments.

An embedded wireless system-on-a-chip (SoC) CYW43907, which includes an ARM-based processor as well as WLAN, offers the lowest RBOM in the industry and is uniquely suited for Internet-of-Things (IoT) applications.

For the WLAN section, several alternative host interfaces are included: SDIO, SPI, and PCIe depending on the products. For the Bluetooth section, a host interface option using high-speed 4-wire UART and PCM (Pulse-Code Modulation) digital audio are provided.

In an integrated single-chip combo solution, the CYW43XXX implements highly sophisticated enhanced collaborative coexistence hardware mechanisms and algorithms to enable the WLAN and Bluetooth to operate simultaneously and to ensure maximum medium access time, high throughput, and audio quality. Collaborative coexistence between WLAN and Bluetooth is implemented according to IEEE 802.15.2 Packet Traffic Arbitration (PTA) and through Cypress's Enhanced Coexistence Interface (ECI). ECI augments PTA signaling by enabling exchange of additional information required for implementing more advanced collaborative coexistence methods. As a result, overall quality for simultaneous voice, video, and data transmission on an embedded system is achieved.

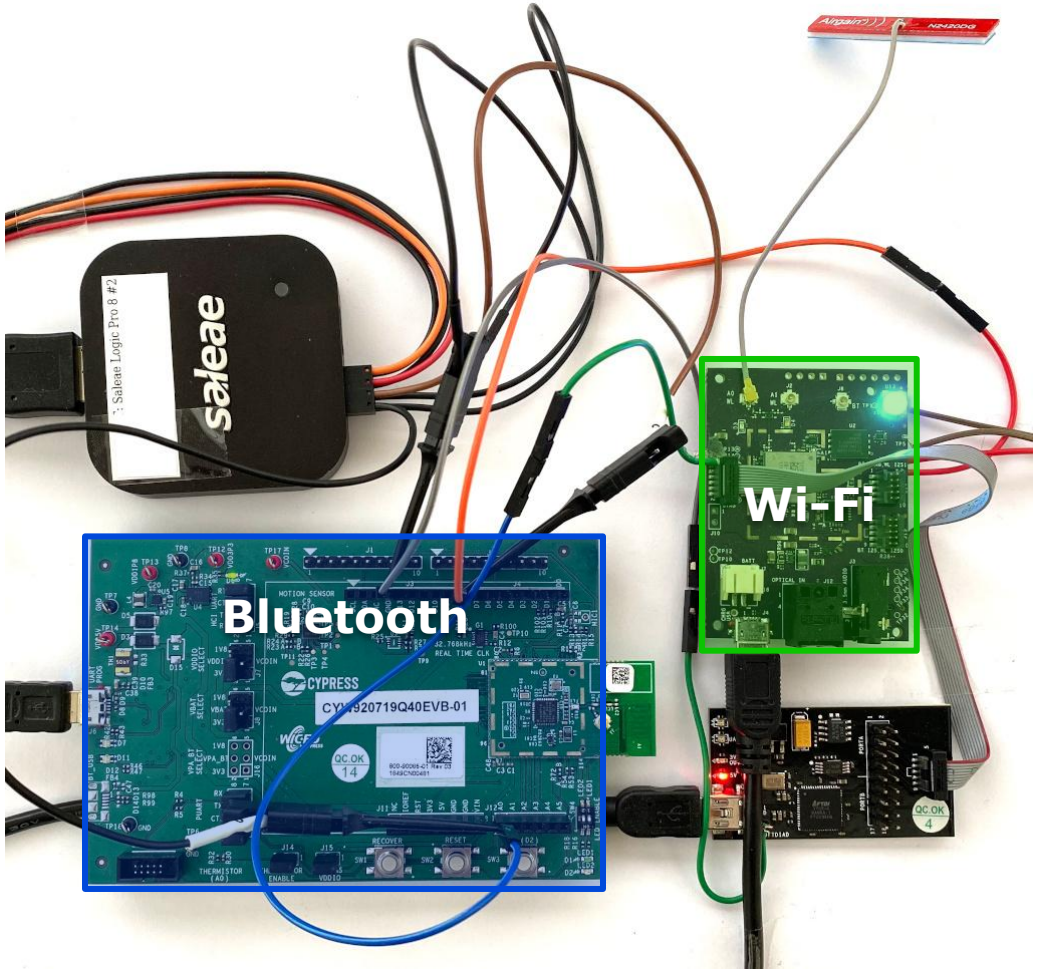
By the way, throughput really sucks with ECI disabled. You cannot stream a video with Wi-Fi and listen to it with your Bluetooth headset.

Serial Enhanced Coexistence Interface

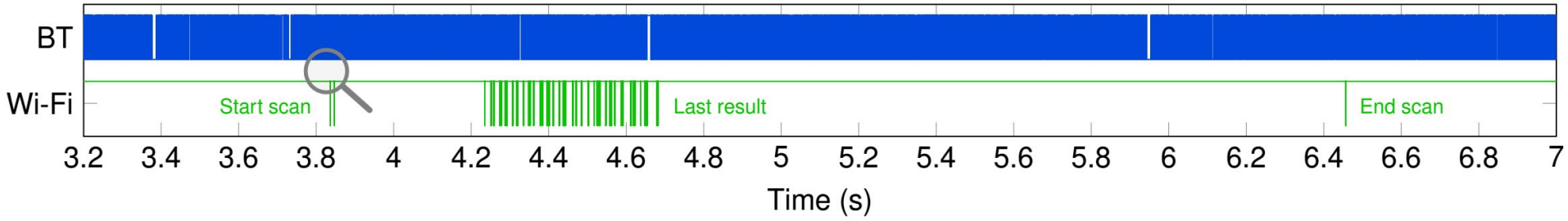
(also SECI, ECI, GCI)

Serial Enhanced Coexistence Interface

- Separate **Bluetooth** (CYW20719) and **Wi-Fi** (CYW490307) boards.
- Only connection: Serial Enhanced Coexistence Interface (SECI).
- Separate antennas, exclude side effects!
- Debugging with logic analyzer.

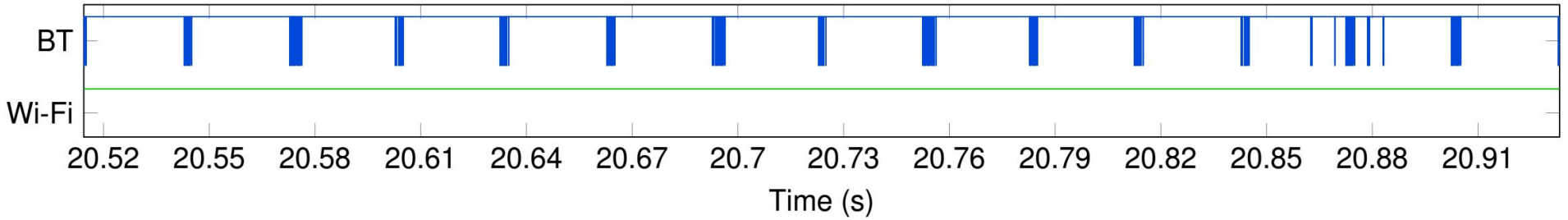


What does it look like?



a) SECI while streaming music via Bluetooth and scanning for Wi-Fi access points.

b) SECI while streaming music via Bluetooth and scanning for Wi-Fi access points, zoomed in to first Wi-Fi peak.



c) SECI while typing on a Bluetooth keyboard, Wi-Fi inactive.

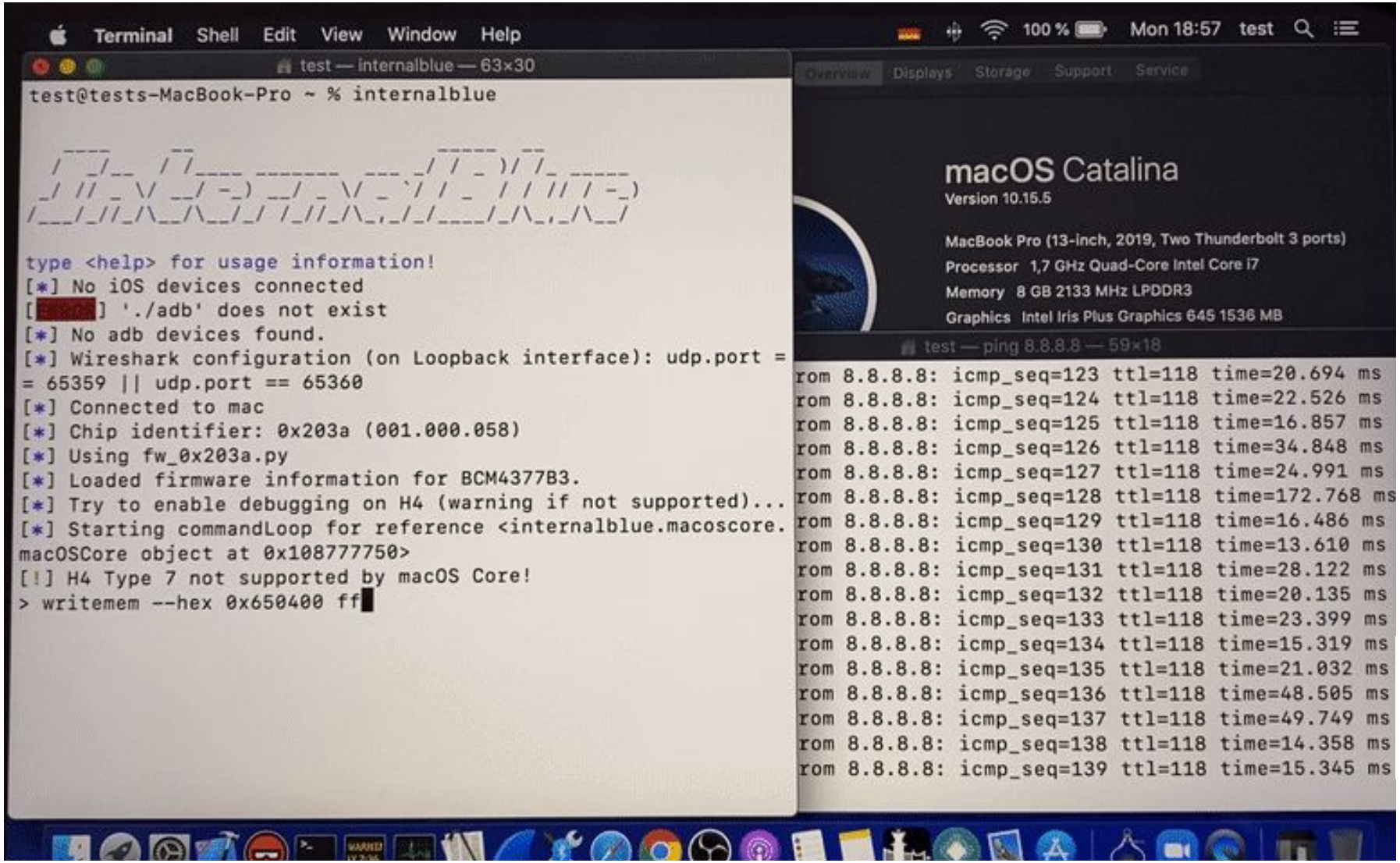
Reconfigure SECI

Denial of Service BT→Wi-Fi

CVE-2019-15063 (reported August 2019)

- When **Bluetooth** writes to the `gci_chipcontrol` register at `0x650200`, this **crashes Wi-Fi**.
- We can observe a **voltage drop** with the logic analyzer.
- Causes a **kernel panic** on various devices, Wi-Fi PCIe behaves really strange afterward...

macOS Kernel Panic Demo



Rewrite SEC1 config →

← Immediate Wi-Fi crash

Denial of Service BT→Wi-Fi

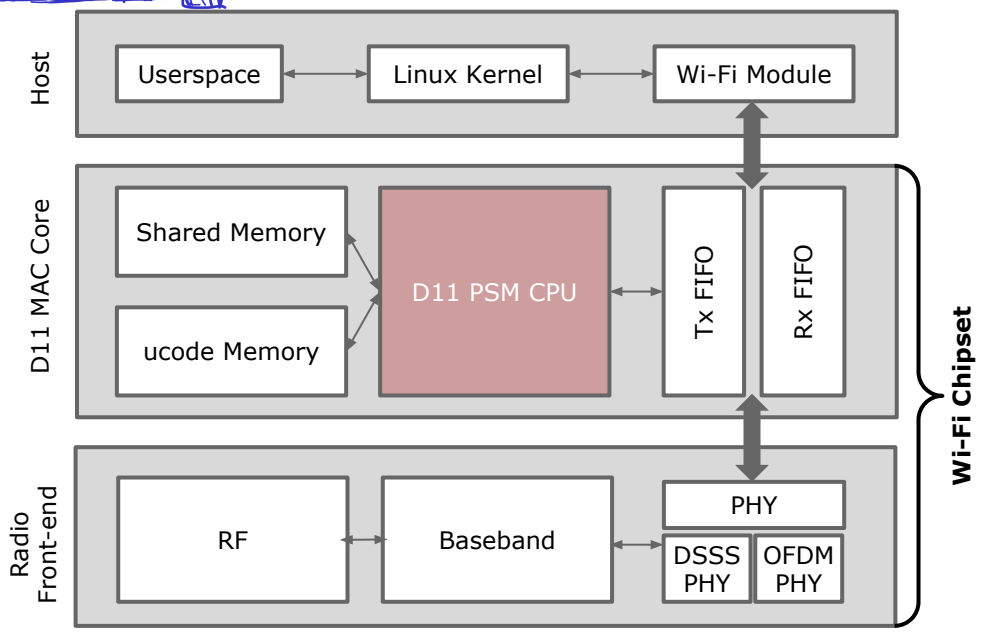
Chip	Device	OS	Build Date	Address	Value	Effect
BCM4335C0	Nexus 5	Android 6.0.1	Dec 11 2012	`0x650440`, `0x650600`	`00`	Disconnects all Wi-Fi, Wi-Fi can be reconnected.
BCM4345B0	iPhone 6	iOS 12.4	Jul 15 2013	`0x650000`–`0x6507ff`		Disables only 2.4 GHz Wi-Fi until restarting Bluetooth.
BCM4345C0	Raspberry Pi 3+/4	Raspbian Buster	Aug 19 2014	`0x650000`–`0x6507ff`	Random	Full and partial Wi-Fi crashes, including Secure Digital Input Output (SDIO), ability to scan for Wi-Fis, speed reduction. Reboot required to restore functionality.
BCM4358A3	Nexus 6P	Android 7.1.2	Oct 23 2014	`0x650000`–`0x6507ff`		Disables all Wi-Fi until restarting Bluetooth.
BCM4358A3	Samsung Galaxy S6	Lineage OS 14.1	Oct 23 2014	`0x650000`–`0x6507ff`		Disables all Wi-Fi until restarting Bluetooth.
BCM4345C1	iPhone SE	iOS 12.4–13.3.1	Jan 27 2015	`0x650200`	`ff`	Kernel panic, resulting in a reboot.
BCM4355C0	iPhone 7	iOS 12.4–13.3.1	Sep 14 2015	`0x650200`		Kernel panic, resulting in a reboot.
BCM4347B0	Samsung Galaxy S8	Android 8.0.0	Jun 3 2016	`0x650200`		Disables all Wi-Fi, kernel panic & reboot when re-enabling it.
BCM4347B0	Samsung Galaxy S8	LineageOS 16.0	Jun 3 2016	`0x650200`		Temporarily disables all Wi-Fi, freezes system for a couple of seconds when re-enabling Wi-Fi.
BCM4347B1	iPhone 8/X/XR	iOS 12.4–13.3.1	Oct 11 2016	`0x650200`		Kernel panic, resulting in a reboot.
CYW490307+ CYW20719	Evaluation Boards	ThreadX+Linux	Jan 17 2017	`0x650200`		SECI voltage drop, only observable with the logic analyzer.
BCM4375B1	Samsung Galaxy S10/S10e/S10+	Android 9	Apr 13 2018	`0x650200`		Disables all Wi-Fi. Reboot required to re-enable Wi-Fi.
BCM4375B1	Samsung Galaxy S10/S10e/S10+/S20	Android 10	Apr 13 2018	`0x650200`		Disables Wi-Fi until disabling Bluetooth, going to airplane mode, and re-enabling them.
BCM4377B3	MacBook Pro/Air 2019–2020	macOS 10.15.1–10.15.5	Feb 28 2018	`0x650400`		System freeze and panic without crash log or with `x86 CPU CATERR detected`.
BCM4364B3	MacBook Pro/Air 2019–2020	macOS 10.15.4–10.15.5	May 9 2018	`0x650600`		System freeze and panic without crash log or with `x86 CPU CATERR detected`.
BCM4378B1	iPhone 11	iOS 13.3	Oct 25 2018	`0x650400`		Kernel panic in `LLC Bus error from cpu1` and reboot.

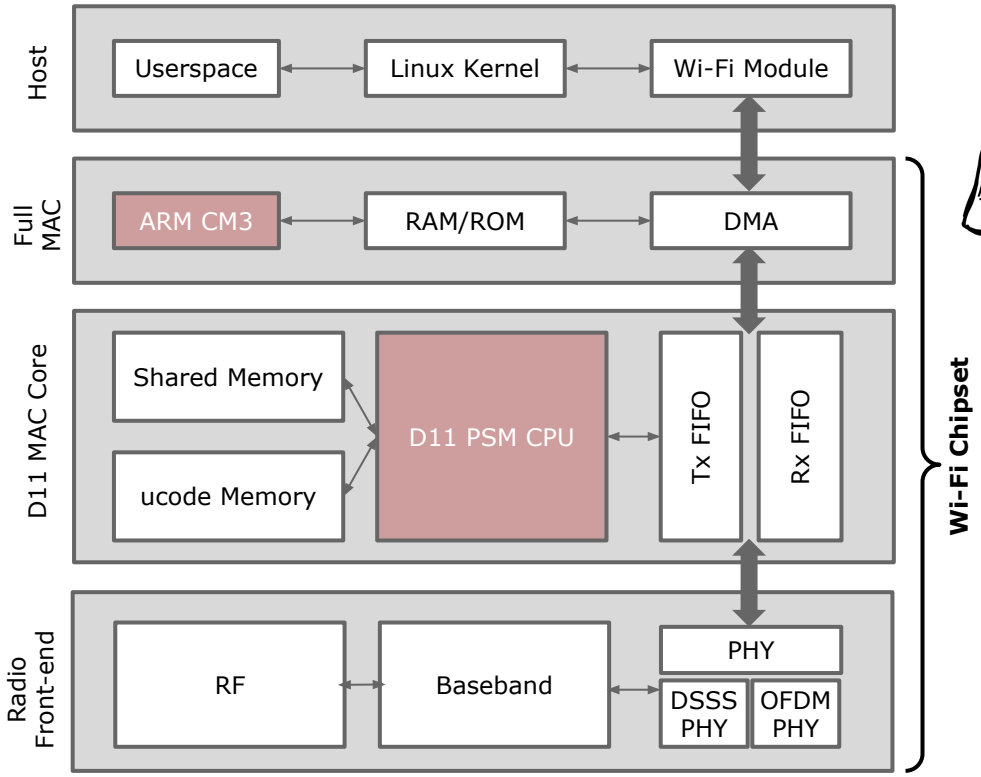
Wi-Fi D11 Core

Broadcom Wi-Fi Architecture

Quite the same real-time architecture since 2003:

- Initial version: **Soft MAC** Linux host talks directly with low level stuff.
- Newer versions: **Full MAC** additional ARM core offloads almost all operations.





Since BCM94303 (2003) and BCM94318E (2006), chipset initially called Airforce One

D11 Core: A Specialized Microcontroller

Runs ucode, instruction set very proprietary, never seen in other architectures

- 8 bytes fixed-length instructions
- three operands instructions plus very weird bit-oriented operators
- tightly connected to PHY hardware
- example from the main loop:

```
jext    EOI (COND_RX_PLCP), rx_plcp  // Preamble (Physical-layer convergence protocol)
jext    COND_RX_COMPLETE, rx_complete
jext    EOI (COND_RX_BADPLCP), rx_badplcp
jnext   COND_RX_FIFOFULL, rx_fifofull
```

- example from send_response code:

```
mov     0x0D4, SPR_TME_VAL6          // ACK indicated by 0xD4
mov     0x035, TX_TYPE_SUBTYPE
je      RX_TYPE_SUBTYPE, TS_PSPOLL, pspoll_frame
```

Existing disassembler/assembler (customized to support later instructions)

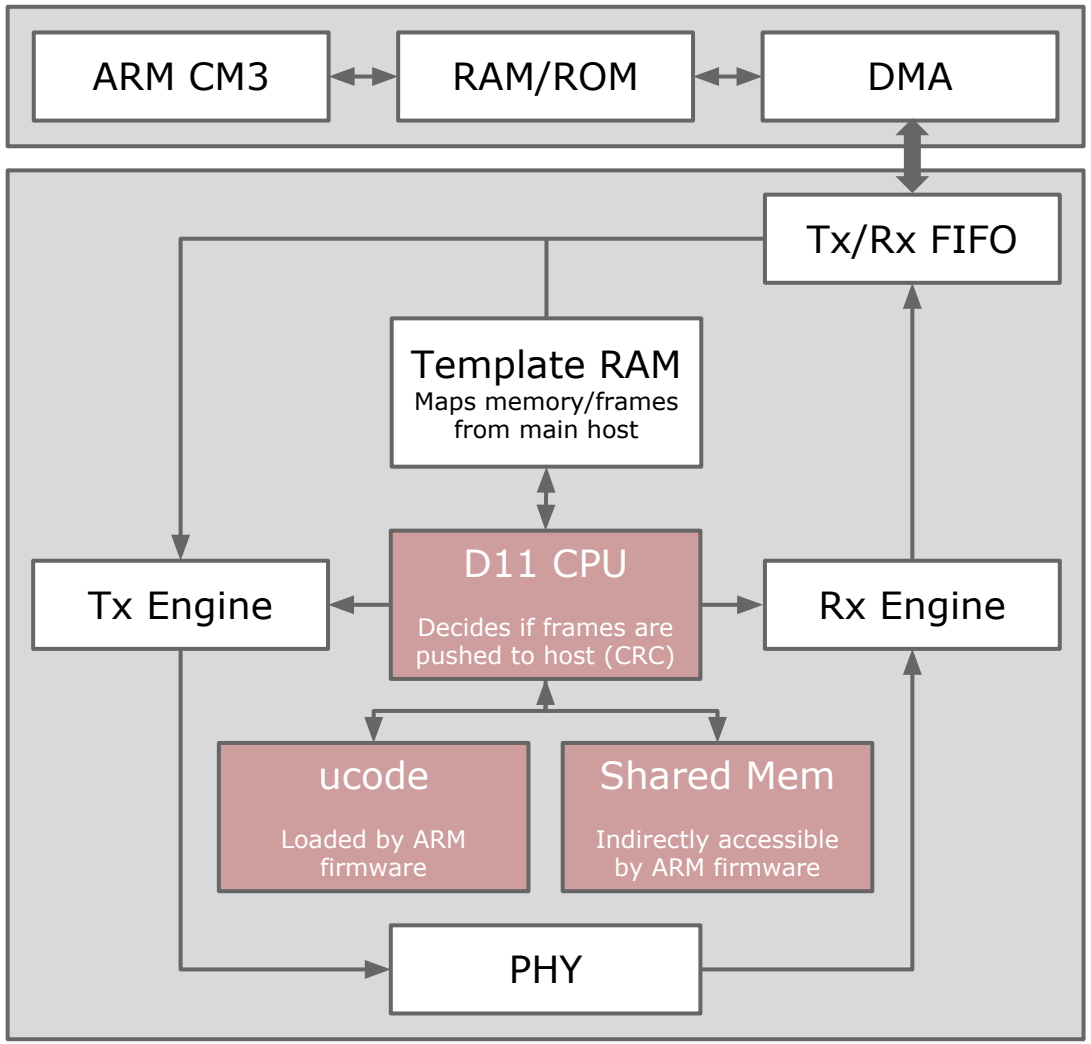
- Michael Büsch created it back in 2007, updated since then within Nexmon.
- hints about registers from many piece of software leaked publicly.

Public ucode tool initially released by Michael Büsch in 2007 (https://bues.ch/cgit/b43-tools.git), continued within Nexmon (https://github.com/seemoo-lab/nexmon).

Inside the D11 Core

Specialized MAC CPU

- Controls Tx and Rx engines
  - channel access scheduling, retransmission
  - filters incoming packets
- Direct access to hardware:
  - PHY registers
  - Radio
  - Interfaces, i.e., coexistence with Bluetooth
- Up to 64kB ucode memory
- Up to 8kB own RAM (called Shared Mem)
- Indirect access to host memory/FIFO
- Sub-µs accuracy
- many interfaces, like SECI...



D11 Coexistence Interface (SECI)

Quite a few registers directly accessible from D11

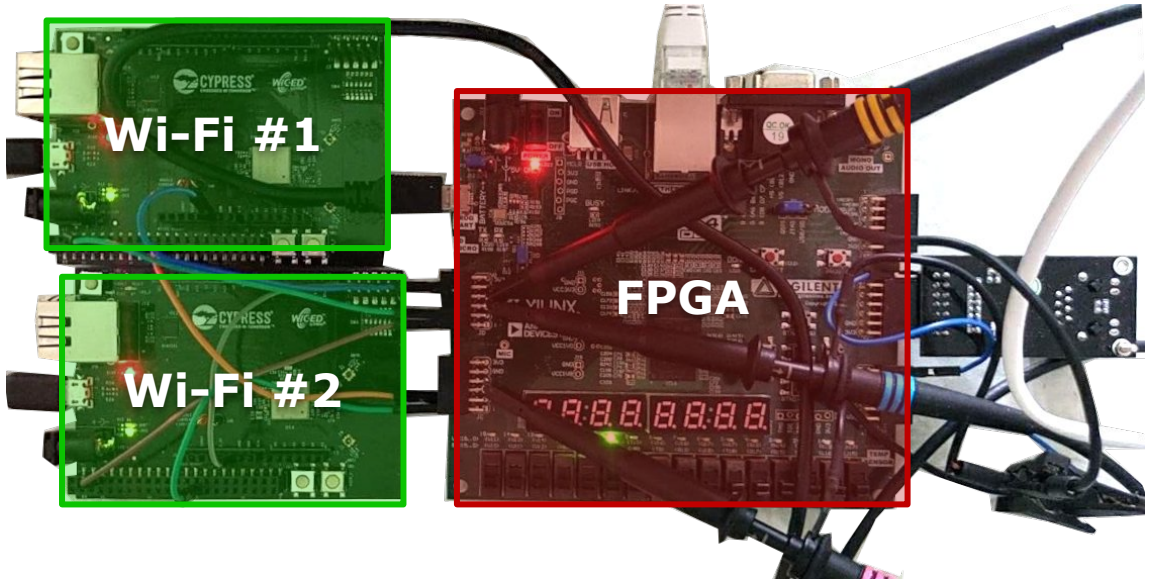
- a 64-bit buffer for rxing messages from Bluetooth (time indications and msg type!)
  - messages are “streamed” from Bluetooth with high rate (every 1.25ms)
- programmable timers
- one register **`btcx_trans_ctrl`** with two bits for telling Bluetooth
  - who has priority
  - who is controlling antenna
  - it is a grant/reject interface

D11 ucode (reference 43909B0 from Cypress):

- 12% of the 47kB ucode for coexistence

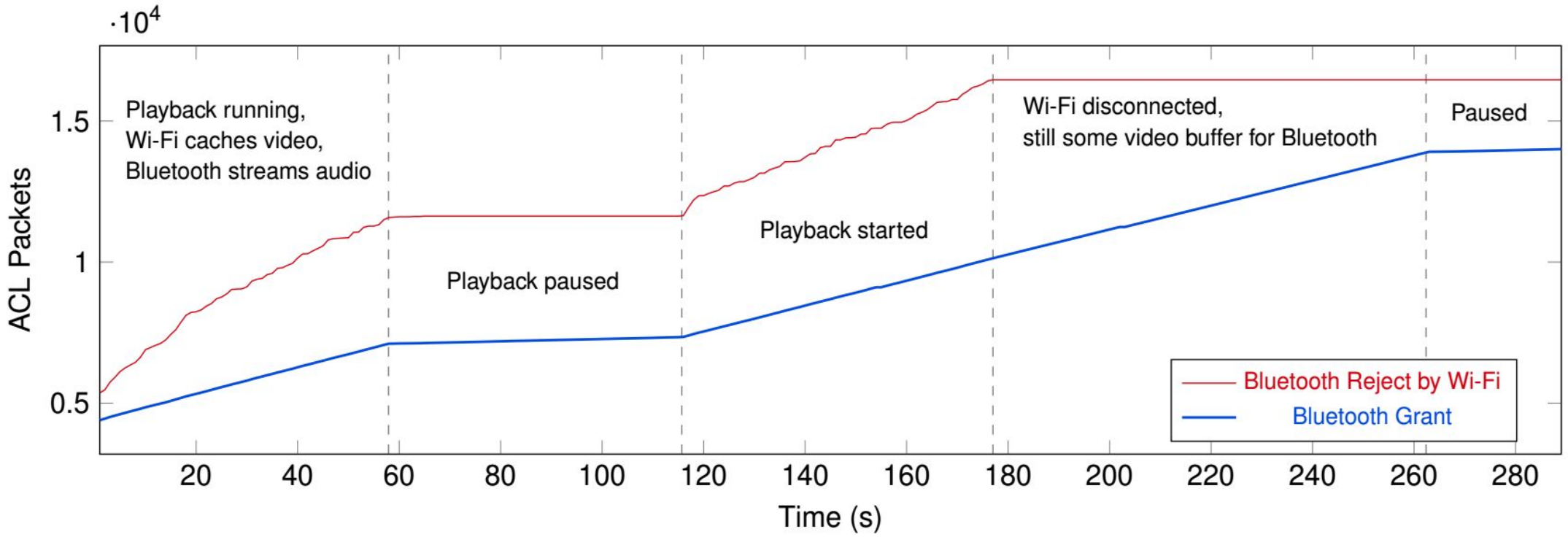
Jitter to Bluetooth measured with FPGA

- receive a frame, wait until the end
- transmit a SECI message
- approximately 200ns std



Breaking the Grant/Reject Scheme

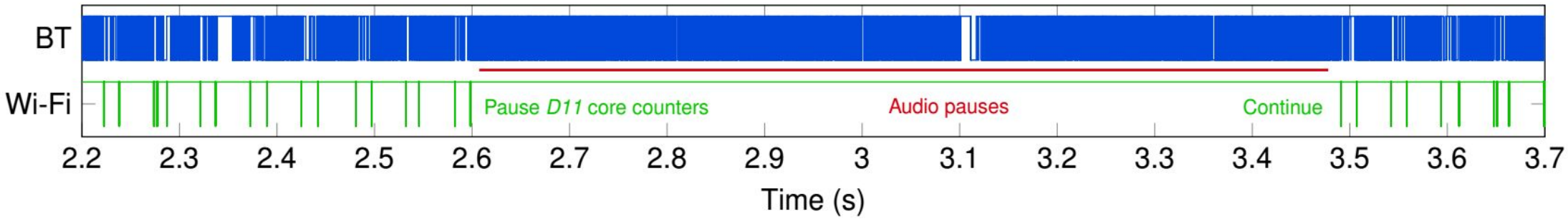
Bluetooth Grant and Reject Counters



Denial of Service Wi-Fi→BT

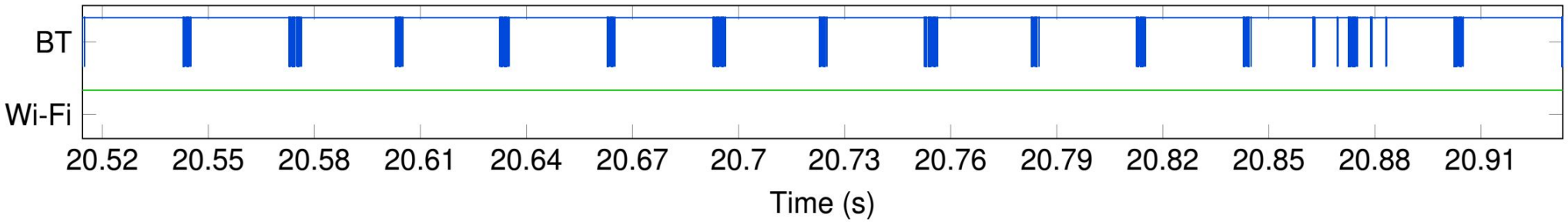
CVE-2020-10370 (reported March 2020)

- When **Wi-Fi** is active and then stops sending SECI messages, **Bluetooth stops** transmitting packets.



Observe SECI in Wi-Fi

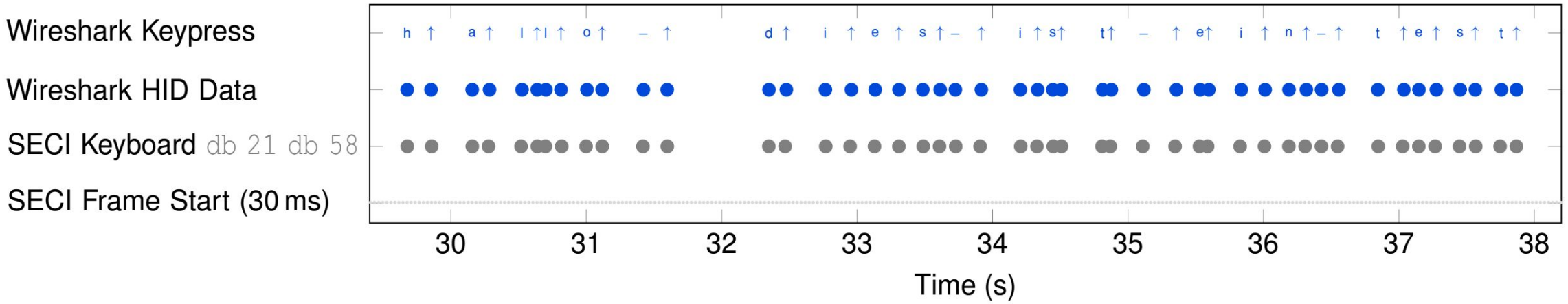
Let’s take a closer look!



c) SECI while typing on a Bluetooth keyboard, Wi-Fi inactive.

Bluetooth keyboard connected, Wi-Fi is idle. Bluetooth sends a message every **30ms** and Wi-Fi is sleeping.

Accurate Key Timings



Keypress timings as observed on logic analyzer (●) when filtering for action `db 21 db 58`. SECI time resolution is indicated by filtering for the frame start `db 03 db 22`, which is each 30 ms for the analyzed keyboard. The aligned *Wireshark* trace is observed on the host and contains the decoded keypresses in addition to the slightly inaccurate timings (●).

Information Disclosure Side Channel

CVE-2020-10369 (reported March 2020)

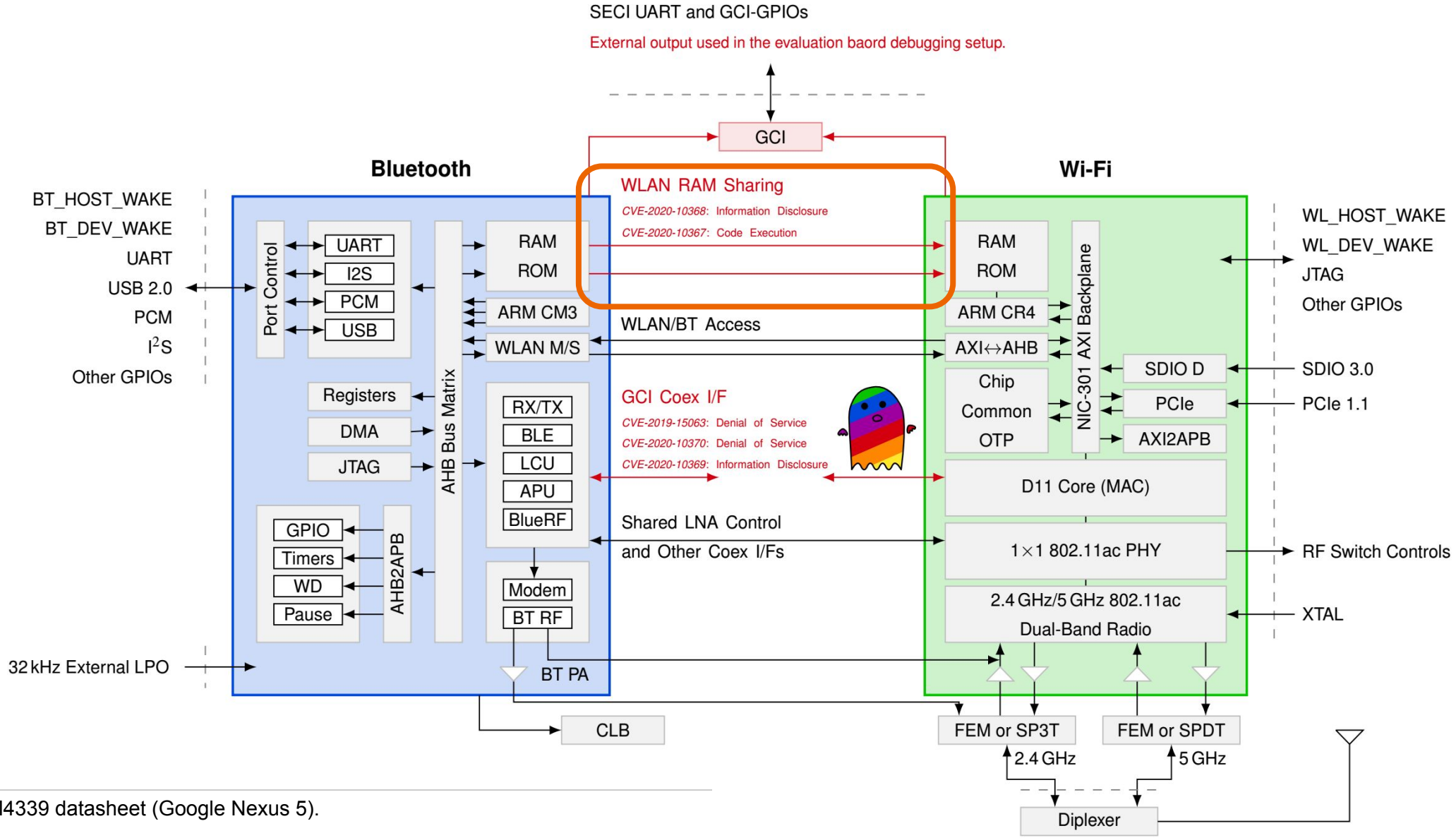
- Each **Bluetooth** Human Interface Device (HID) event generates a SECI message.
- HID devices exist in different event timing variants, the keyboard under test had 30ms, but other keyboards have 12.5ms, 15ms, etc.
- SECI messages are polled every 1.25ms by the **Wi-Fi** D11 core.
- The SECI message for keep alive packets is different from the SECI message containing a HID keystroke.

  → **Infer keystroke timings and keypress amounts.**

WLAN RAM Sharing

When you spent too much time looking for side channels...

RAM sharing??! Only one direction?



From the BCM4339 datasheet (Google Nexus 5).

Where is the shared RAM?

- Bluetooth-only chips with coexistence interface?

  Cypress WICED Studio contains partial symbols for CYW20719, CYW20735, CYW20819 including register mappings, but nothing in there.

- Bluetooth/Wi-Fi combo chips?

  But they also forgot the **symbols** of one **MacBook Pro** (2016 model).

  `wlan_buf_*` ... let's go for this!

Information Disclosure

CVE-2020-10368 (reported March 2020)

- **Bluetooth** can **read** information from the Wi-Fi RAM starting at register `0x680000`. This is mapped to **Wi-Fi** `0x180000`. This range starts with a packet buffer.

// TODO insert unicorn here !

Code Execution

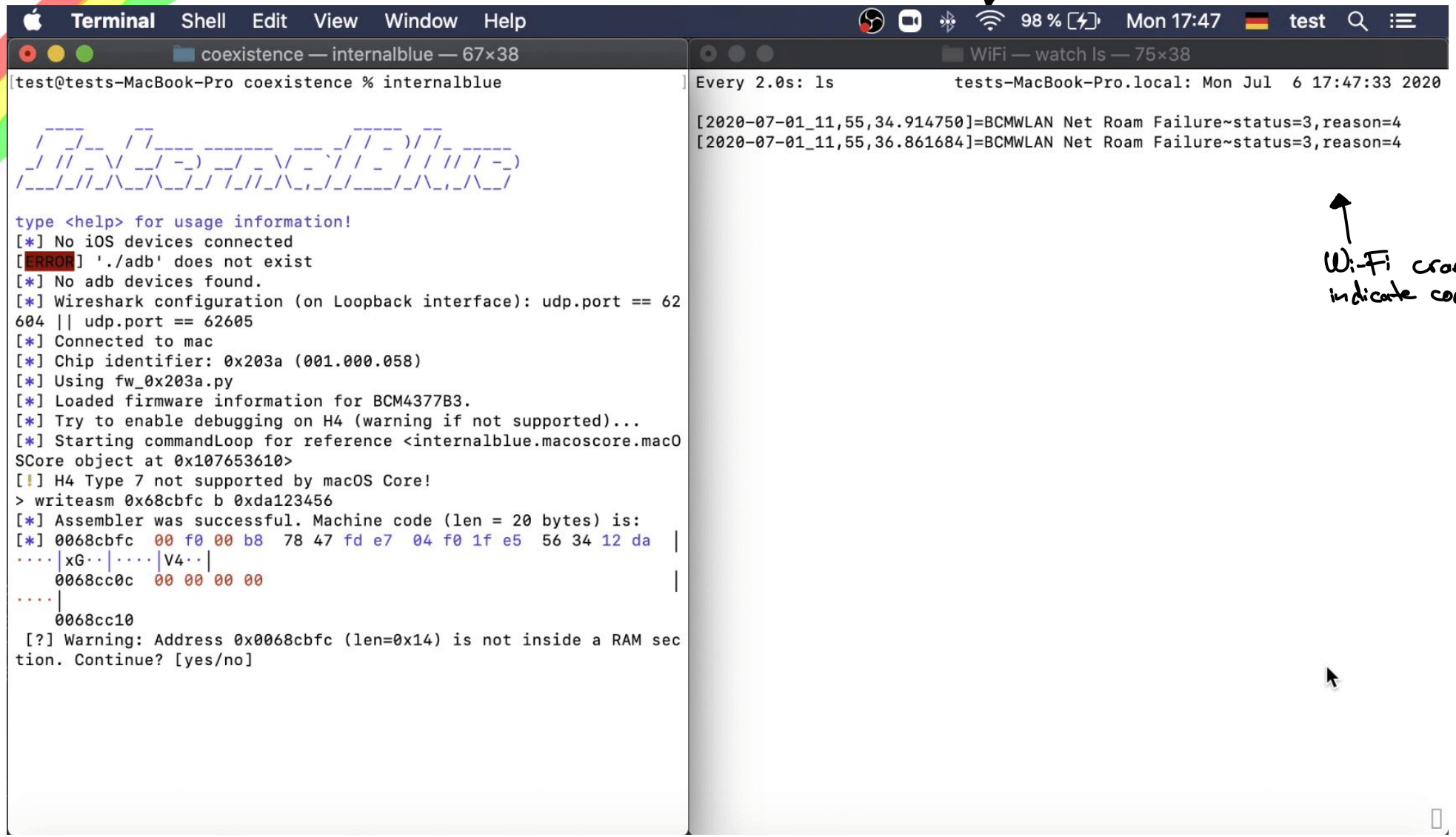
CVE-2020-10367 (reported March 2020)

- **Bluetooth** can write data to the Wi-Fi RAM starting at register `0x680000`. This is mapped to Wi-Fi `0x180000`.
- At `0x181000`, Wi-Fi contains a function pointer table.
  We can gain **Wi-Fi code execution** on a Samsung Galaxy S10 by writing to `0x681024` in Bluetooth.

```
CONSOLE: 000288.686 THREADX TRAP INFO:
CONSOLE: 000288.686 Thread: main_thread(ID:0x54485244) run cnt:7792
CONSOLE: 000288.686 Thread: Stack:002fff24 Start Addr:002fdff0 End Addr:002fffef Size:8192
CONSOLE: 000288.686 Thread: Entry func:001c556d
CONSOLE: 000288.686 Thread: Timer:0022cfcc
CONSOLE: 000288.686
CONSOLE: FWID 01-a4172c0
CONSOLE: flags 30040007
CONSOLE: 000288.686
CONSOLE: TRAP 3(2ffeb8): pc 67452300, lr 19b569, sp 2fff10, cpsr 68000193, spsr 68000033
CONSOLE: 000288.686   ifsr 0, ifar 67452300
CONSOLE: 000288.686   srpwr: 0x100b0000 clk:0xb0040 pmu:0x13e 0x5fcbc7df 0x0
CONSOLE: 000288.686   r0 2e15a8, r1 2c96a4, r2 2ca298, r3 0, r4 2c9708, r5 19c46f, r6 467ae
CONSOLE: 000288.686   r7 40, r8 28bde0, r9 2f9224, r10 2fdff0, r11 0, r12 67452300
CONSOLE: 000288.686
CONSOLE:    sp+0 00000000 13d75f00 002d3a84 0028bde0
CONSOLE: 000288.686   sp+10 00299b74 00000000 0022d084 0028bde0
CONSOLE:
CONSOLE: 000288.686 sp+20 0019c46f
CONSOLE: 000288.686 sp+3c 00195a55
CONSOLE: 000288.686 sp+54 0019c46f
```

...also on macOS, MBP 2019/2020 (BCM4377)

OMG Wi-Fi is restarting!!!11



Execute this! →

Wi-Fi crash logs indicate code execution :^)

CVE-2020-10367 and -10368: A few devices...

Chip	Device	OS	Build Date	`wl_buff` in Firmware	Vulnerable
BCM4335C0	Nexus 5	Android 6.0.1	Dec 11 2012	×	?
BCM4345B0	iPhone 6	iOS 12.4	Jul 15 2013	×	✓ ✸
BCM43430A1	Raspberry Pi 3	Raspbian Buster	Jun 2 2014	×	?
BCM4345C0	Raspberry Pi 3+/4	Raspbian Buster	Aug 19 2014	✓	✓
BCM4358A3	Samsung Galaxy S6, Google Nexus 6P	Lineage OS 14.1	Oct 23 2014	×	✓
BCM20703A2	MacBook Pro 2016		Oct 22 2015	✓	●
BCM4355C0	iPhone 7	iOS 13.3	Sep 14 2015	×	✓
BCM4347B0	Samsung Galaxy S8	Android 8.0.0	Jun 3 2016	✓	✓ ✸
BCM4347B1	iPhone 8/X/XR	iOS 13.3	Oct 11 2016	✓	✓ ✸
BCM4375B1	Samsung Galaxy S10/S10e/S10+	Android 9	Apr 13 2018	×	✓ ✸
BCM4375B1	Samsung Galaxy S10/S10e/S10+/S20	Android 10	Apr 13 2018	×	✓
BCM4377B3	MacBook Pro+Air, 2019–2020	macOS Catalina 10.15.1–10.5.5	Feb 28 2018	×	✓ ✸
BCM4364B3	MacBook Pro+Air, 2019–2020	macOS Catalina 10.15.4–10.5.5	May 9 2018	×	✓
BCM4378B1	iPhone 11	iOS 13.3	Oct 25 2018	×	✓

? Mentioned in datasheet but probably different mapping, did not crash in our test.

● Likely vulnerable but no physical device available for testing.

✸ Kernel panic observed on the operating system.

PCIe

When you have no idea what you're doing…

/Library/Logs/CrashReporter/CoreCapture/WiFi — sleep | /Library/Logs/CrashReporter/CoreCapture/WiFi — watch ls

```
Every 2.0s: ls                                    MacBook-Pro.local: Mon Jul  6 17:33:31 2020

[2020-07-06_13,17,29.925442]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x56EC6,LR=0x56EB7
[2020-07-06_13,17,46.927561]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16DB7F,LR=0x16E093
[2020-07-06_13,18,07.548388]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16DB7F,LR=0x16DB95
[2020-07-06_13,18,25.291703]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x8CB2,LR=0x1B2C1D
[2020-07-06_13,18,28.660271]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16E4A7,LR=0x16E4A7
[2020-07-06_13,18,39.909139]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x1AAFC8,LR=0x3843F
[2020-07-06_13,18,44.205612]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16E3D7,LR=0x2AAF
[2020-07-06_13,19,17.785250]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16DB7F,LR=0x16E093
[2020-07-06_13,19,51.322313]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16DB7F,LR=0x16E093
[2020-07-06_13,20,47.611553]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0xB9836,LR=0xBAEF1
[2020-07-06_13,21,04.703997]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16CA10,LR=0x16D2E7
[2020-07-06_13,22,07.809095]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16DB7F,LR=0x16E093
[2020-07-06_13,22,08.104307]=BCMWLAN Core Wake Reason Unexpected~Failed to get cached FW wakeup data
[2020-07-06_13,22,12.680043]=watchdog@BCMWLAN Bus Unexpected Phase Bit~phase=0,expect=1,r=249,w=0
[2020-07-06_13,26,49.153053]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16DB7F,LR=0x16E093
[2020-07-06_13,34,14.164190]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16E3D7,LR=0x898F
[2020-07-06_13,34,18.321268]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16E4A7,LR=0x16E4A7
[2020-07-06_13,34,21.571824]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16E3D7,LR=0x19DB0D
[2020-07-06_13,34,24.782521]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16E3D7,LR=0x5B94F
[2020-07-06_13,34,59.115526]=watchdog@BCMWLAN Cmdr Pending Queue Stall~cmd=WLC_SET_VAR: event_msgs_ext
[2020-07-06_13,35,02.160704]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16E3D7,LR=0x898F
[2020-07-06_13,35,13.418129]=watchdog@BCMWLAN Cmdr Outbound Queue Stall~cmd=WLC_GET_VAR: ver
[2020-07-06_13,35,14.762369]=BCMWLAN Failed to Create Debug Ring~
[2020-07-06_14,52,16.248124]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x55F4D,LR=0x79353
[2020-07-06_16,51,14.035242]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16DB7F,LR=0x16E093
[2020-07-06_16,51,57.560663]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16DB7F,LR=0x16E093
[2020-07-06_16,52,09.880882]=watchdog@BCMWLAN Chip Trap~Type=3,PC=0xFECABEB8,LR=0x18A46B
[2020-07-06_16,52,24.051009]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x4DFA2,LR=0x8BC33
[2020-07-06_16,52,41.425674]=watchdog@BCMWLAN Chip Trap~Type=7,PC=0xB9C9E,LR=0xBA349
[2020-07-06_16,52,51.769749]=watchdog@BCMWLAN Chip Trap~Type=3,PC=0xCABEBAFC,LR=0x18A46B
[2020-07-06_16,53,04.368957]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x39100,LR=0x38BE1
[2020-07-06_16,53,18.384791]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16B104,LR=0x16B0E5
[2020-07-06_16,53,33.230057]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16DB7F,LR=0x16E093
[2020-07-06_16,53,50.442298]=watchdog@BCMWLAN Chip Trap~Type=4,PC=0x16DB7F,LR=0x16E093
```

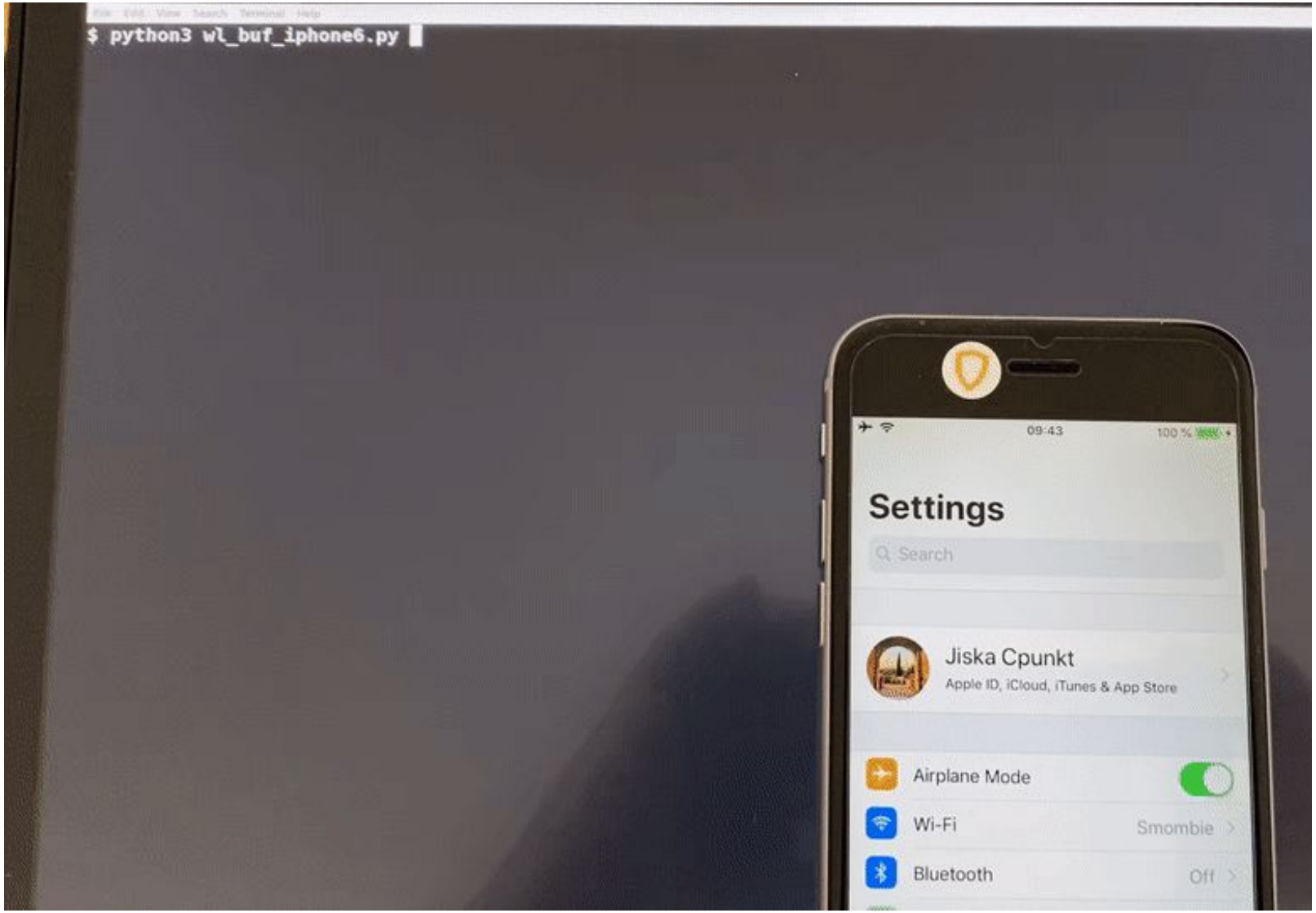
← Code execution on BCM4364B3

Wi-Fi code execution leads to various kernel panics

Kernel panics captured so far:

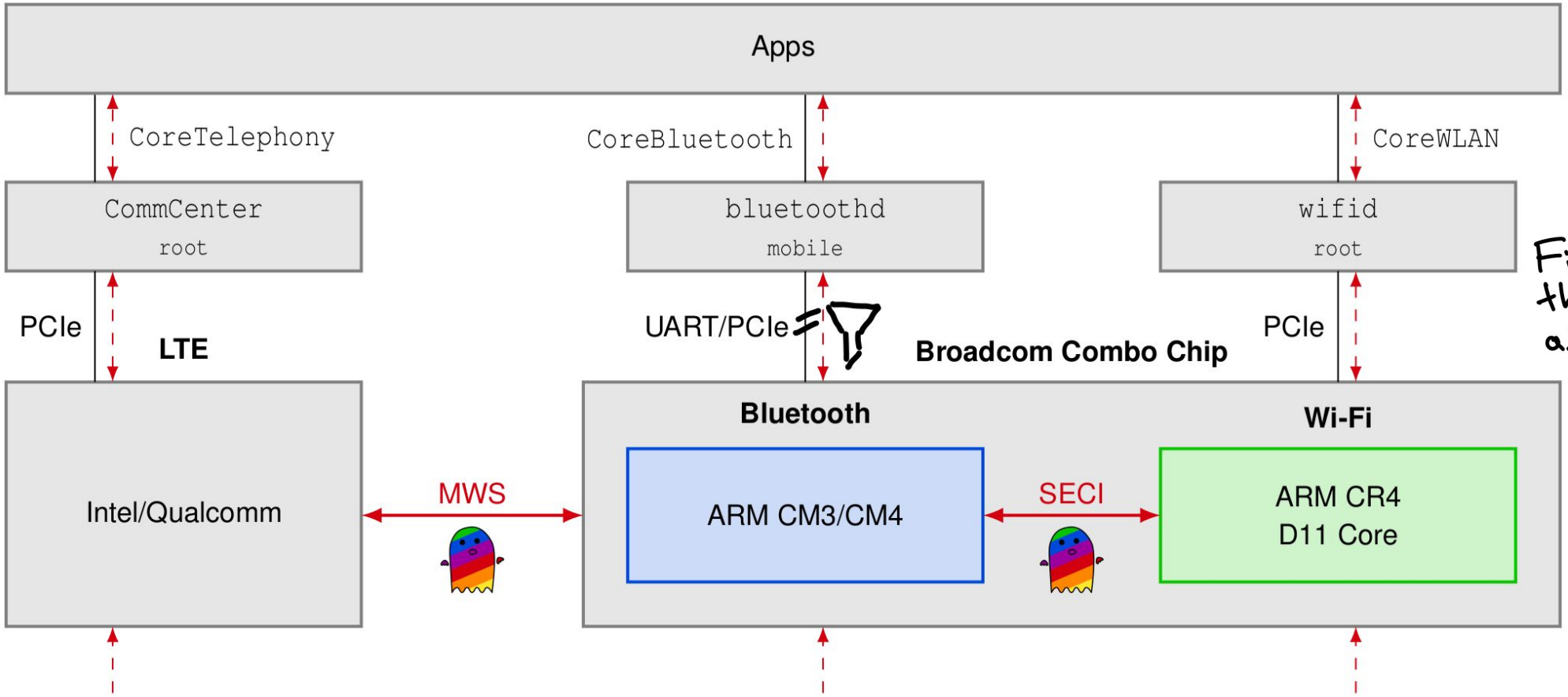
- Samsung Galaxy S10e on Android 9
- iPhone 8 on iOS 13.3, iPhone 6 on iOS 12.4
- ...also macOS but likely another issue in the Bluetooth driver.

iOS Kernel Panic Demo



← Wi-Fi disconnects before panic

The "Patch"

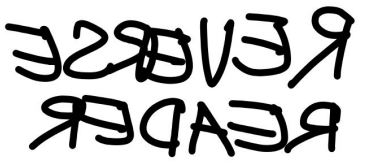


Filter HCI commands that allow memory access after driver initialization

Other Chips

Mobile Wireless Standards: Bluetooth/LTE Coexistence

Wireless Coexistence Interface 2 (WCI-2) Transport Specification



3.1.4 MWS Inactivity Duration message (Type 3)

The MWS Inactivity Duration message is used to send the MWS_INACTIVITY_DURATION signal from the MWS device to the Bluetooth Controller.

The message is sent at the beginning of the MWS inactivity period.

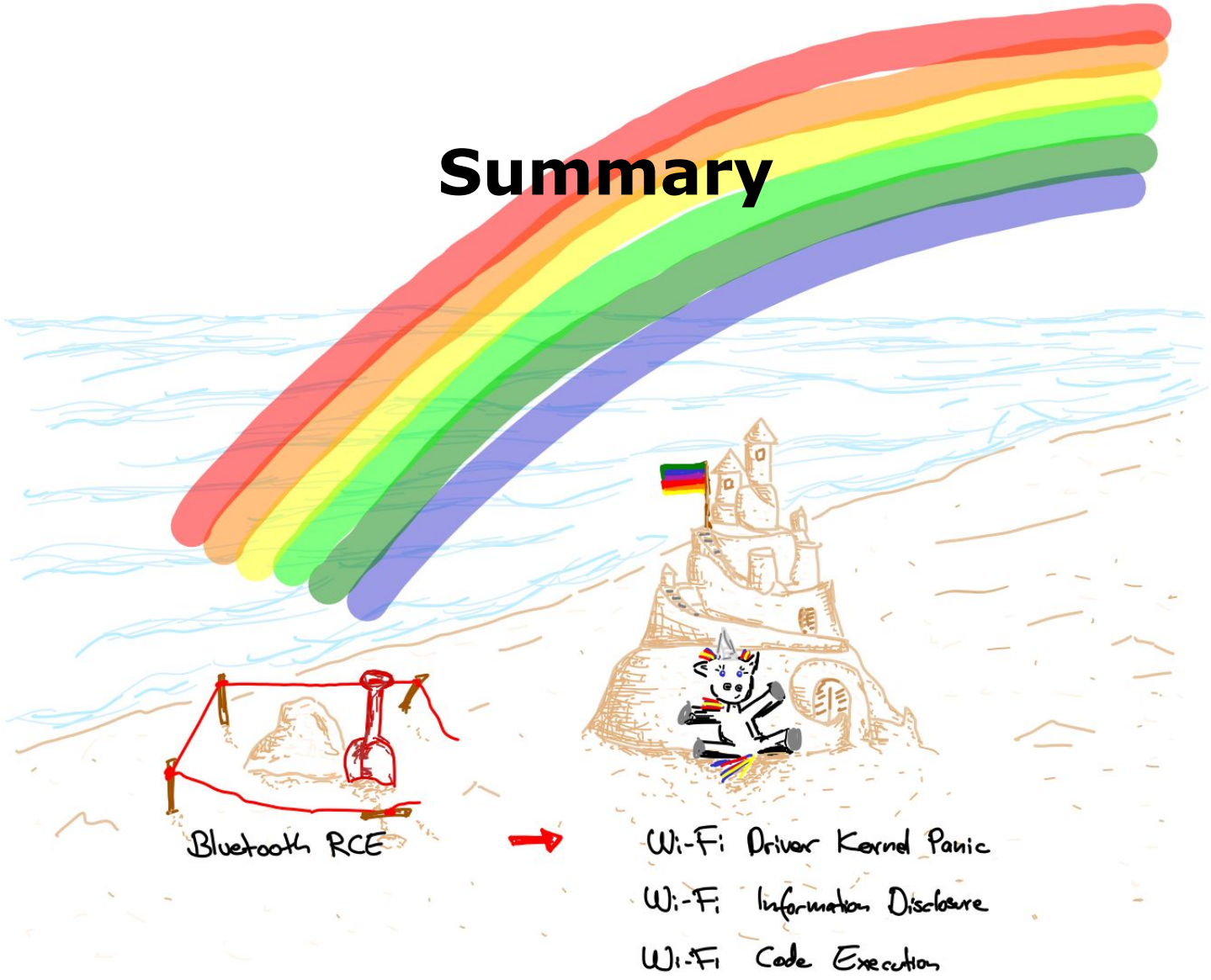
MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
DURATION[0]	DURATION[1]	DURATION[2]	DURATION[3]	DURATION[4]

Table 3.10: MWS Inactivity Duration message

Everyone has proprietary coexistence features \o/

- Asked Broadcom if we can also include other wireless manufacturers into the responsible disclosure process.
- Yes, we can :)
- Forwarded to Intel, MediaTek, Qualcomm, Texas Instruments, Marvell, NXP.
  They all mention similar coexistence interfaces in their datasheets.
- Some wireless chips do not separate wireless cores at all.
  → Not directly vulnerable to Spectra?
  Operating system based side channels might exist...

Summary


Bluetooth RCE
Wi-Fi Driver Kernel Panic
Wi-Fi Information Disclosure
Wi-Fi Code Execution
Bluetooth DoS
Wi-Fi RCE
Bluetooth Information Disclosure

Q&A

Twitter: @naehrdine, @seemoolab

jiska@bluetooth.lol, francesco.gringoli@unibs.it