DEVCORE

Arise from the Wireless: Breaking the Security Barrier in Wi-Fi

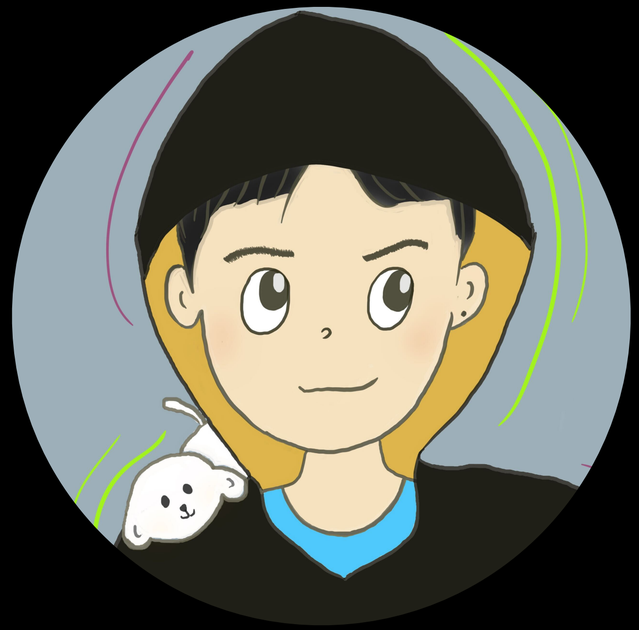
Wei Che Kao (@xiaobye_tw)

weichekao@devco.re

Hexacon 2025 | 2025.10.10

Who am I

- Wei Che Kao (@xiaobye_tw)
- Security Researcher at DEVCORE
- Focus on
  - Wireless Security, Virtual Machine
- Speaker at
  - USENIX Security



DEVCORE

Agenda

- Motivation
- The Beginning of the Journey
  - Wi-Fi 6 MCU - MT7981
  - Wi-Fi 7 MCU - MT7991
- The Forgotten Frame Injection
  - CVE-2025-20674

DEVCORE

Agenda

- One Frame to Break Them All
  - Action Frame - CVE-2025-20711
  - FT Action Frame - CVE-2025-20686
  - Wi-Fi 7 - CVE-2025-20712
  - Beacon Frame - CVE-2025-20685
- Internal Network Party
- Conclusion

DEVCORE

Motivation

Why  ?

Previous Research

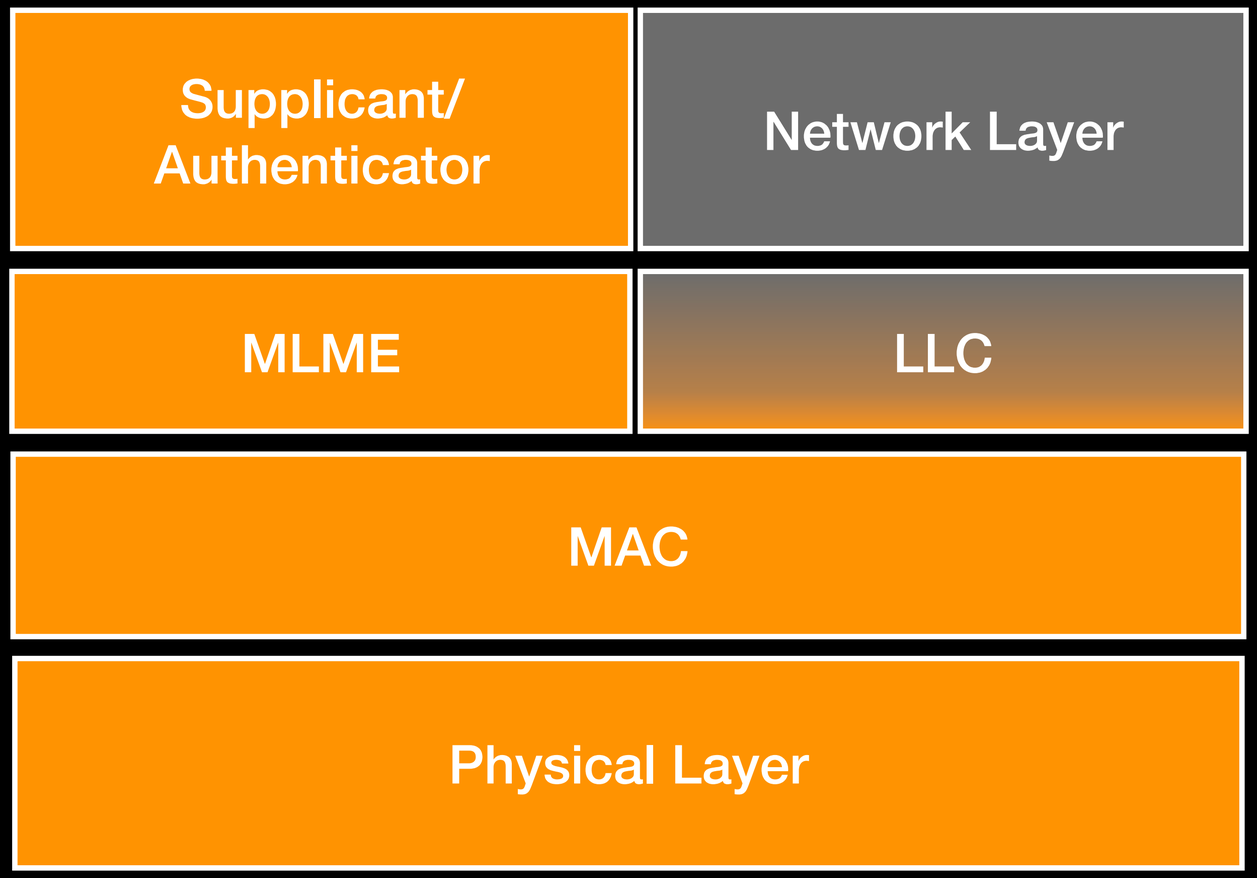
- 2017 - Broadpwn: Remotely Compromising Android and iOS via a Bug in Broadcom's Wi-Fi Chipsets
- 2018 - Researching Marvell Avastar Wi-Fi: From Zero Knowledge to Over-the-Air Zero-Touch RCE
- 2019 - Exploiting Qualcomm WLAN and Modem Over the Air
- 2021 - Fragment and Forge: Breaking Wi-Fi Through Frame Aggregation and Fragmentation
- 2021 - Qualcomm WiFi: Infinity War
- 2022 - BadMesher: New Attack Surfaces of Wi-Fi Mesh Network
- 2022 - Ghost in the Wireless, iwlwifi edition
- 2022 - WIFI Security - From 0 To 1
- 2024 - Listen-Up: Sonos Over-The-Air Remote Kernel Exploitation and Covert Wiretap
- 2025 - Unlock hidden Superpowers in MediaTek Wi-Fi Chips

DEVCORE

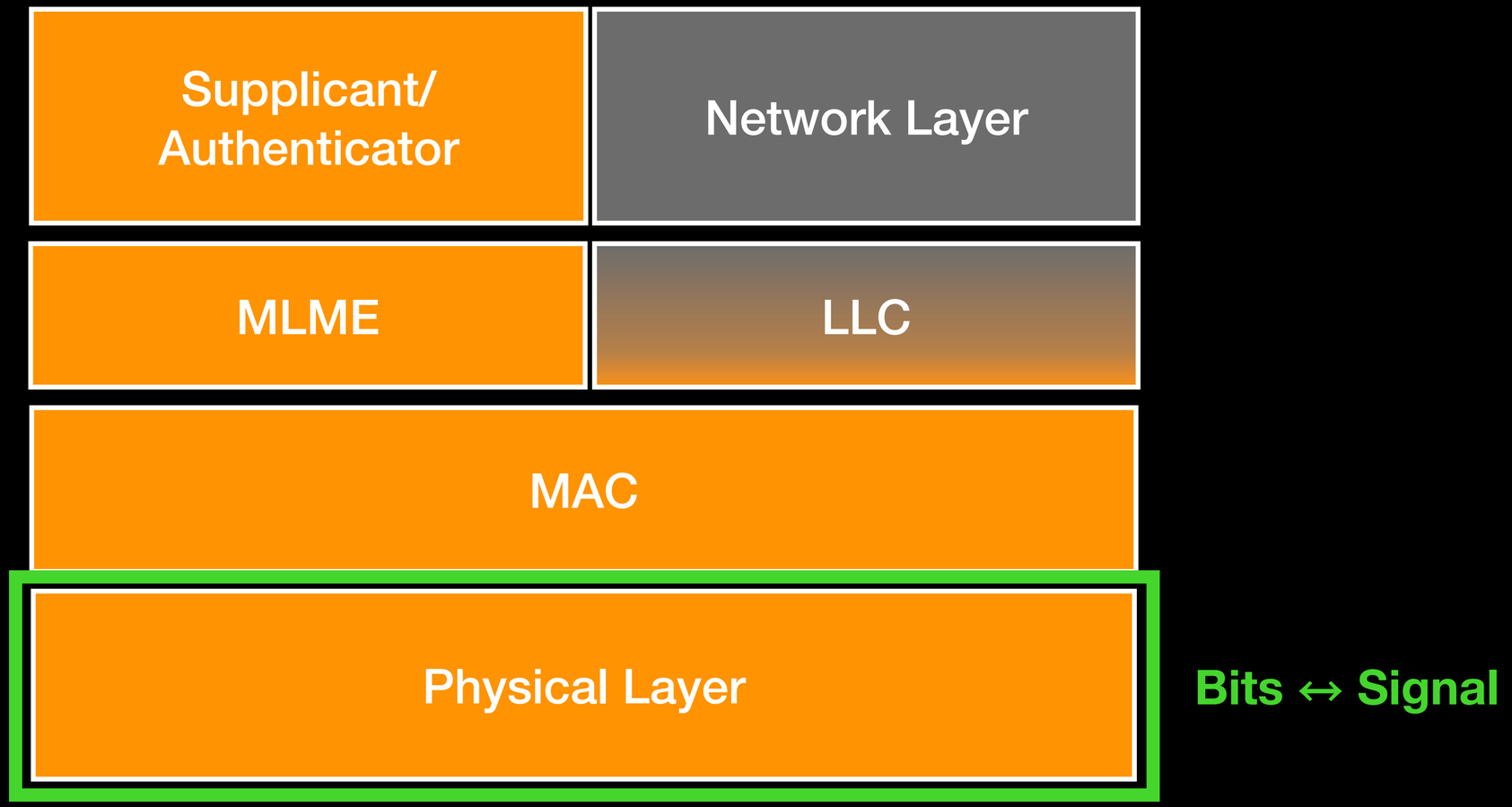
The Beginning of the Journey

DEVCORE

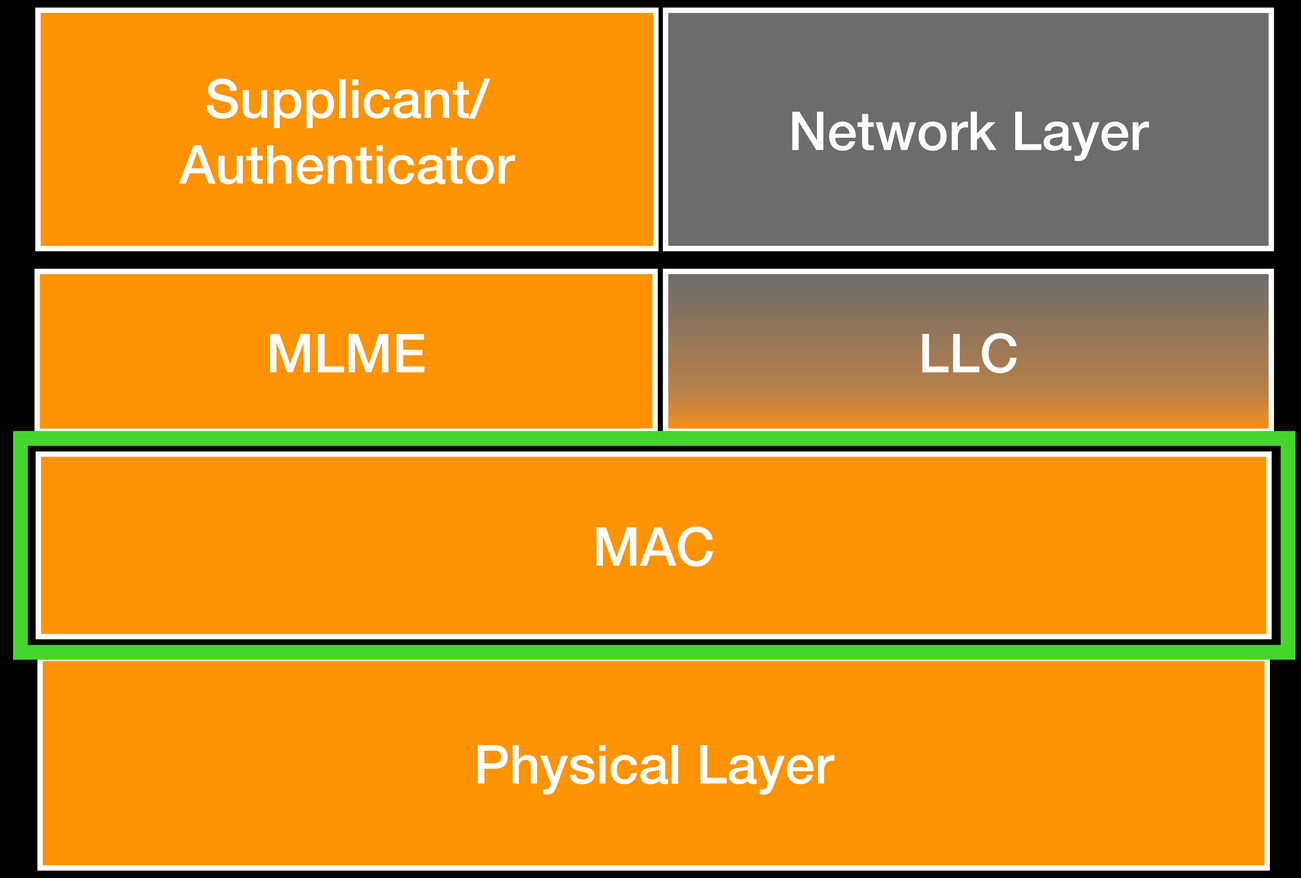
Wi-Fi Protocol Stack



Physical Layer



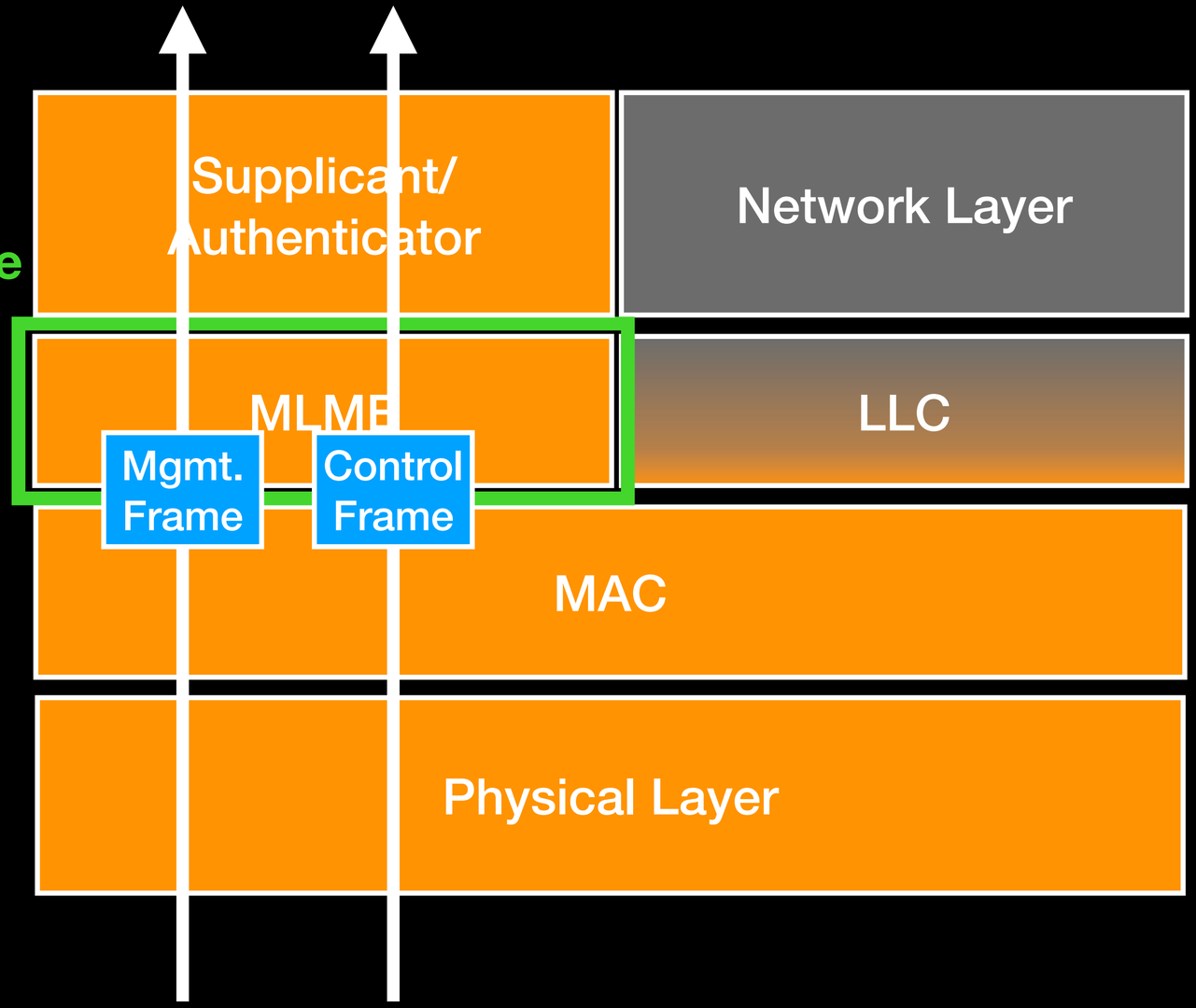
MAC Layer



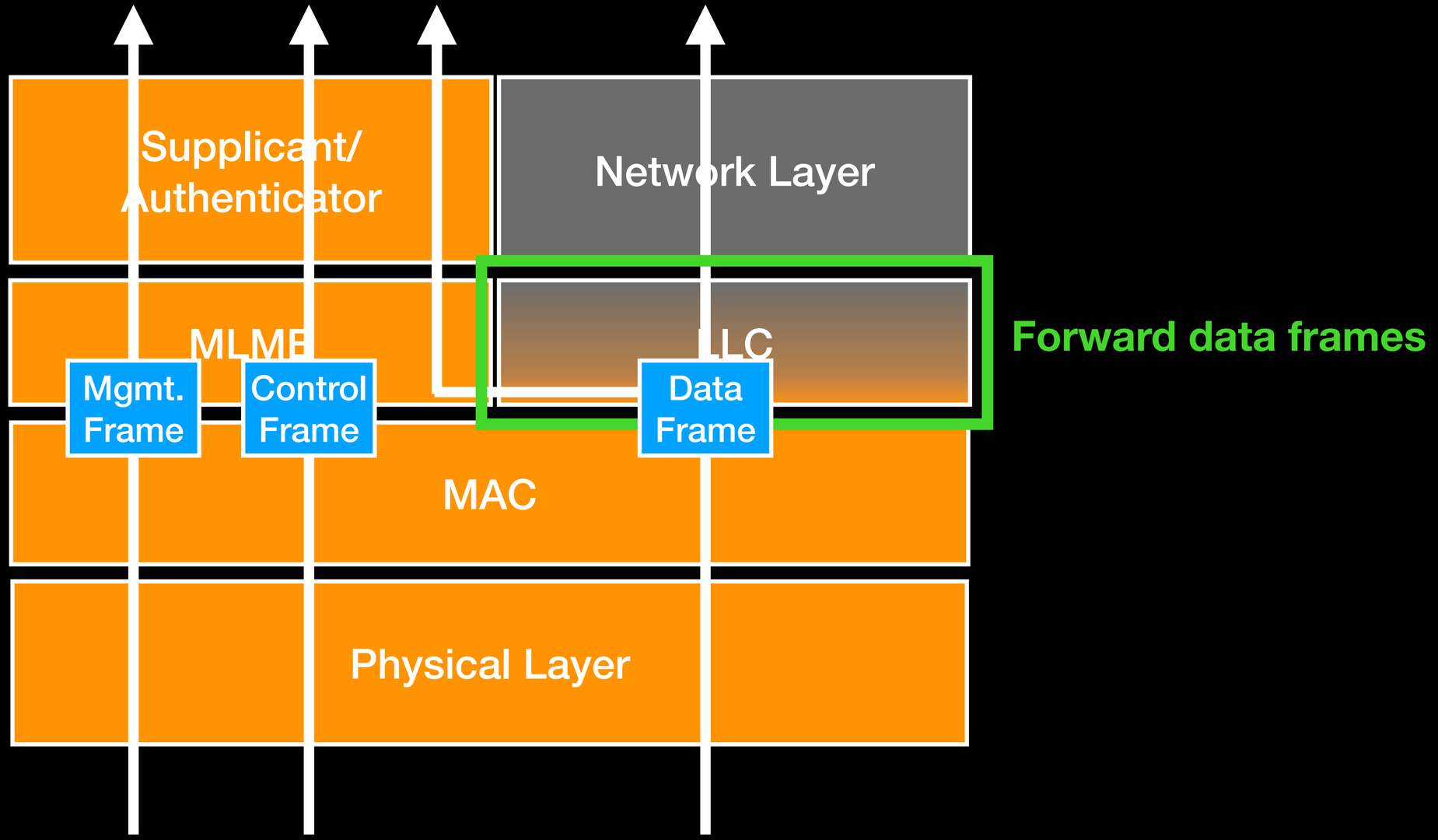
Access PHY
Encapsulation
Decapsulation

MLME Layer

Process Mgmt. frames
Maintains state machine
Handles control OPs

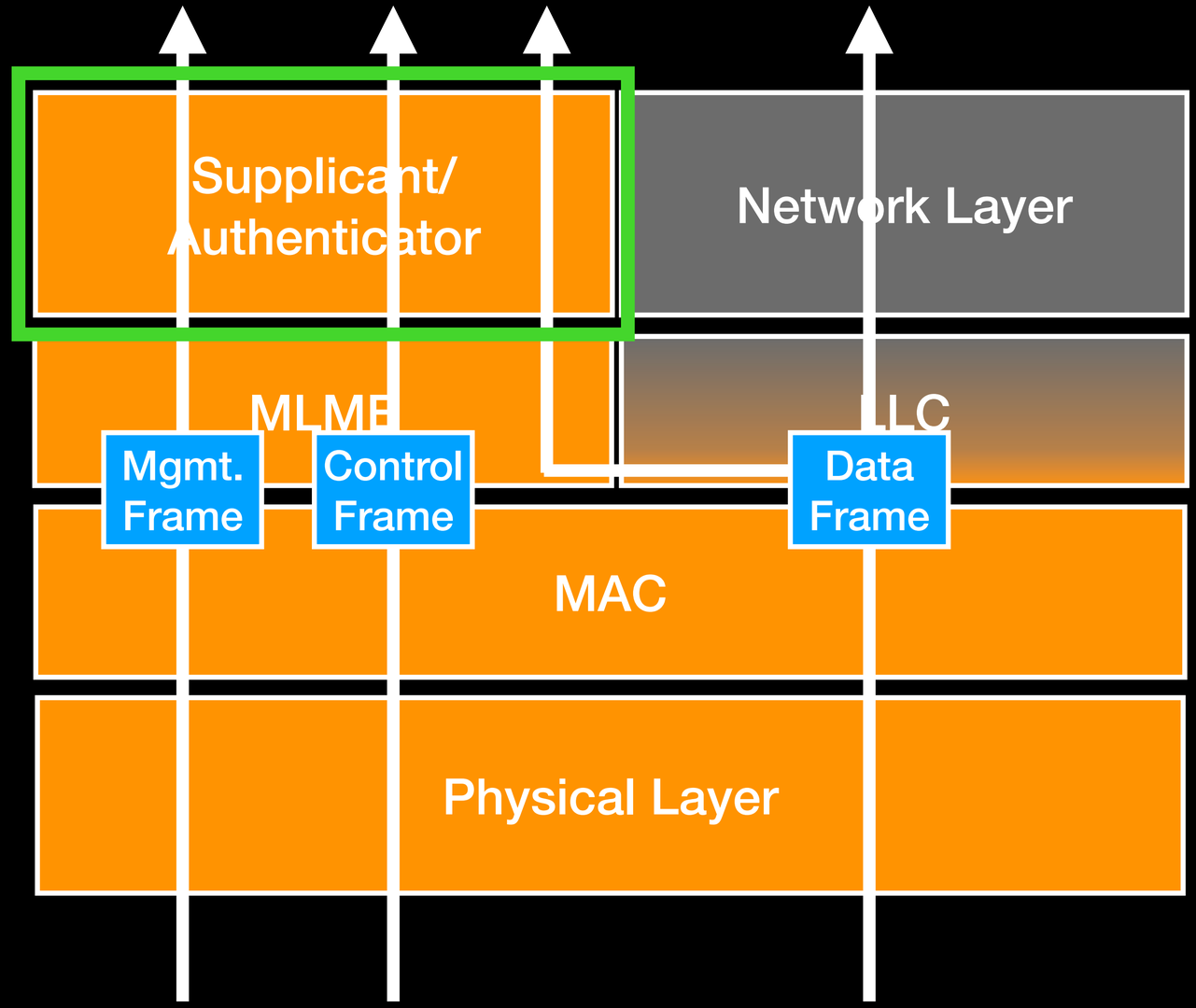


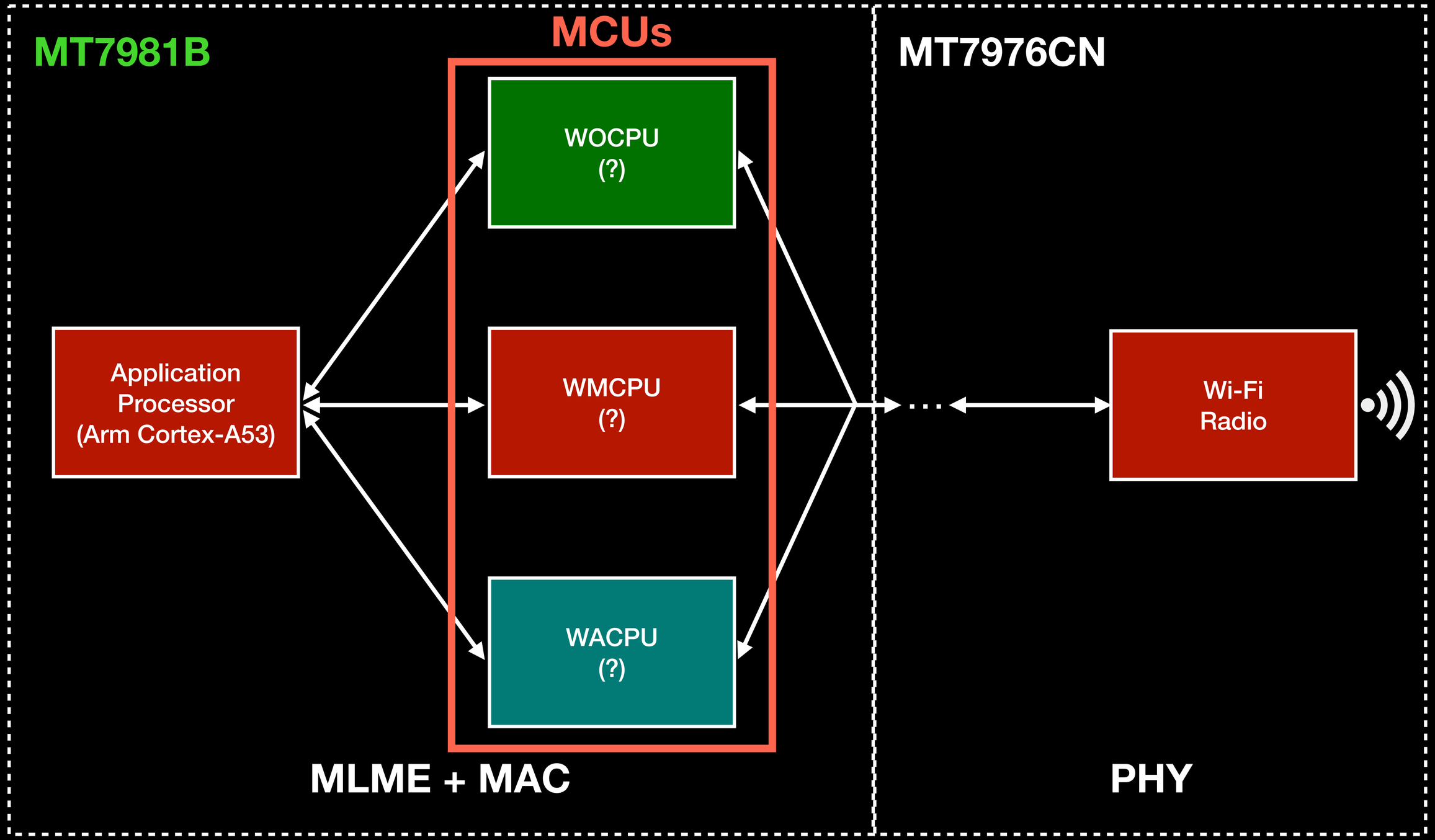
LLC Layer

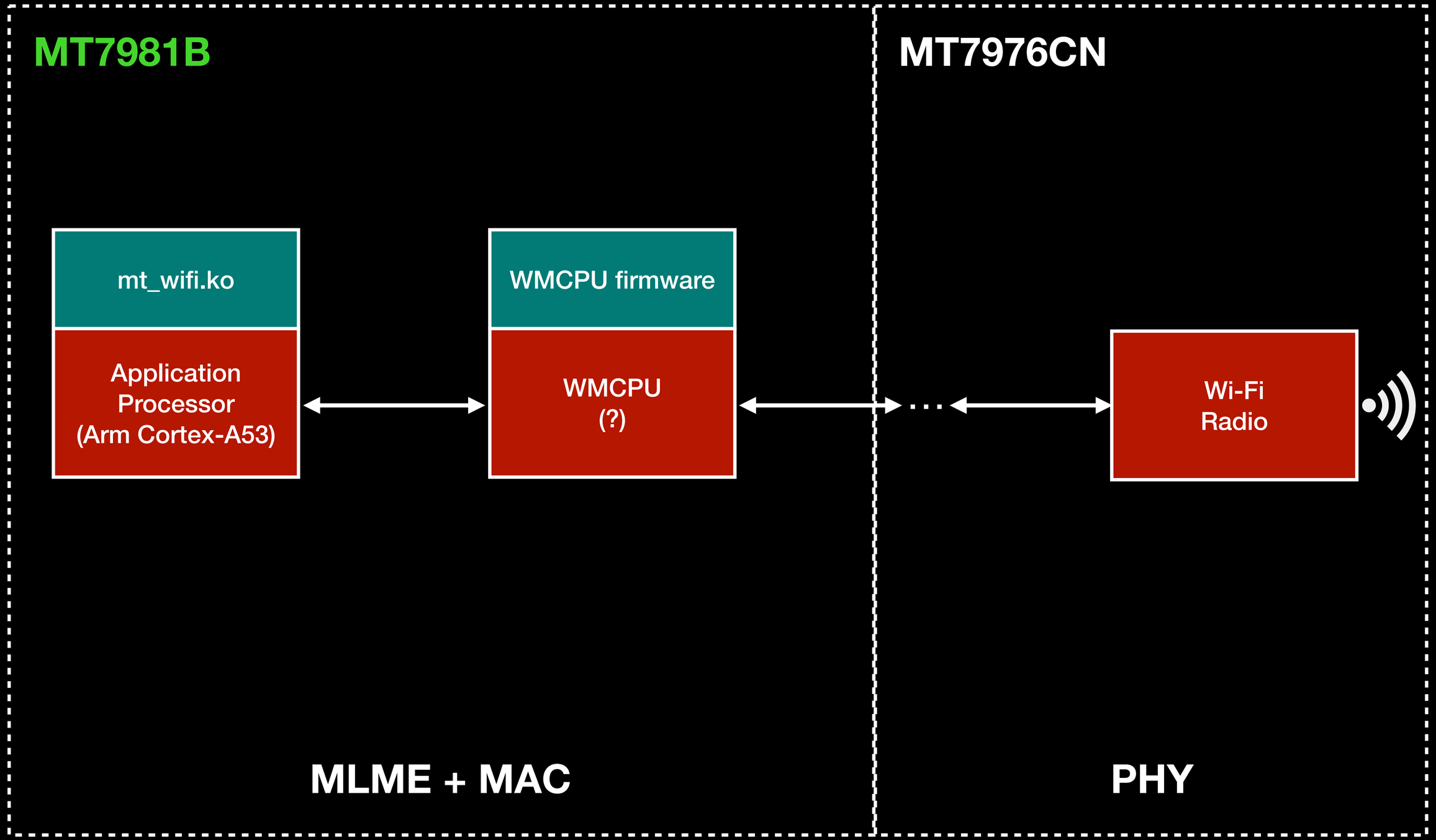

Supplicant/
Authenticator
Network Layer
MLME
LLC
Mgmt.
Frame
Control
Frame
Data
Frame
MAC
Physical Layer
Forward data frames

Supplicant/Authenticator

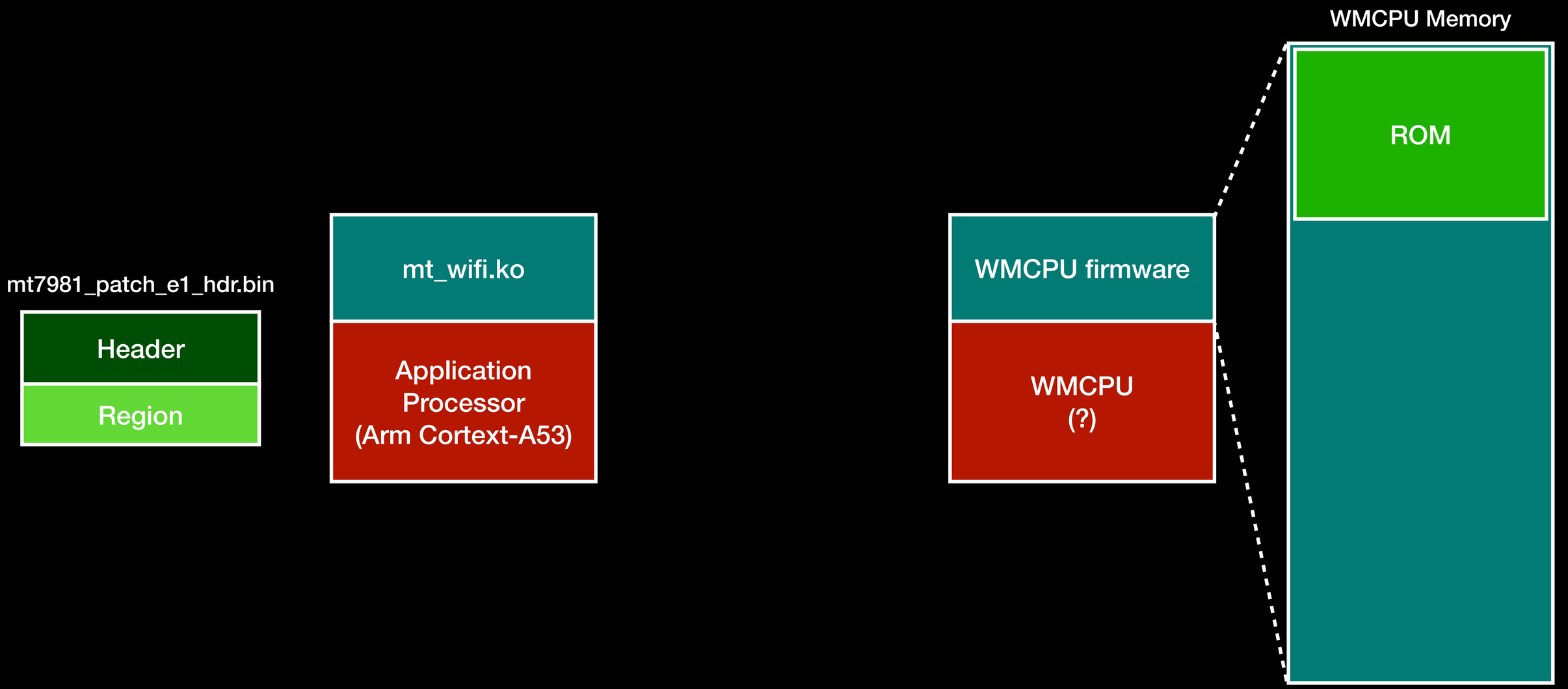
Authentication
Key exchange



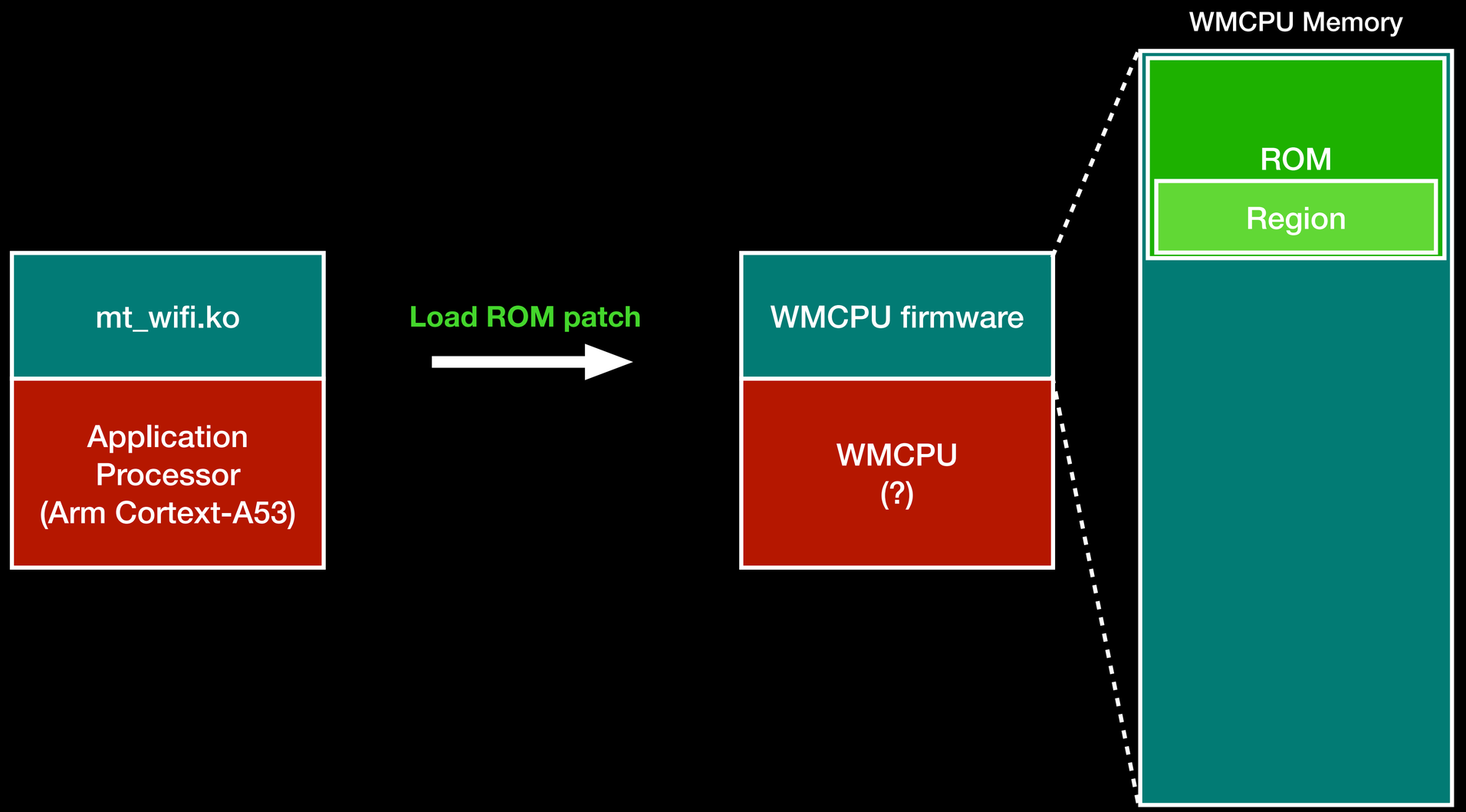

MT7981B
MCUs
MT7976CN
WOCPU
(?)
Application
Processor
(Arm Cortex-A53)
WMCPU
(?)
Wi-Fi
Radio
WACPU
(?)
MLME + MAC
PHY


MT7981B
MT7976CN
mt_wifi.ko
Application Processor (Arm Cortex-A53)
WMCPU firmware
WMCPU (?)
Wi-Fi Radio
MLME + MAC
PHY

Load ROM Patch



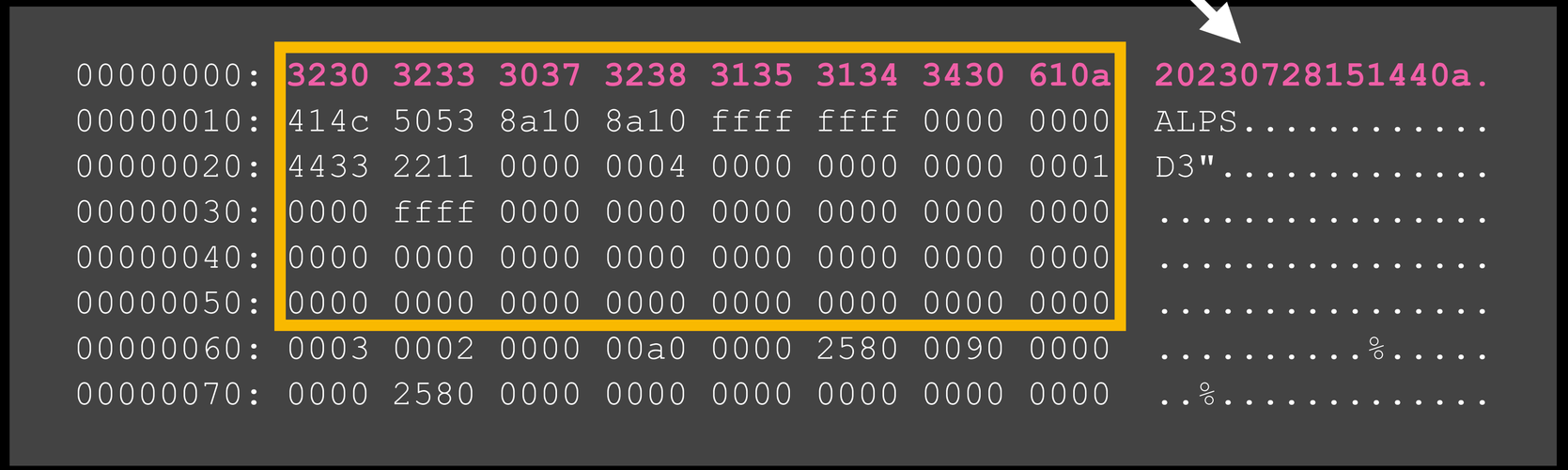
Load ROM Patch



ROM Patch Header

- Patch Header
  - Built Date

It’s built at 2023/07/28/ 15:14:40



ROM Patch Header

- Patch Header
  - Built Date
  - Number of Regions

```
00000000: 3230 3233 3037 3238 3135 3134 3430 610a  20230728151440a.
00000010: 414c 5053 8a10 8a10 ffff ffff 0000 0000  ALPS............
00000020: 4433 2211 0000 0004 0000 0000 0000 0001  D3".............
00000030: 0000 ffff 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 0003 0002 0000 00a0 0000 2580 0090 0000  ..........%.....
00000070: 0000 2580 0000 0000 0000 0000 0000 0000  ..%.............
```

This firmware has 1 region

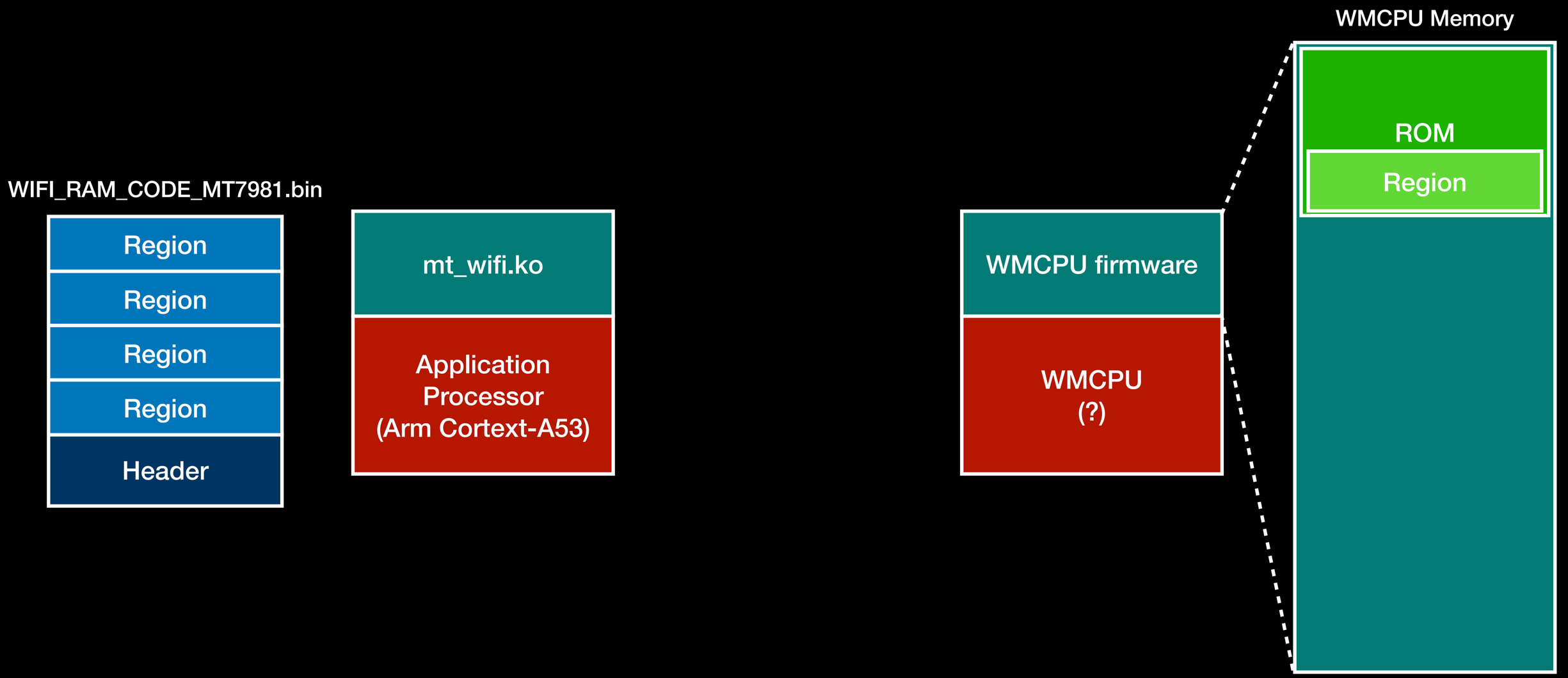
ROM Region Header

- Region Header
  - Address
  - Length

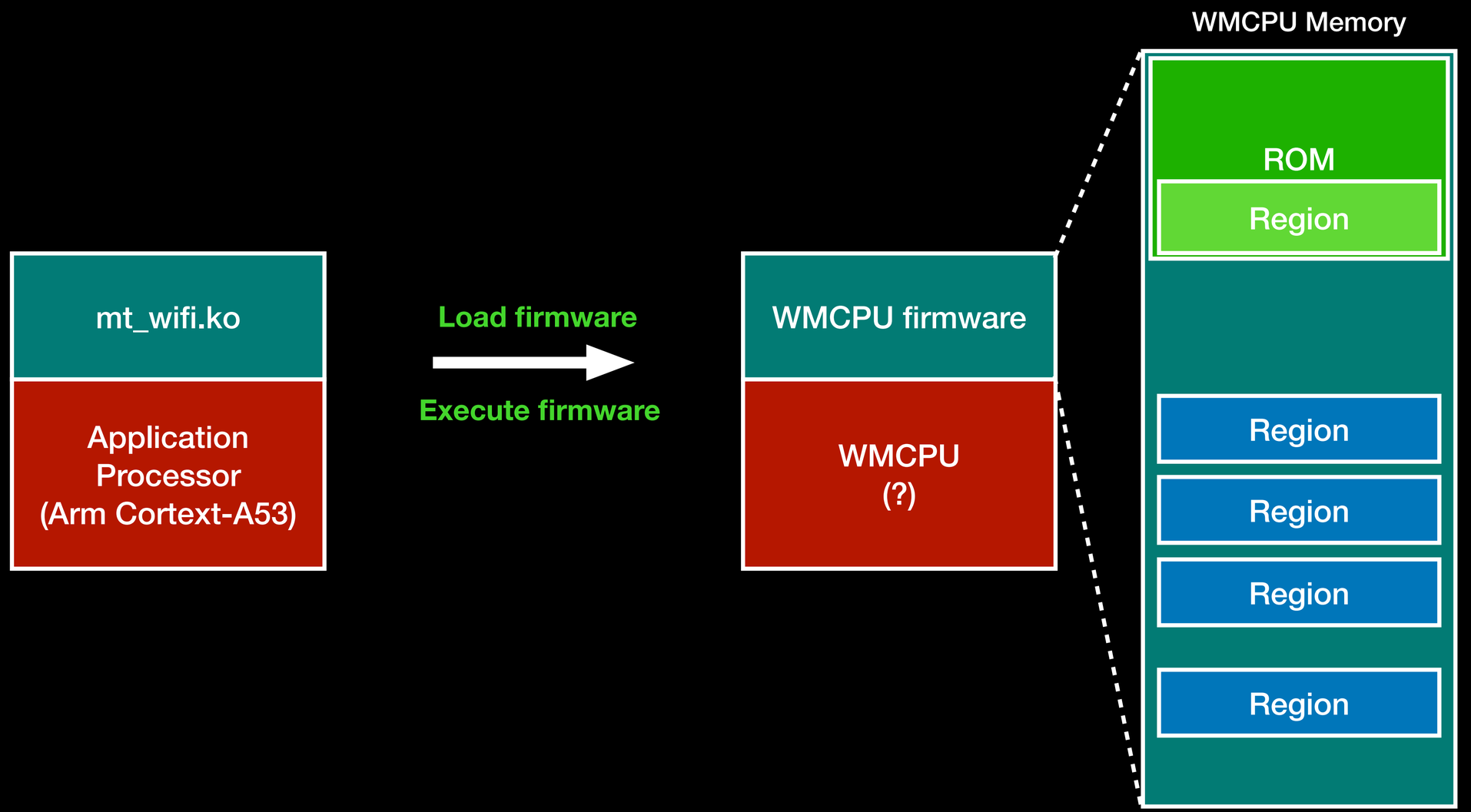
```
00000000: 3230 3233 3037 3238 3135 3134 3430 610a  20230728151440a.
00000010: 414c 5053 8a10 8a10 ffff ffff 0000 0000  ALPS............
00000020: 4433 2211 0000 0004 0000 0000 0000 0001  D3".............
00000030: 0000 ffff 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 0003 0002 0000 00a0 0000 2580 0090 0000  ..........%.....
00000070: 0000 2580 0000 0000 0000 0000 0000 0000  ..%.............
```

Region 0 - 0x0090000 0x00002580

Load RAM Firmware



Load RAM Firmware



RAM Firmware Header

- Firmware Header
  - Built Date

```
001f5120: 2323 2323 0000 0000 0000 0000 0000 0000  ####............
001f5130: 0000 0000 00a8 2002 00f0 0100 2000 0000  ...... ..... ...
001f5140: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001f5150: 0000 0000 0000 0000 0000 0000 00dc 3102  ..............1.
001f5160: 0024 0300 0000 0000 0000 0000 0000 0000  .$..............
…
001f52a0: 0020 0300 8000 0000 0000 0000 0000 0000  . ..............
001f52b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001f52c0: 0000 0000 0060 09f0 b066 0100 8000 0000  .....`...f......
001f52d0: 0000 0000 0000 0000 0000 0000 1400 0b02  ................
001f52e0: 0100 005f 5f5f 5f30 3030 3030 3032 3032  ..____000000202
001f52f0: 3330 3732 3831 3531 3435 3500 72d5 d356  30728151455.r..V
```

It’s built at 2023/07/28/ 15:14:55

RAM Firmware Header

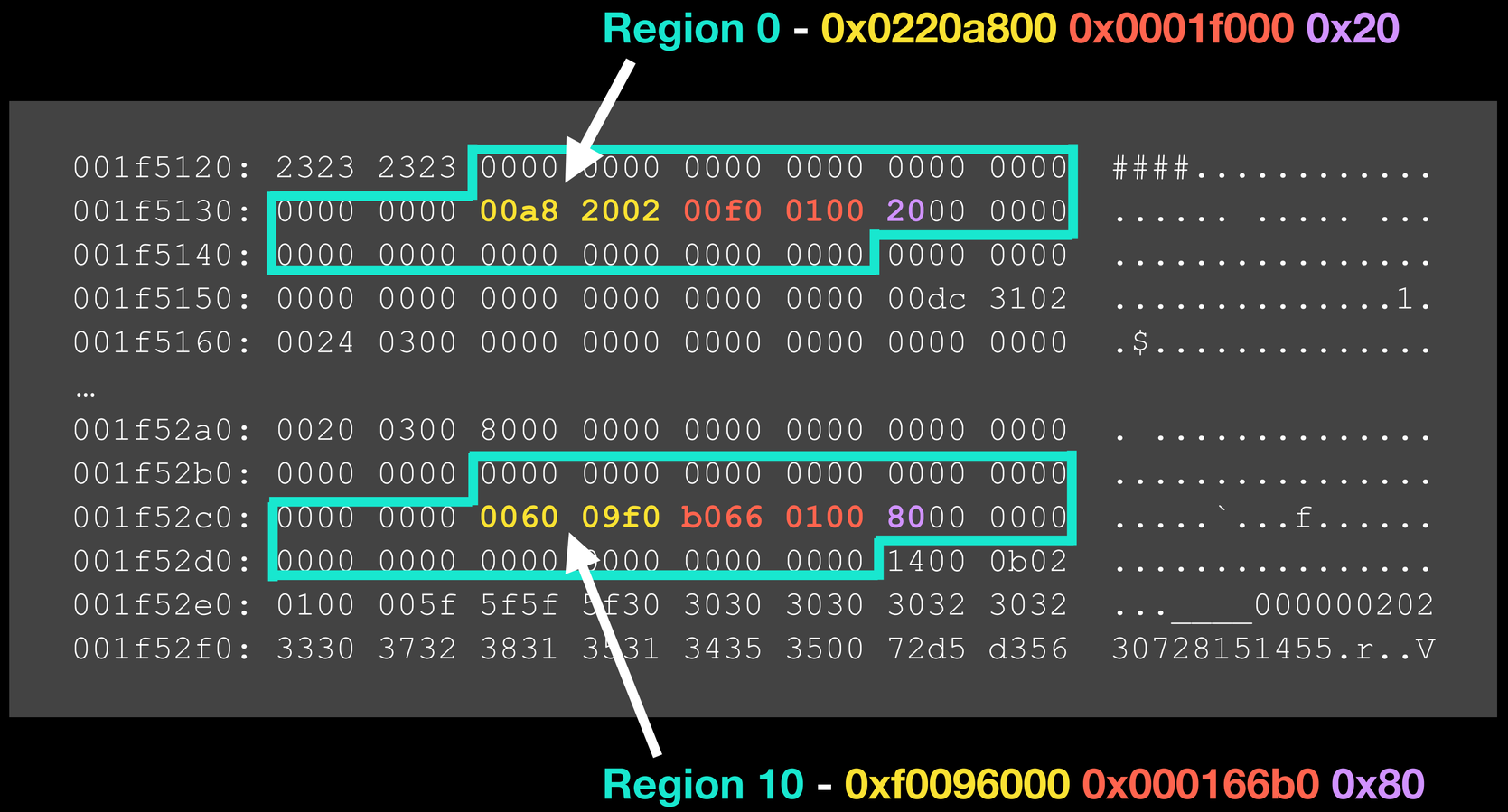
- Firmware Header
  - Built Date
  - Number of Regions

```
001f5120: 2323 2323 0000 0000 0000 0000 0000 0000  ####............
001f5130: 0000 0000 00a8 2002 00f0 0100 2000 0000  ...... ..... ...
001f5140: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001f5150: 0000 0000 0000 0000 0000 0000 00dc 3102  ..............1.
001f5160: 0024 0300 0000 0000 0000 0000 0000 0000  .$..............
…
001f52a0: 0020 0300 8000 0000 0000 0000 0000 0000  . ..............
001f52b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001f52c0: 0000 0000 0060 09f0 b066 0100 8000 0000  .....`...f......
001f52d0: 0000 0000 0000 0000 0000 0000 1400 0b02  ................
001f52e0: 0100 005f 5f5f 5f30 3030 3030 3033 3032  ..____000000202
001f52f0: 3330 3732 3831 3531 3435 3500 72d5 d356  30728151455.r..V
```

This firmware has 11 regions

RAM Region Header

- Region Header
  - Address
  - Length
  - Feature Set



Region0

0x20 is FW_FEATURE_OVERRIDE_RAM_ADDR,
so the entry point will be set to 0x0220a800

```
001f5120: 2323 2323 0000 0000 0000 0000 0000 0000  ####............
001f5130: 0000 0000 00a8 2002 00f0 0100 2000 0000  ...... ..... ...
001f5140: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001f5150: 0000 0000 0000 0000 0000 0000 00dc 3102  ..............1.
001f5160: 0024 0300 0000 0000 0000 0000 0000 0000  .$..............
…
001f52a0: 0020 0300 8000 0000 0000 0000 0000 0000  . ..............
```

Andes NDS32

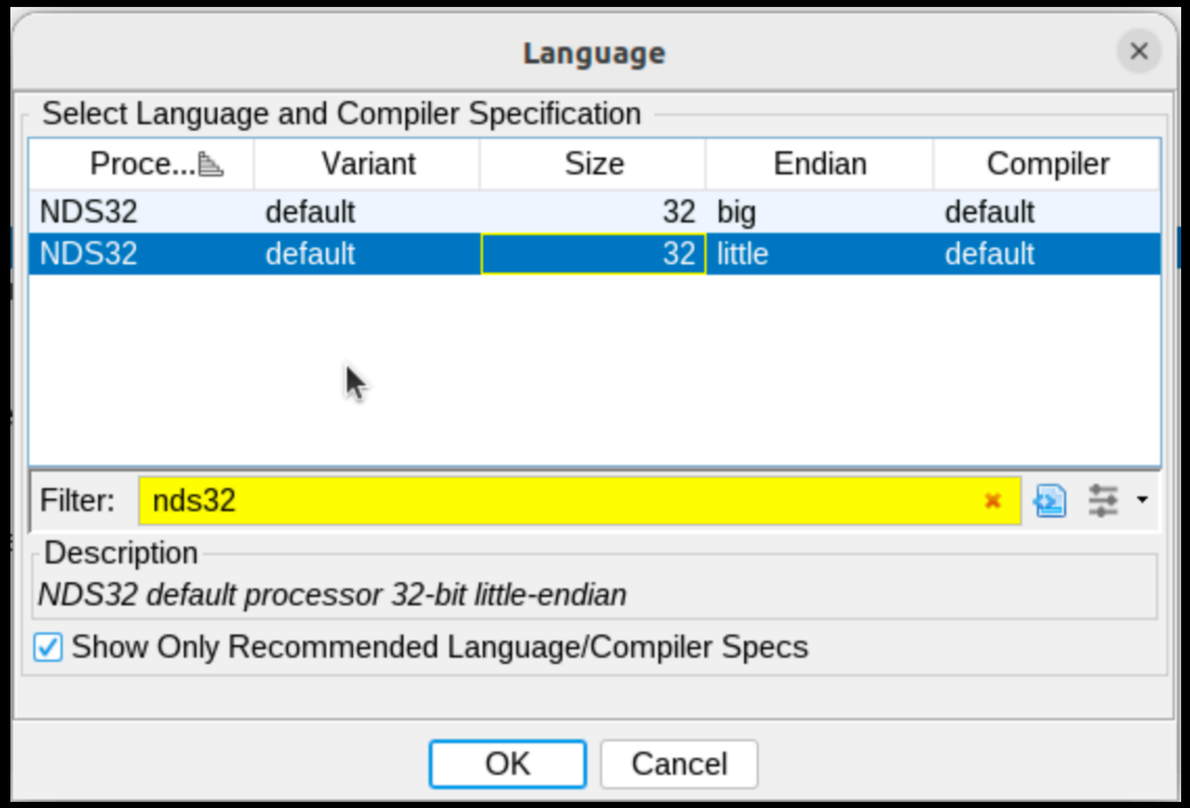
- The architecture of WOCPU is **NDS32**
  - **Does the WMCPU has the same architecture?**

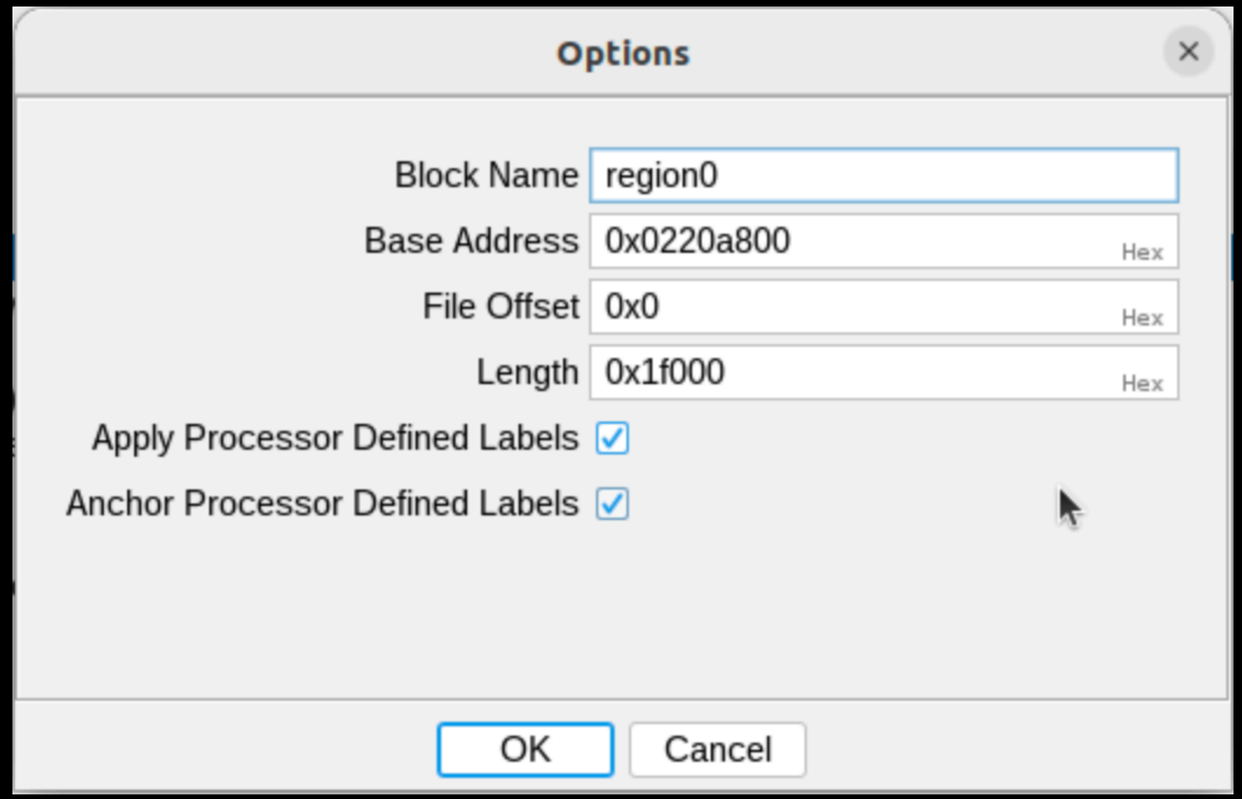
```
$ ls /lib/firmware
7981_WOCPU0_RAM_CODE_release.bin  7986_WOCPU0_RAM_CODE_release.bin  7986_WOCPU1_RAM_CODE_release.bin

$ strings 7981_WOCPU0_RAM_CODE_release.bin|grep '/.*/'
coe/mcu/driver/gpt/gpt.c
…
coe/kernel/FreeRTOSv10.0.0/portable/GCC/NDS32/port.c
```

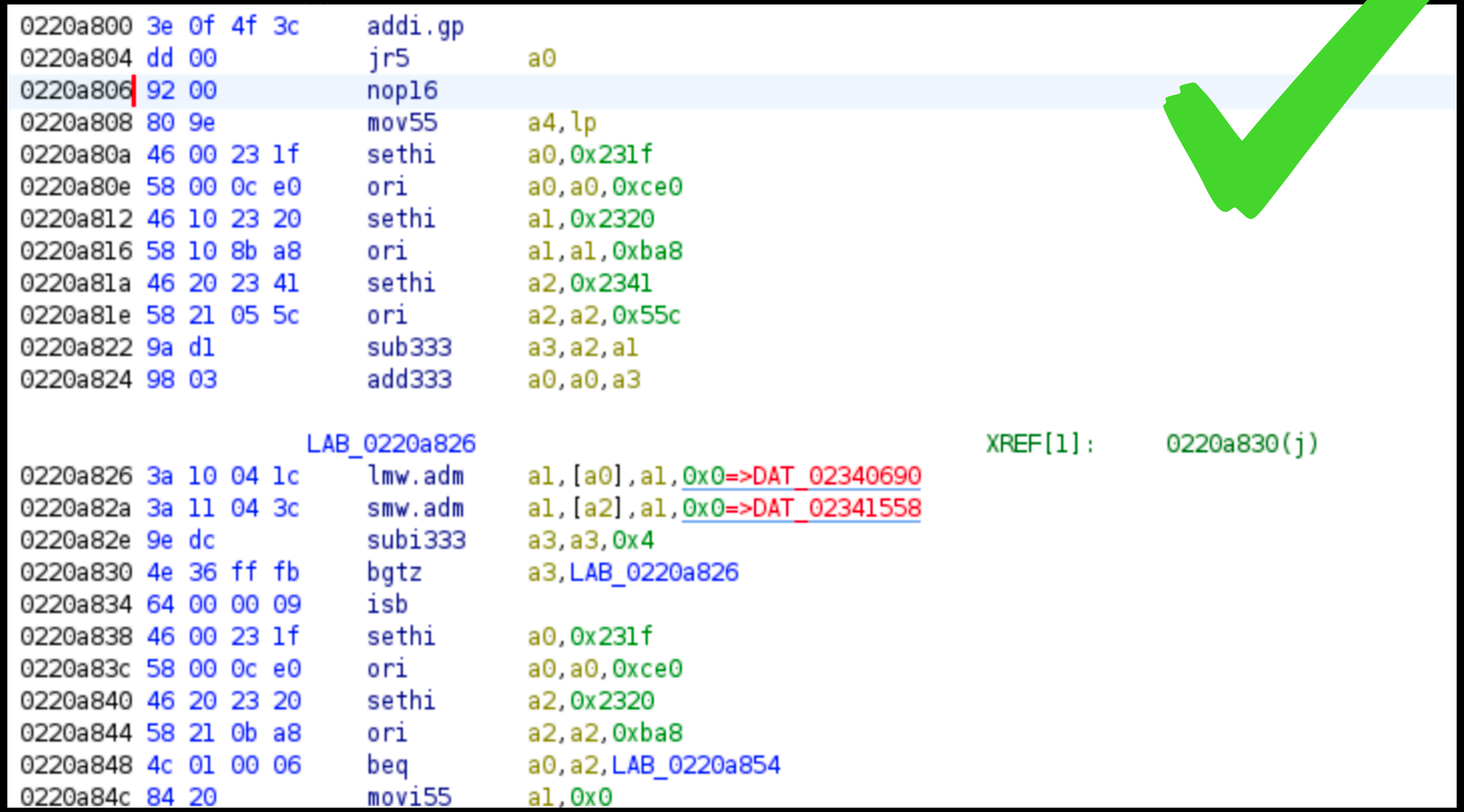
Andes NDS32

- We apply the NDS32 patch from the ghidra-staging to Ghidra
- And then Load the Region 0 to the Ghidra





Andes NDS32



```
0220a800 3e 0f 4f 3c     addi.gp
0220a804 dd 00           jr5         a0
0220a806 92 00           nop16
0220a808 80 9e           mov55       a4,lp
0220a80a 46 00 23 1f     sethi       a0,0x231f
0220a80e 58 00 0c e0     ori         a0,a0,0xce0
0220a812 46 10 23 20     sethi       a1,0x2320
0220a816 58 10 8b a8     ori         a1,a1,0xba8
0220a81a 46 20 23 41     sethi       a2,0x2341
0220a81e 58 21 05 5c     ori         a2,a2,0x55c
0220a822 9a d1           sub333      a3,a2,a1
0220a824 98 03           add333      a0,a0,a3

                     LAB_0220a826                                XREF[1]:     0220a830(j)
0220a826 3a 10 04 1c     lmw.adm     a1,[a0],a1,0x0=>DAT_02340690
0220a82a 3a 11 04 3c     smw.adm     a1,[a2],a1,0x0=>DAT_02341558
0220a82e 9e dc           subi333     a3,a3,0x4
0220a830 4e 36 ff fb     bgtz        a3,LAB_0220a826
0220a834 64 00 00 09     isb
0220a838 46 00 23 1f     sethi       a0,0x231f
0220a83c 58 00 0c e0     ori         a0,a0,0xce0
0220a840 46 20 23 20     sethi       a2,0x2320
0220a844 58 21 0b a8     ori         a2,a2,0xba8
0220a848 4c 01 00 06     beq         a0,a2,LAB_0220a854
0220a84c 84 20           movi55      a1,0x0
```

Unknown GP Register

```
57        else {
58          iVar1 = FUN_e004c970();
59          if (iVar1 != 1) {
60            FUN_e003b50c(5,0x45);
61            iVar1 = FUN_e004c9f0();
62            if (iVar1 == 1) {
63              FUN_e00405d6();
64            }
65            goto LAB_0220be8c;
66          }
67          FUN_e0042482(0x15,4);
68          FUN_e004232c(s_Skip_send_notify_due_to_hif_susp_e00a6410);
69  LAB_0220be8e:
70          FUN_e003b50c(5,0x4a);
71          iVar1 = FUN_e003da16(&DAT_e00f9294,unaff_gp + 0x1b58c,0x803);
72          if (iVar1 != 0) {
73            FUN_e003dbfc(&DAT_e00f9294,unaff_gp + 0x1b58c,0x804);
74          }
75          FUN_e003b50c(5,0x4b);
76          FUN_e003dfd0(&DAT_e00f9294,5,unaff_gp + 0x1b58c,0x807);
77        }
78        uVar3 = 0x4c;
79        goto LAB_0220bf06;
80      }
81    }
82    FUN_e003b50c(5,0x26);
83    FUN_e00597c0(0);
```

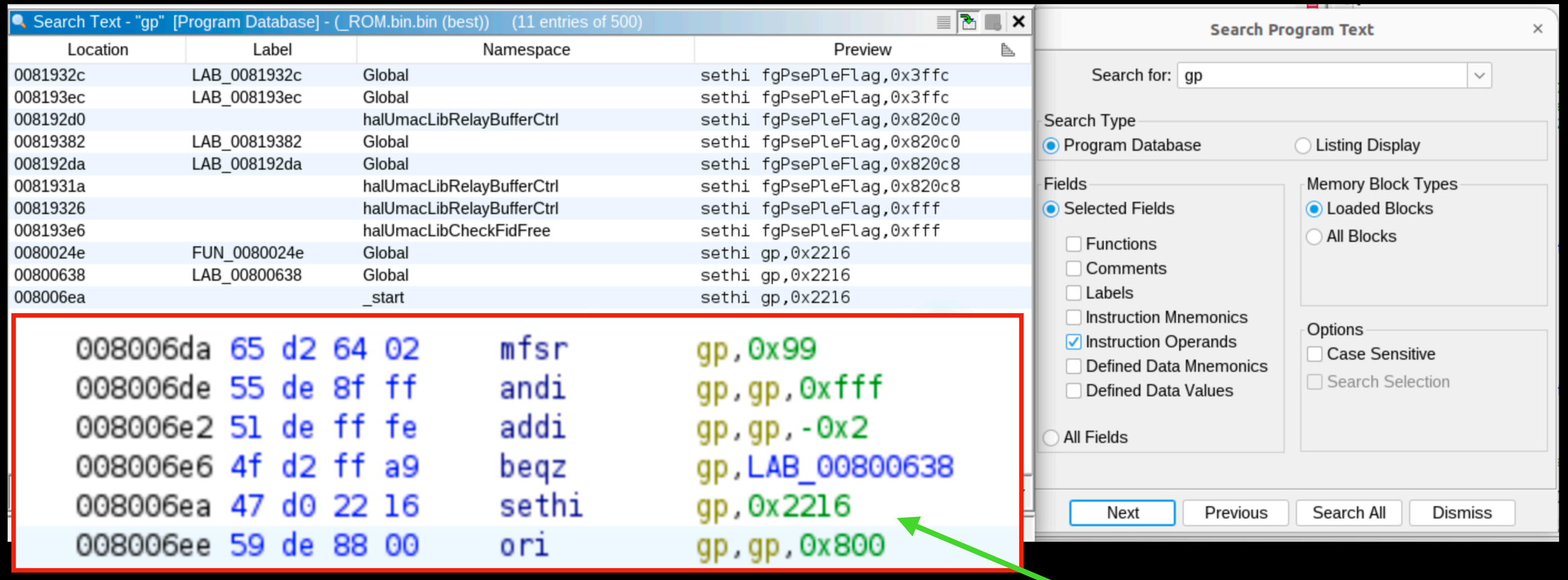
Unknown value of gp register

Debug Commands

- Command for dumping memory from WMCPU
  - iwpriv wl0 show core_dump

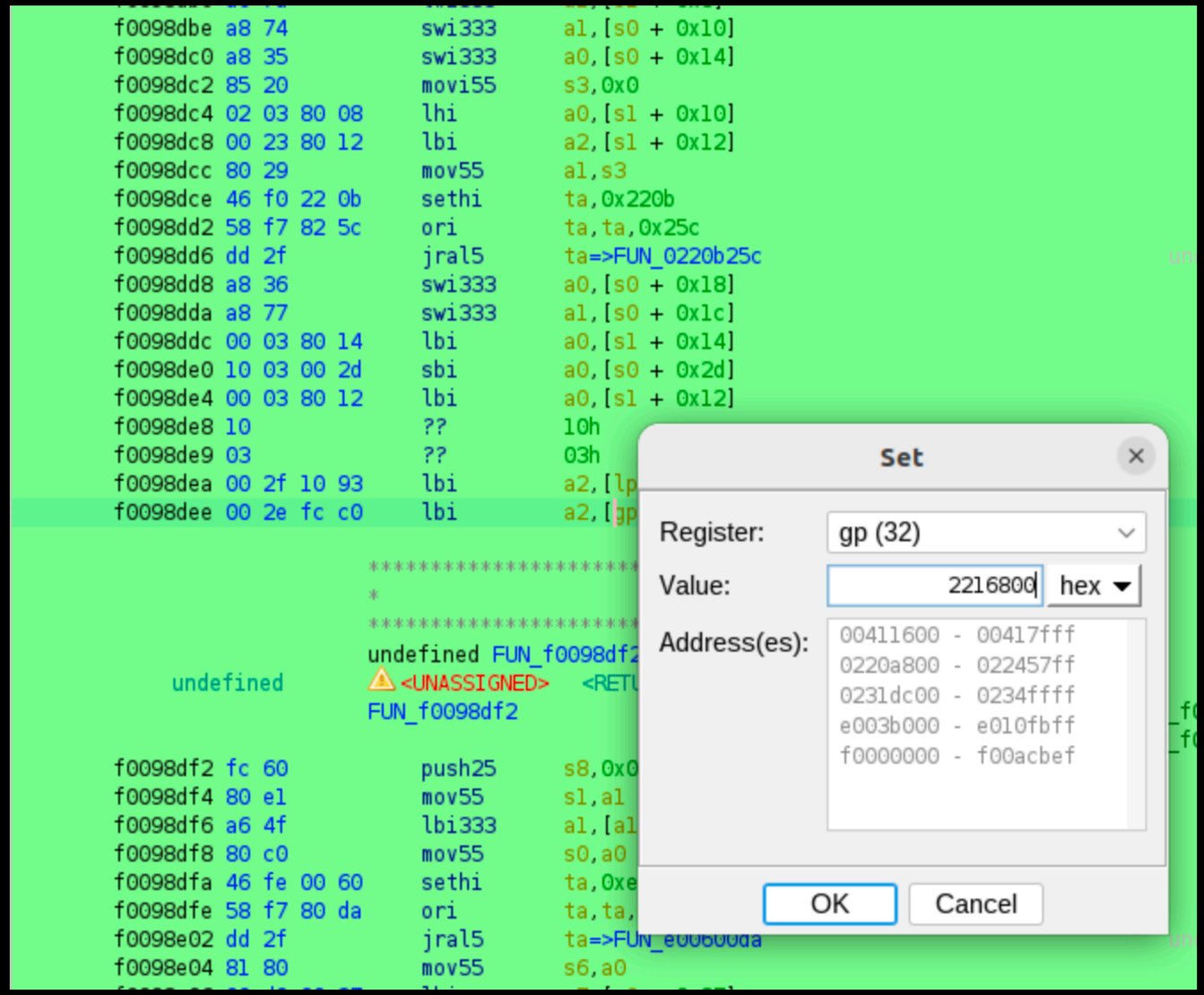
```
/etc/ _ROM.bin.bin      addr:0x800000
/etc/ULM1.bin           addr:0x900000
/etc/ULM2.bin           addr:0x2200000
/etc/ULM3.bin           addr:0x2300000
/etc/SRAM.bin           addr:0x400000
/etc/CRAM.bin           addr:0xe0000000
```

Locate GP Register Setup



The value of gp register is 0x2216800

Modify GP Register



Readable Decompiled Code

```
110   else {
111     iVar1 = FUN_e004c970();
112     if (iVar1 != 1) {
113       FUN_e003b50c(5,0x45);
114       iVar1 = FUN_e004c9f0();
115       if (iVar1 == 1) {
116         FUN_e00405d6();
117       }
118       goto LAB_0220be8c;
119     }
120     FUN_e0042482(0x15,4);
121     FUN_e004232c(s_Skip_send_notify_due_to_hif_susp_e00a6410);
122 LAB_0220be8e:
123     FUN_e003b50c(5,0x4a);
124     iVar1 = FUN_e003da16(&DAT_e00f9294,s_pwrCtrlInactivateAllHw_02231d8c,0x803);
125     if (iVar1 != 0) {
126       FUN_e003dbfc(&DAT_e00f9294,s_pwrCtrlInactivateAllHw_02231d8c,0x804);
127     }
128     FUN_e003b50c(5,0x4b);
129     FUN_e003dfd0(&DAT_e00f9294,5,s_pwrCtrlInactivateAllHw_02231d8c,0x807);
130   }
131   uVar2 = 0x4c;
132 LAB_0220bf06:
133   FUN_e003b50c(5,uVar2);
```

Readable strings

Found Vulnerability!

Found Vulnerability!!

Found Vulnerability!!!

Found Vulnerability???

Found Vulnerability????

How?

Version of Firmware

- I am sure that I update the firmware of my device to date
  - The latest version is released at 2023/12/22

Version of Firmware

- I am sure that I update the firmware of my device to date
  - The latest version is released at 2023/12/22
- The built time of WMCPU is 2023/07/28 15:14:55 in my device
  - However, the vulnerability is patched at 2023/10/24 20:11:49

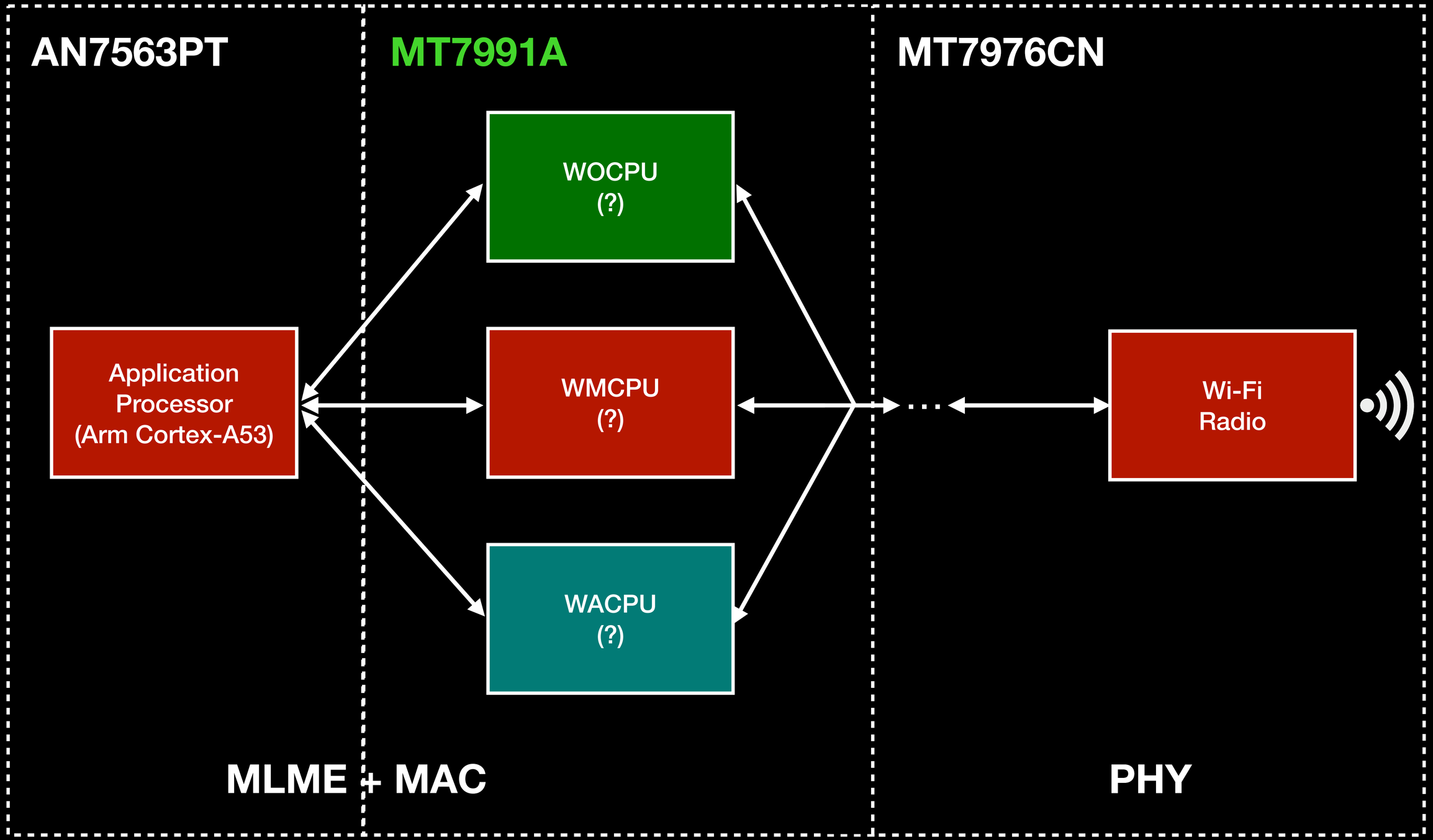
Version of Firmware

- I am sure that I update the firmware of my device to date
  - The latest versio
- The built time of ny device
  - However, the vu 1:49

Fun Fact

- Device vendors update firmware but sometimes skip chipset vendor patches
- I encountered the same problem throughout my research

How about Wi-Fi 7 ?

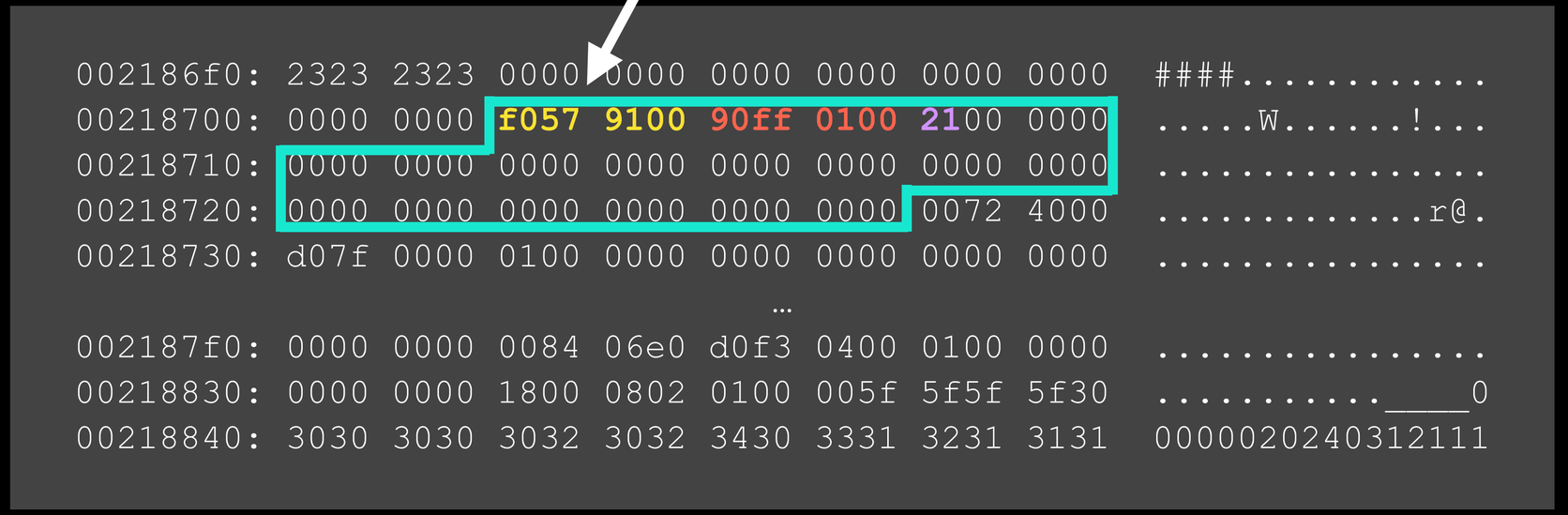

AN7563PT
MT7991A
MT7976CN
WOCPU
(?)
Application
Processor
(Arm Cortex-A53)
WMCPU
(?)
Wi-Fi
Radio
WACPU
(?)
MLME + MAC
PHY

RAM Region Header

- Region Header
  - Address
  - Length
  - Feature Set

0x01 is FW_FEATURE_SET_ENCRYPT !

Region 0 - 0x009157f0 0x0001f000 0x21



Encrypted MCU Firmware

- All Regions are encrypted
  - No decryption algorithm is present in the both firmware and kernel driver
  - The decryption appears to be handled by hardware
  - Unable to retrieve the decryption key

Plaintext MCU Firmware

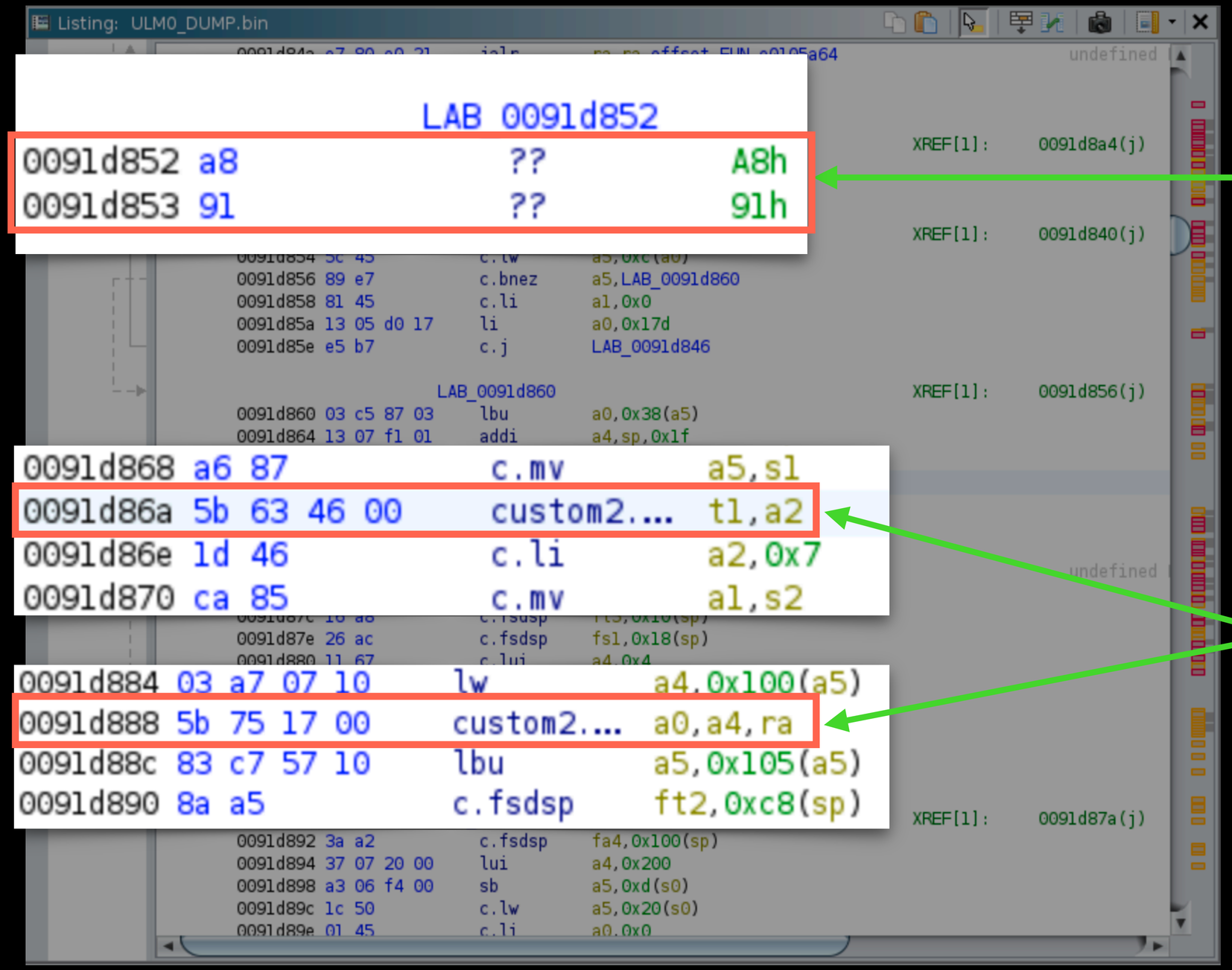
- Commands for dumping memory from WMCPU
  - mwctl dev wl0 set trig_core_dump=1
  - mwctl dev wl0 show core_dump

```
/etc/ULM0_DUMP.bin          addr:0x00800000
/etc/ULM1_DUMP.bin          addr:0x00900000
/etc/ULM2_DUMP.bin          addr:0x02200000
/etc/fw_dump_wmcpu.cmm      addr:0x00400000
/etc/CRAM_DUMP.bin          addr:0xE0000000
/etc/CONN_INFRA_DUMP.bin addr:0x7C050000
```

RISC-V

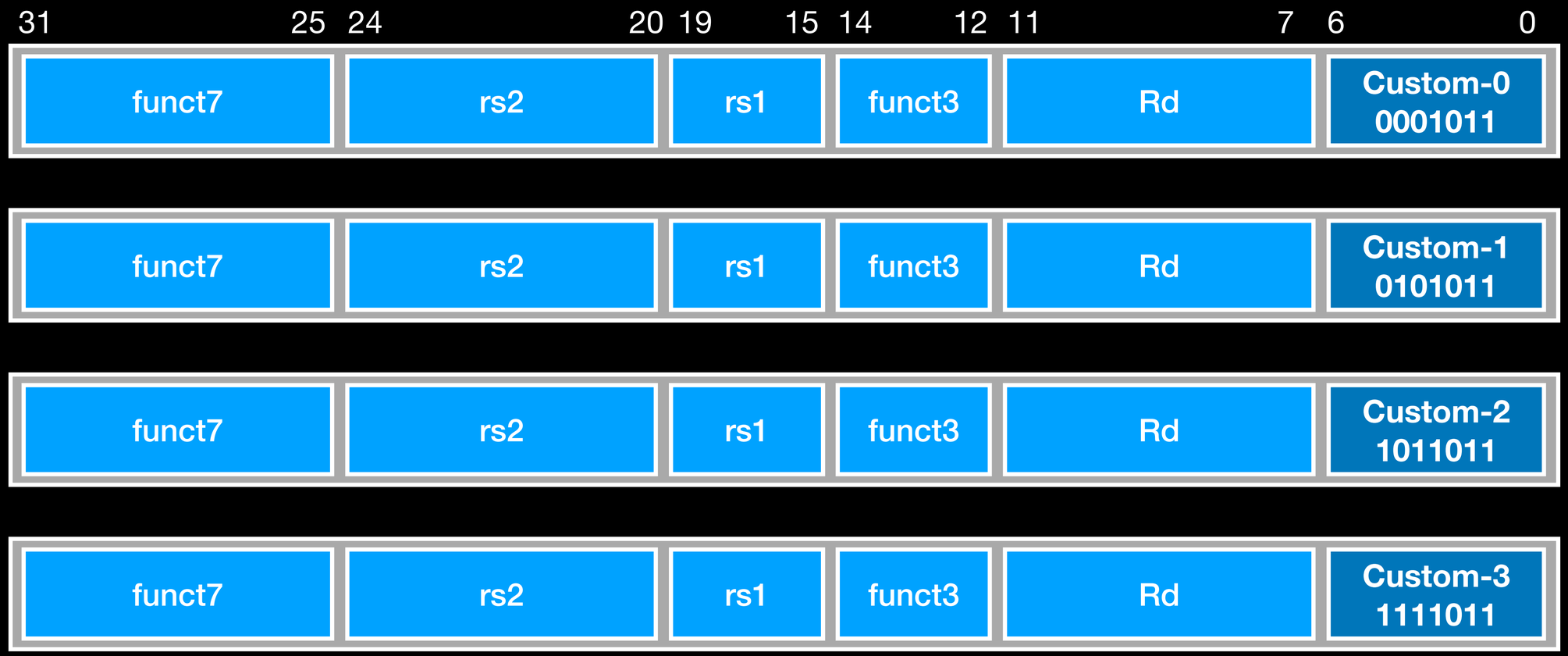
- The architecture of WMCPU is **RISC-V**

```
$ strings *|grep '/.*/'
…
mcu/system/rom/common/cos_task_rom.c
mcu/system/rom/common/intrCtrl_rom.c
mcu/system/rom/common/sys_api_rom.c
…
mcu/os/FreeRTOS/FreeRTOSv10.4.3_LTS_Patch_3/portable/Common/RISC-V/port.c
```


Listing: ULM0_DUMP.bin
LAB 0091d852
0091d852 a8 ?? A8h
0091d853 91 ?? 91h
XREF[1]: 0091d8a4(j)
XREF[1]: 0091d840(j)
0091d856 89 e7 c.bnez a5,LAB_0091d860
0091d858 81 45 c.li a1,0x0
0091d85a 13 05 d0 17 li a0,0x17d
0091d85e e5 b7 c.j LAB_0091d846
LAB_0091d860
XREF[1]: 0091d856(j)
0091d860 03 c5 87 03 lbu a0,0x38(a5)
0091d864 13 07 f1 01 addi a4,sp,0x1f
0091d868 a6 87 c.mv a5,s1
0091d86a 5b 63 46 00 custom2.... t1,a2
0091d86e 1d 46 c.li a2,0x7
0091d870 ca 85 c.mv a1,s2
undefined
0091d87e 26 ac c.fsdsp fs1,0x18(sp)
0091d884 03 a7 07 10 lw a4,0x100(a5)
0091d888 5b 75 17 00 custom2.... a0,a4,ra
0091d88c 83 c7 57 10 lbu a5,0x105(a5)
0091d890 8a a5 c.fsdsp ft2,0xc8(sp)
XREF[1]: 0091d87a(j)
0091d892 3a a2 c.fsdsp fa4,0x100(sp)
0091d894 37 07 20 00 lui a4,0x200
0091d898 a3 06 f4 00 sb a5,0xd(s0)
0091d89c 1c 50 c.lw a5,0x20(s0)
Unknown RISC-V Instruction
Unknown Custom RISC-V Instruction

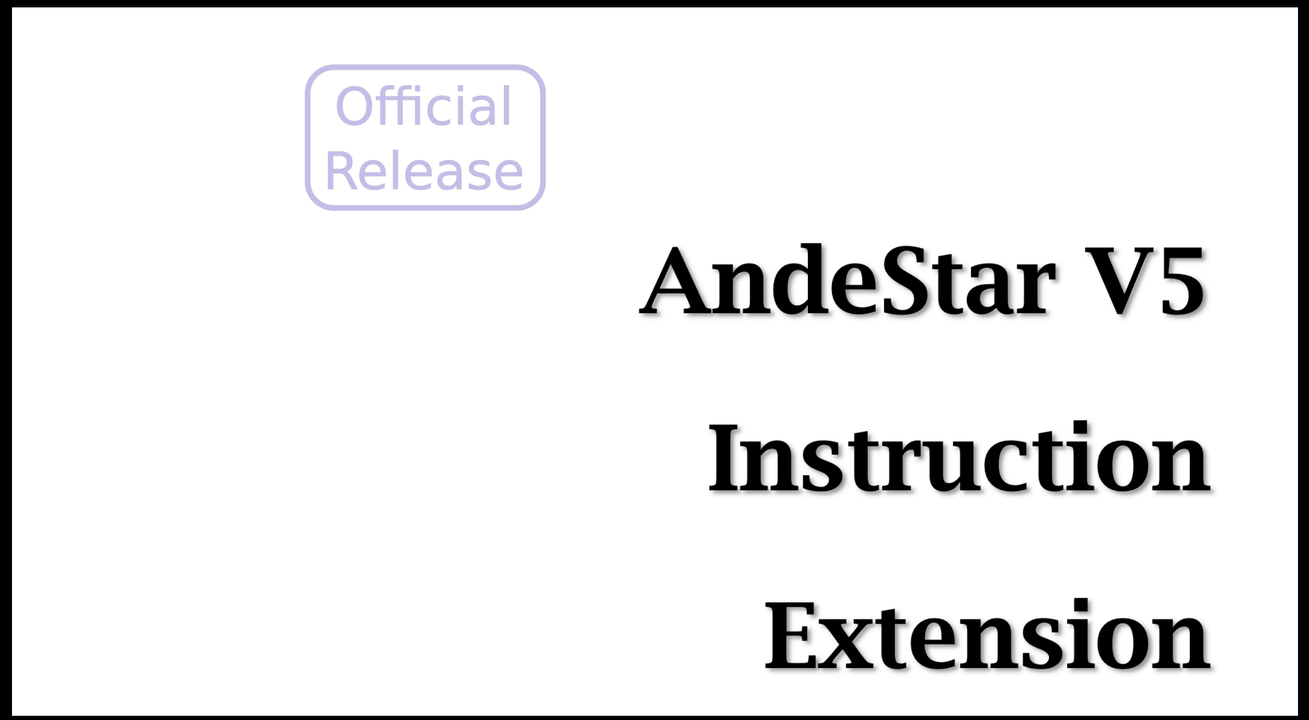
Custom RISC-V instructions

R-type

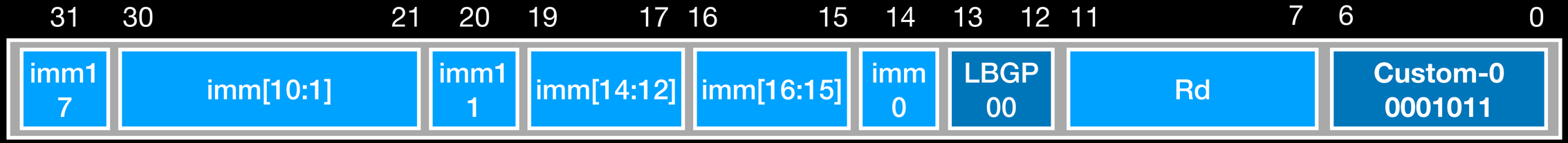


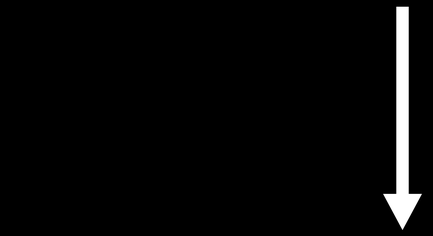
Andes AndeStar V5

- As we know, the Wi-Fi MCU of Wi-Fi 6 is from Andes
- Andes is also well known for their RISC-V CPUs



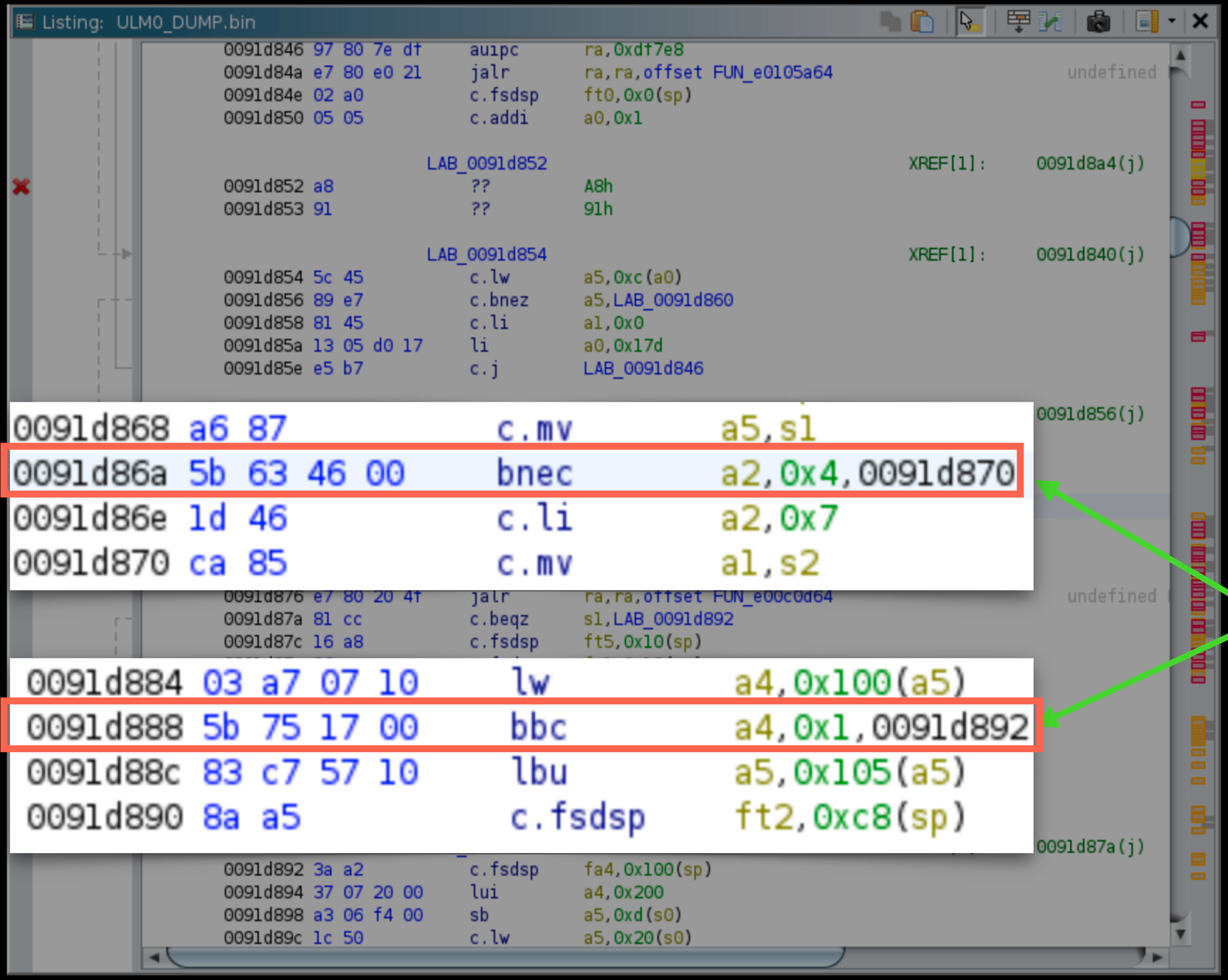
SLEIGH language

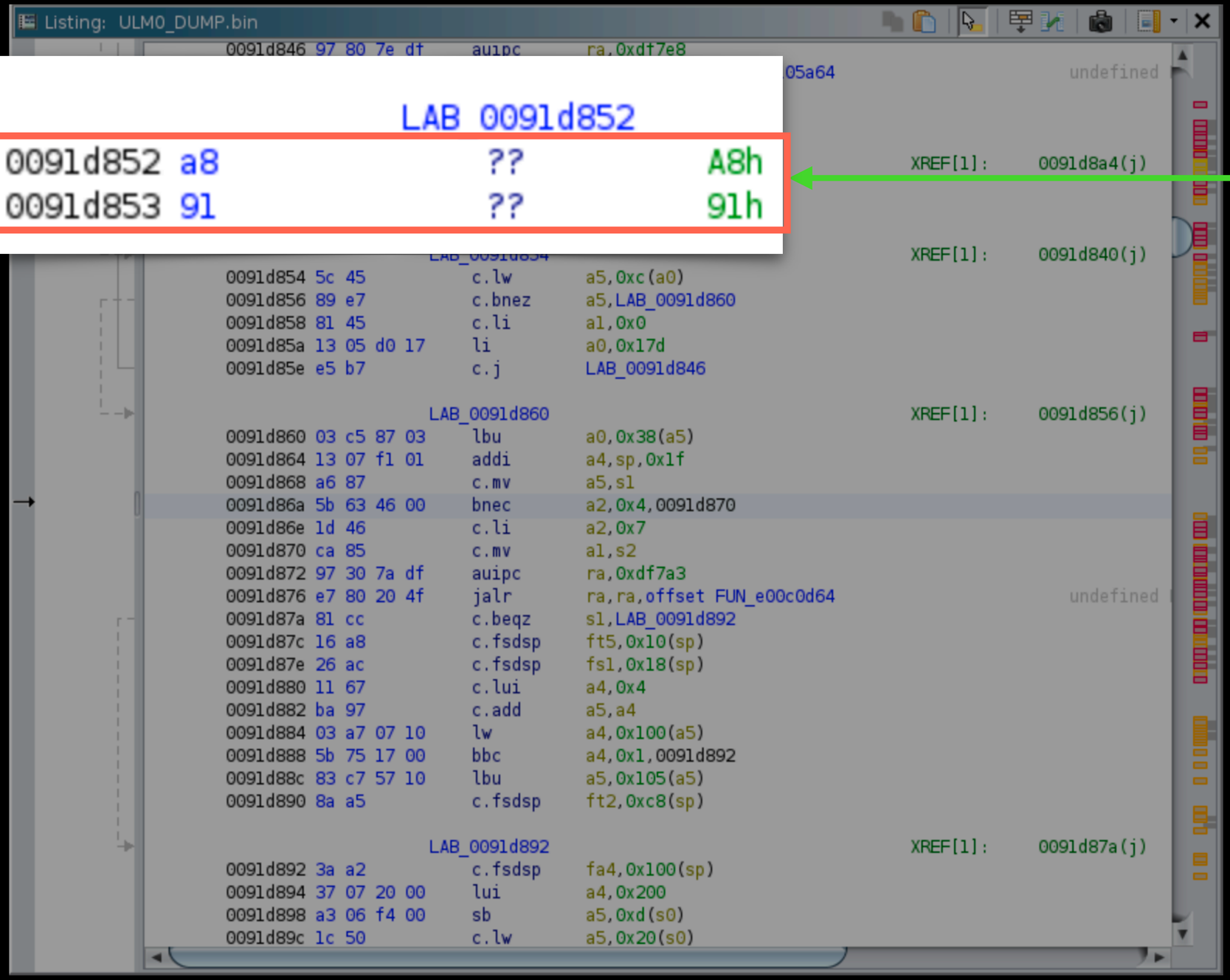




SLEIGH Language

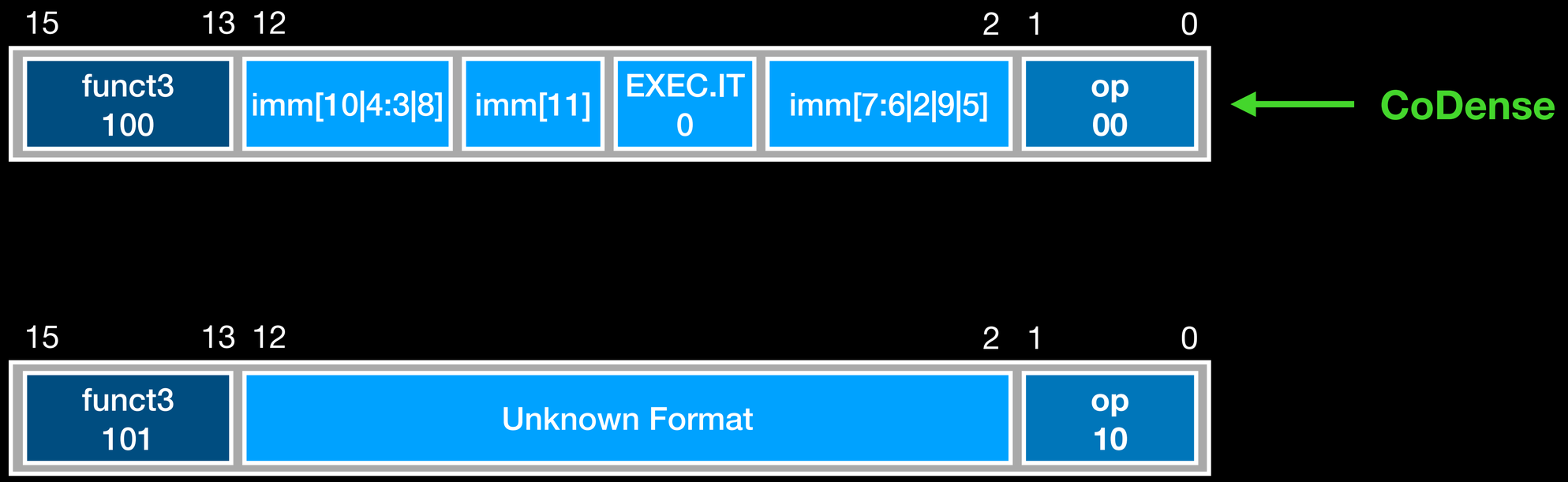
```
:ldgp rd,[+GpAddressLoadD] is op0001=0x3 & op0204=0x2 & op0506=0x1 & op1214=0x3 & GpAddressLoadD & rd
{
    local tmp = *[ram] GpAddressLoadD;
    rd = tmp;
}
```


Listing: ULM0_DUMP.bin
0091d846 97 80 7e df auipc ra,0xdf7e8
0091d84a e7 80 e0 21 jalr ra,ra,offset FUN_e0105a64
0091d84e 02 a0 c.fsdsp ft0,0x0(sp)
0091d850 05 05 c.addi a0,0x1
LAB_0091d852
0091d852 a8 ?? A8h
0091d853 91 ?? 91h
LAB_0091d854
0091d854 5c 45 c.lw a5,0xc(a0)
0091d856 89 e7 c.bnez a5,LAB_0091d860
0091d858 81 45 c.li a1,0x0
0091d85a 13 05 d0 17 li a0,0x17d
0091d85e e5 b7 c.j LAB_0091d846
0091d868 a6 87 c.mv a5,s1
0091d86a 5b 63 46 00 bnec a2,0x4,0091d870
0091d86e 1d 46 c.li a2,0x7
0091d870 ca 85 c.mv a1,s2
0091d87a 81 cc c.beqz s1,LAB_0091d892
0091d87c 16 a8 c.fsdsp ft5,0x10(sp)
0091d884 03 a7 07 10 lw a4,0x100(a5)
0091d888 5b 75 17 00 bbc a4,0x1,0091d892
0091d88c 83 c7 57 10 lbu a5,0x105(a5)
0091d890 8a a5 c.fsdsp ft2,0xc8(sp)
0091d892 3a a2 c.fsdsp fa4,0x100(sp)
0091d894 37 07 20 00 lui a4,0x200
0091d898 a3 06 f4 00 sb a5,0xd(s0)
0091d89c 1c 50 c.lw a5,0x20(s0)
Andes
Custom
RISC-V
Instructions

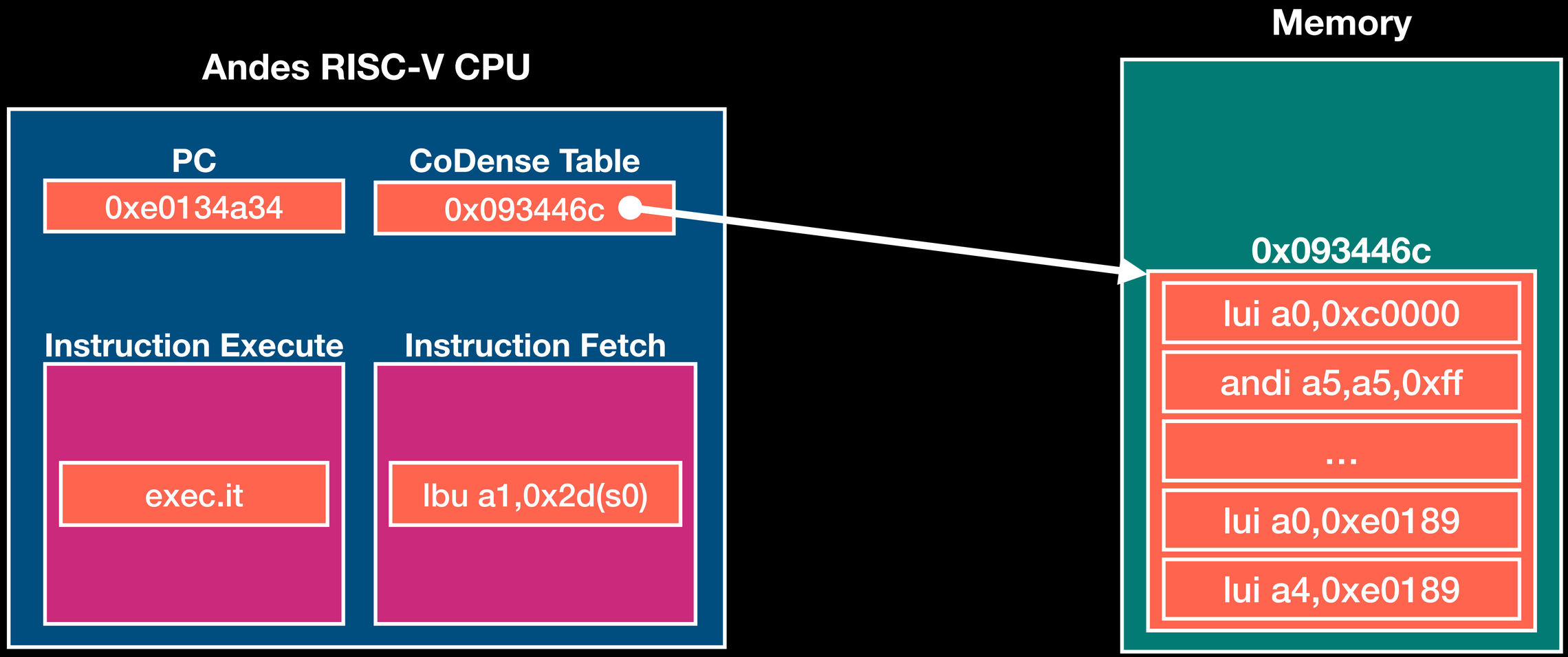

Listing: ULM0_DUMP.bin
LAB 0091d852
0091d852 a8 ?? A8h
0091d853 91 ?? 91h
Unknown RISC-V Instruction
0091d846 97 80 7e df auipc ra,0xdf7e8
05a64 undefined
XREF[1]: 0091d8a4(j)
XREF[1]: 0091d840(j)
0091d854 5c 45 c.lw a5,0xc(a0)
0091d856 89 e7 c.bnez a5,LAB_0091d860
0091d858 81 45 c.li a1,0x0
0091d85a 13 05 d0 17 li a0,0x17d
0091d85e e5 b7 c.j LAB_0091d846
LAB_0091d860
XREF[1]: 0091d856(j)
0091d860 03 c5 87 03 lbu a0,0x38(a5)
0091d864 13 07 f1 01 addi a4,sp,0x1f
0091d868 a6 87 c.mv a5,s1
0091d86a 5b 63 46 00 bnec a2,0x4,0091d870
0091d86e 1d 46 c.li a2,0x7
0091d870 ca 85 c.mv a1,s2
0091d872 97 30 7a df auipc ra,0xdf7a3
0091d876 e7 80 20 4f jalr ra,ra,offset FUN_e00c0d64
undefined
0091d87a 81 cc c.beqz s1,LAB_0091d892
0091d87c 16 a8 c.fsdsp ft5,0x10(sp)
0091d87e 26 ac c.fsdsp fs1,0x18(sp)
0091d880 11 67 c.lui a4,0x4
0091d882 ba 97 c.add a5,a4
0091d884 03 a7 07 10 lw a4,0x100(a5)
0091d888 5b 75 17 00 bbc a4,0x1,0091d892
0091d88c 83 c7 57 10 lbu a5,0x105(a5)
0091d890 8a a5 c.fsdsp ft2,0xc8(sp)
LAB_0091d892
XREF[1]: 0091d87a(j)
0091d892 3a a2 c.fsdsp fa4,0x100(sp)
0091d894 37 07 20 00 lui a4,0x200
0091d898 a3 06 f4 00 sb a5,0xd(s0)
0091d89c 1c 50 c.lw a5,0x20(s0)

Unknown Compressed Instructions

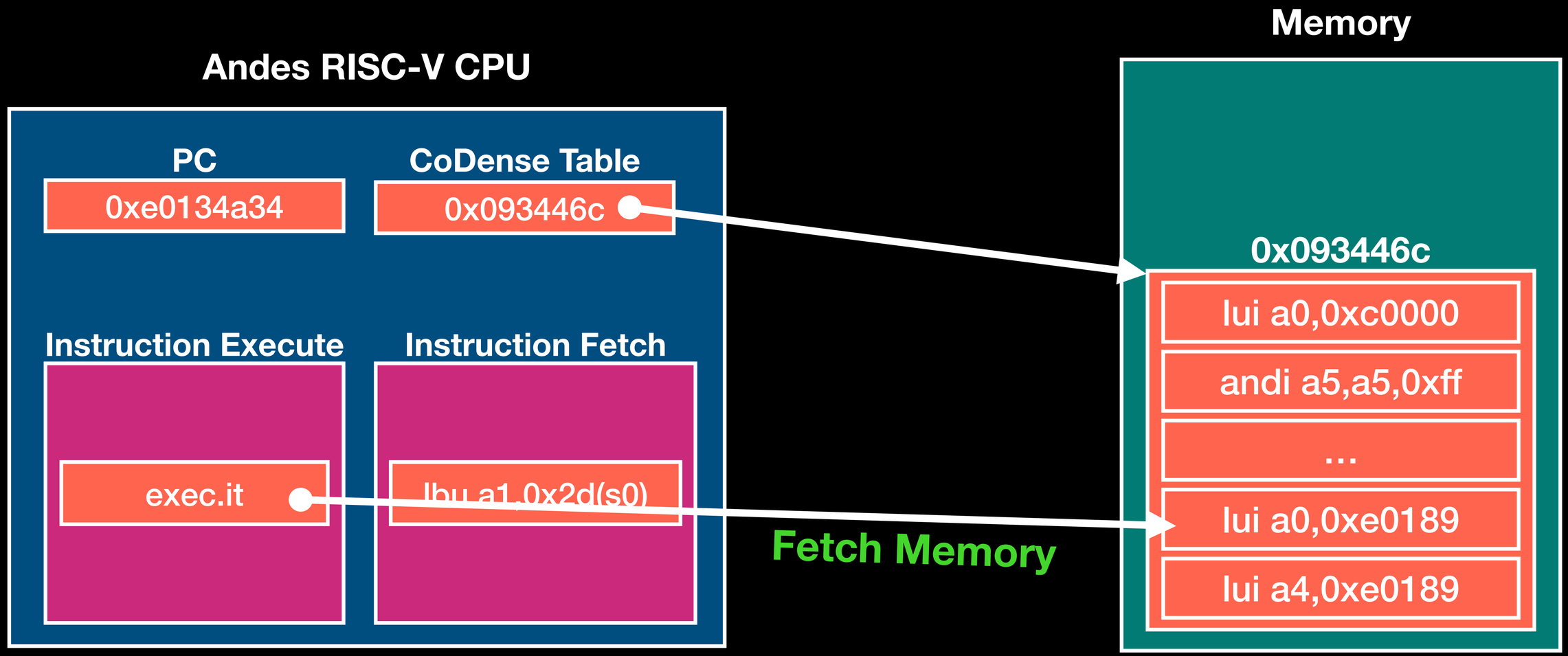
CoDense Extension



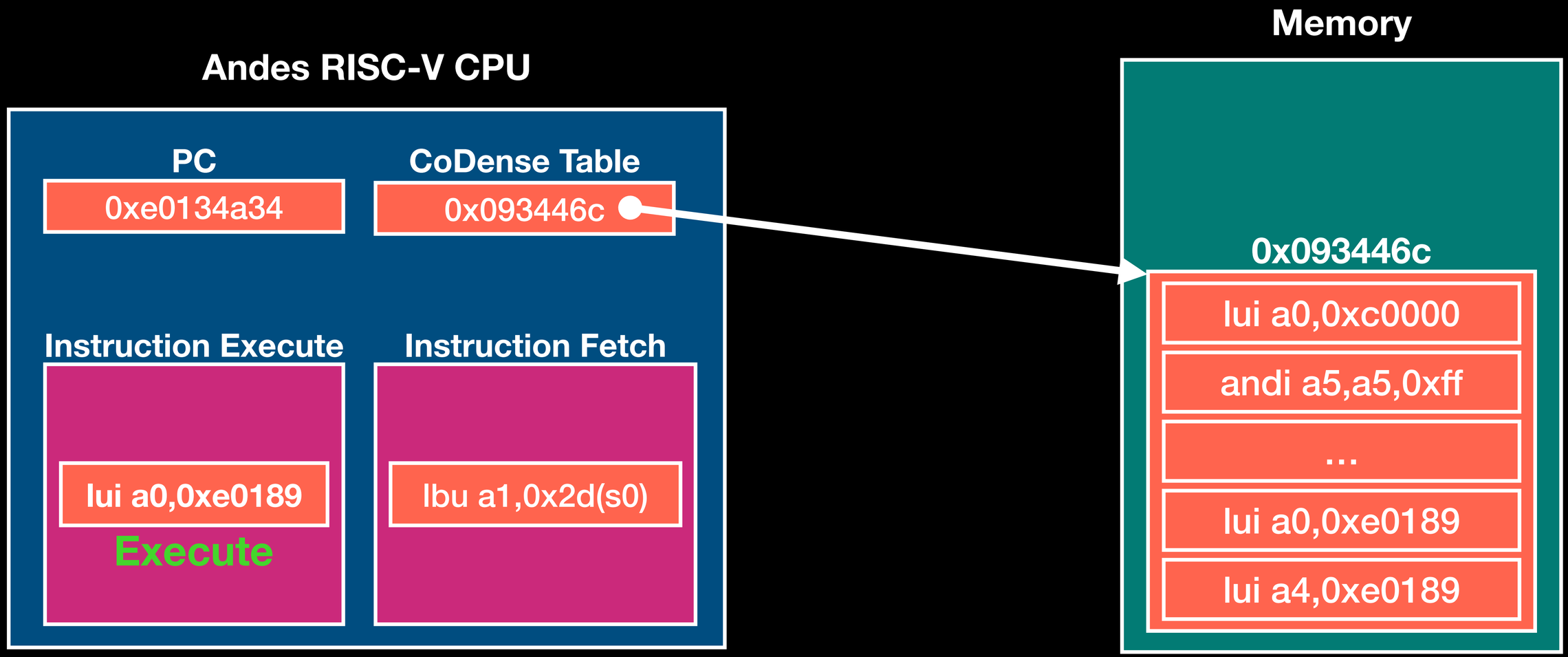
CoDense Extension



CoDense Extension



CoDense Extension



Unknown Compressed Instruction

What about this one?

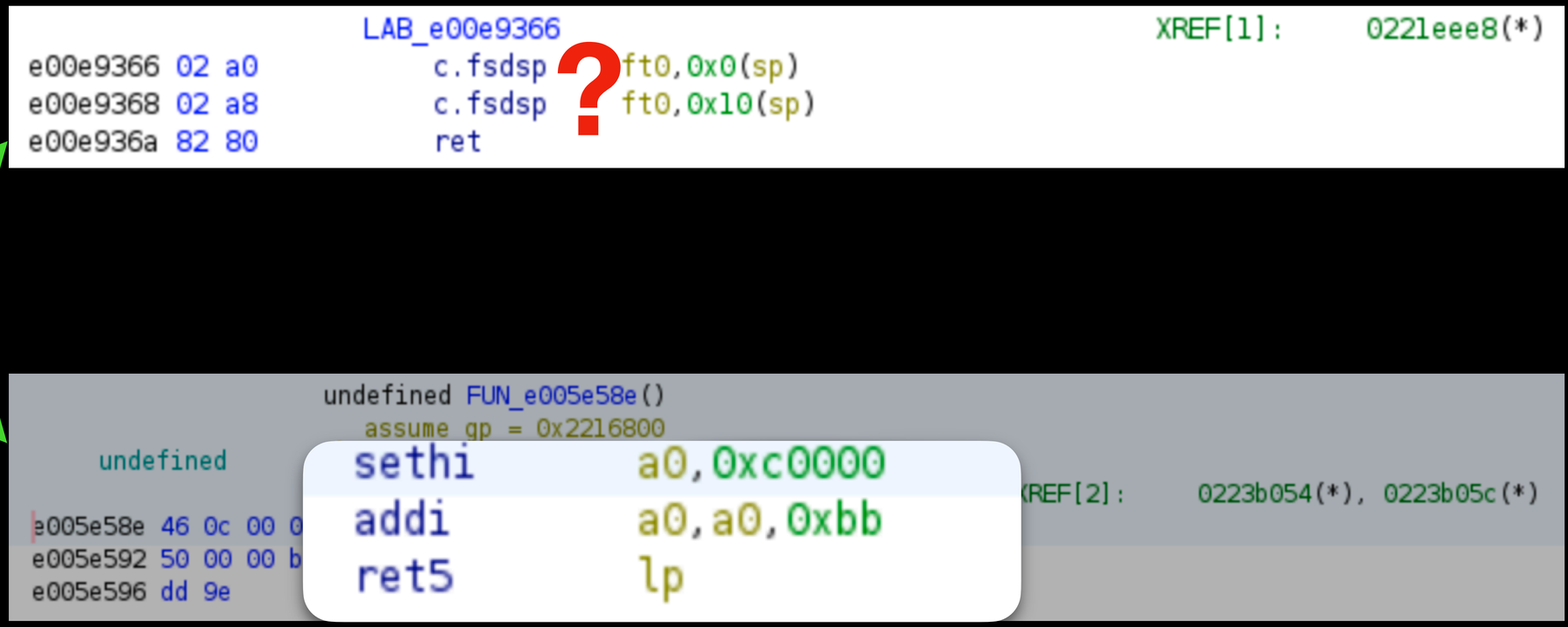
Unknown Compressed Instruction

Unknown Compressed Instruction

Wi-Fi 7

Should have same functionality

Wi-Fi 6

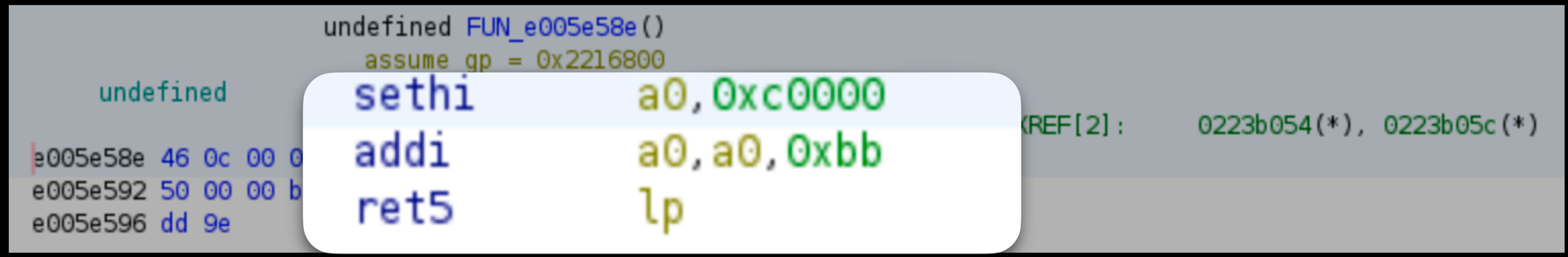


Compare the Same Function

Wi-Fi 7

I guess it should look like above

Wi-Fi 6



Compare the Same Function

Wi-Fi 7

```
e00e9366 02 a0
e00e9368 02 a8
e00e936a 82 80
```

```
exec.it.new  0
exec.it.new 16
```

=

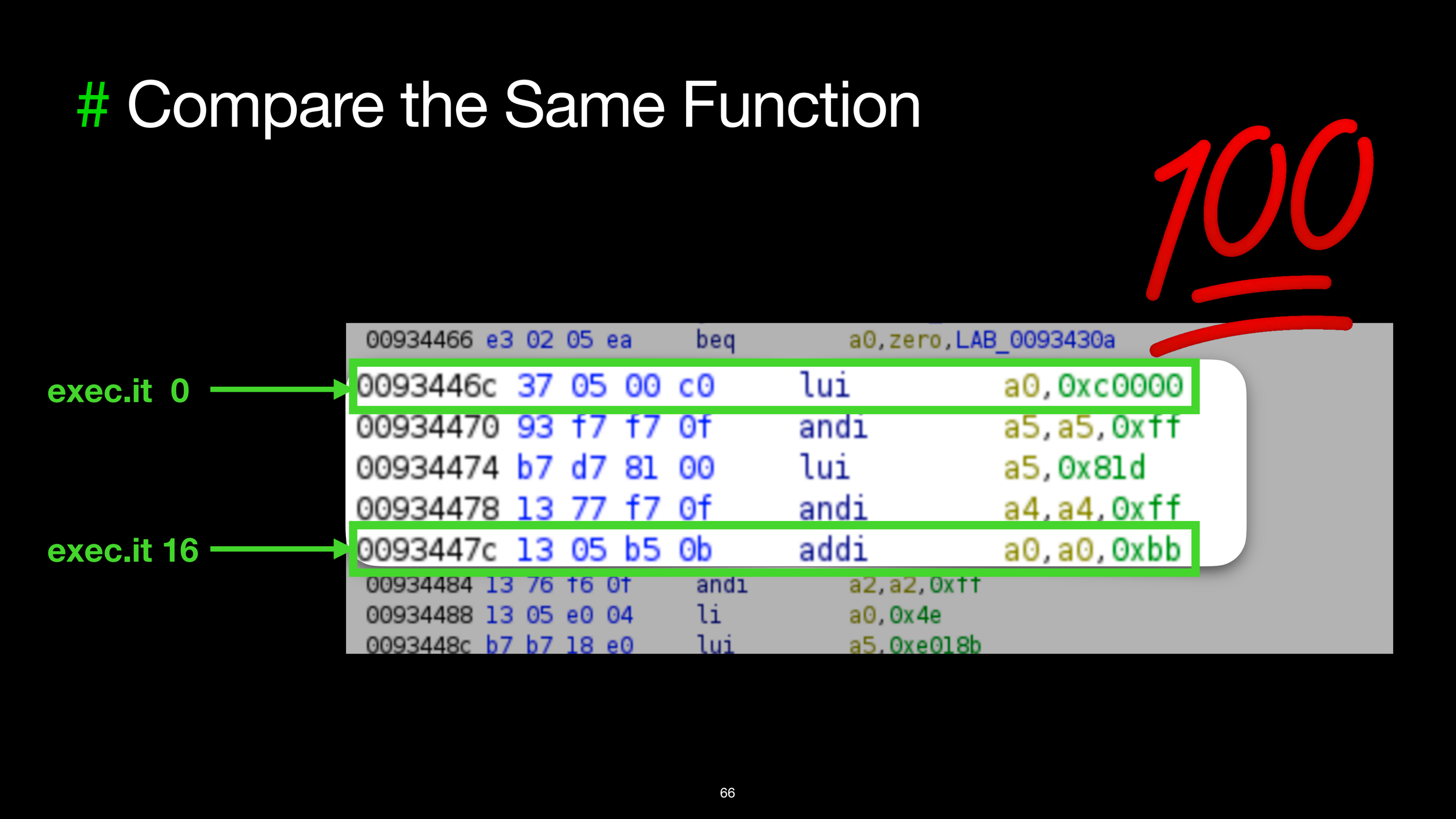
```
lui a0,0xc0000
addi a0,a0,0xbb
```

The actual Instructions that it should execute

Wi-Fi 6

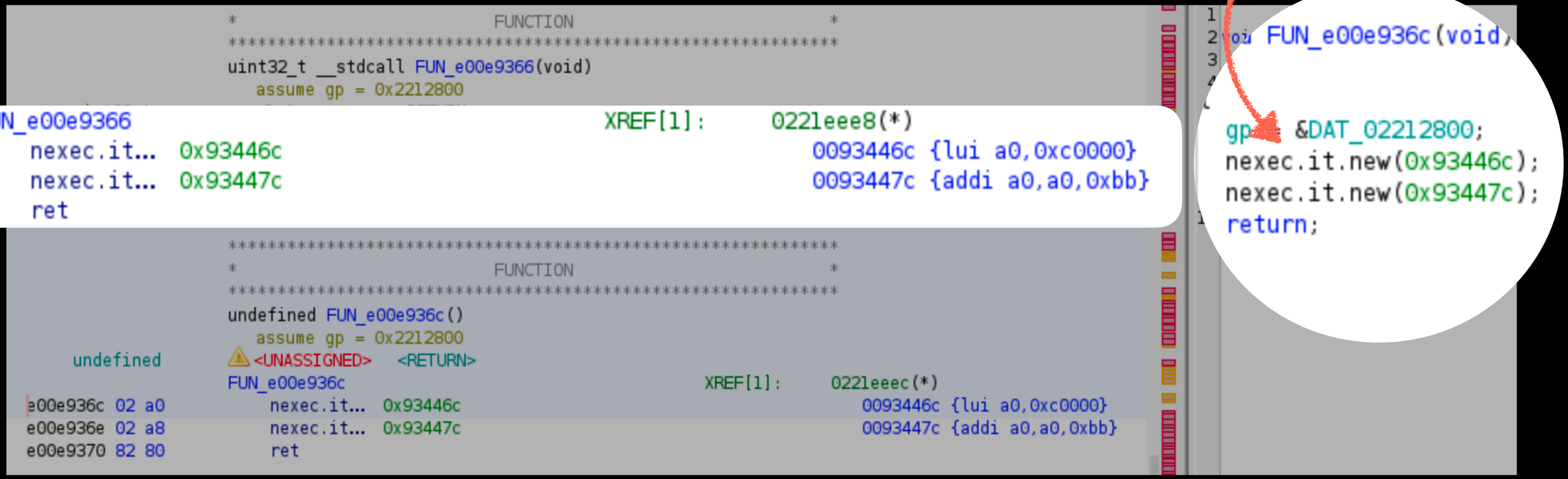
```
                    undefined FUN_e005e58e()
                      assume gp = 0x2216800
     undefined
                    sethi       a0,0xc0000
                    addi        a0,a0,0xbb
                    ret5        lp
e005e58e 46 0c 00 0                                  XREF[2]:     0223b054(*), 0223b05c(*)
e005e592 50 00 00 b
e005e596 dd 9e
```

Compare the Same Function



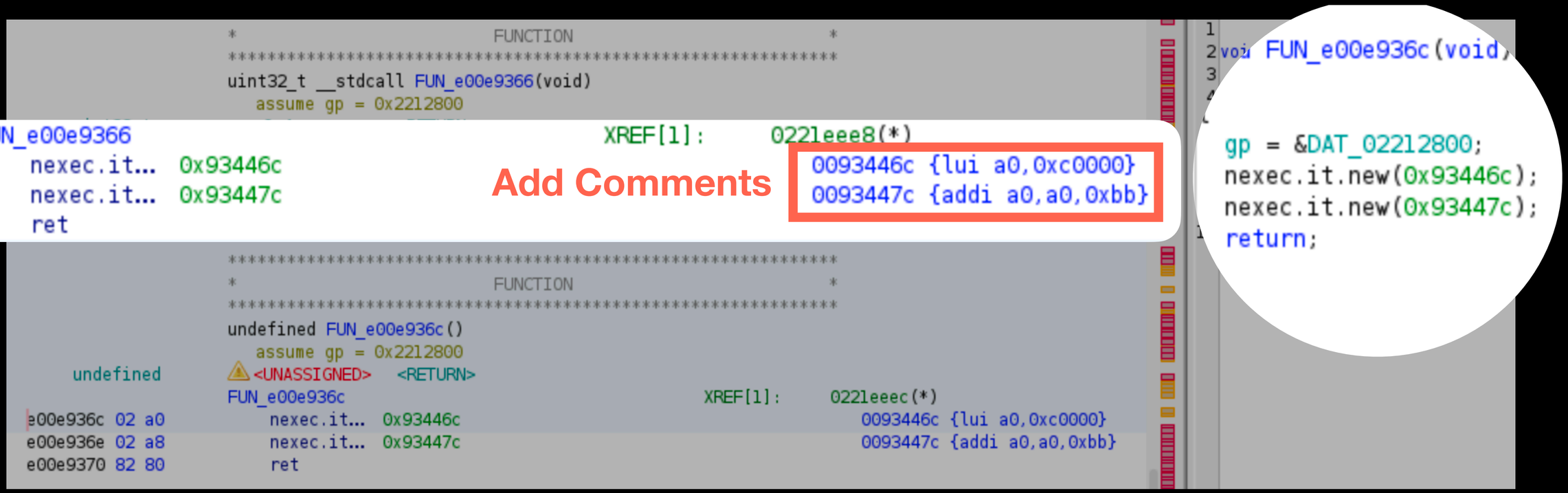
Instructions Fully Fixed

Can’t decompiled



Thanks to my colleague @h3xr4bb1t

Instructions Fully Fixed

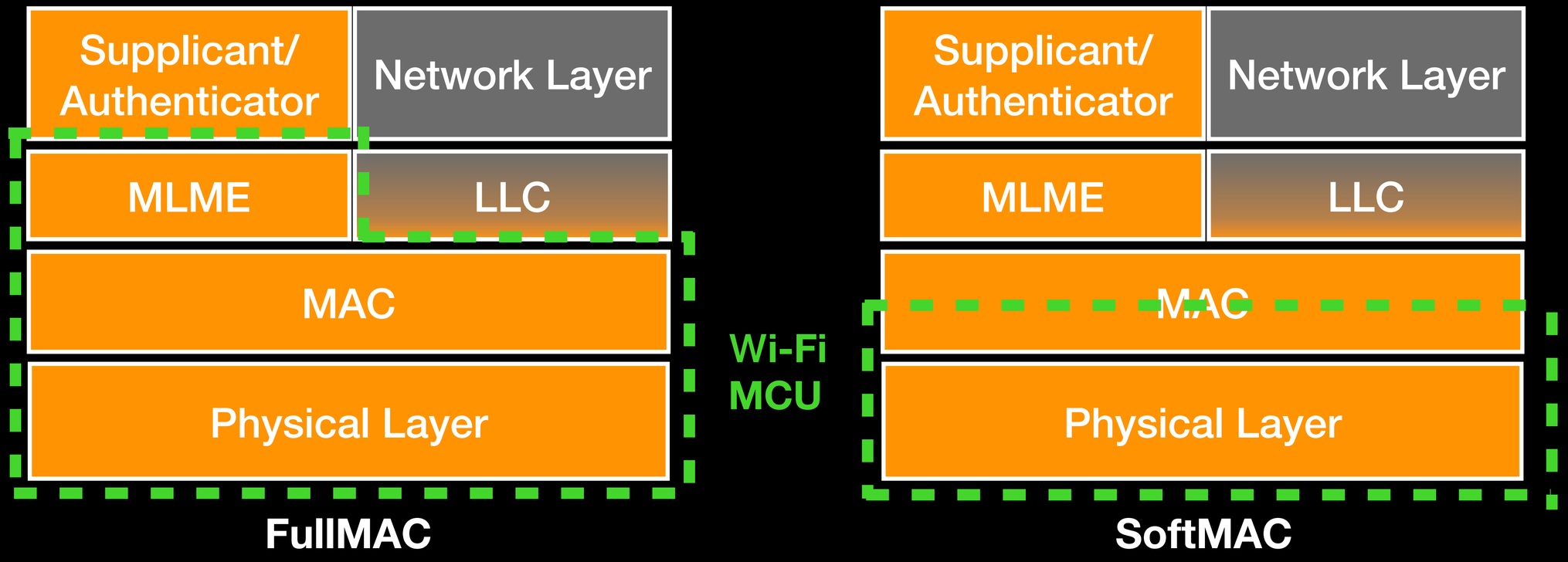


Thanks to my colleague @h3xr4bb1t

Instructions Fully Fixed

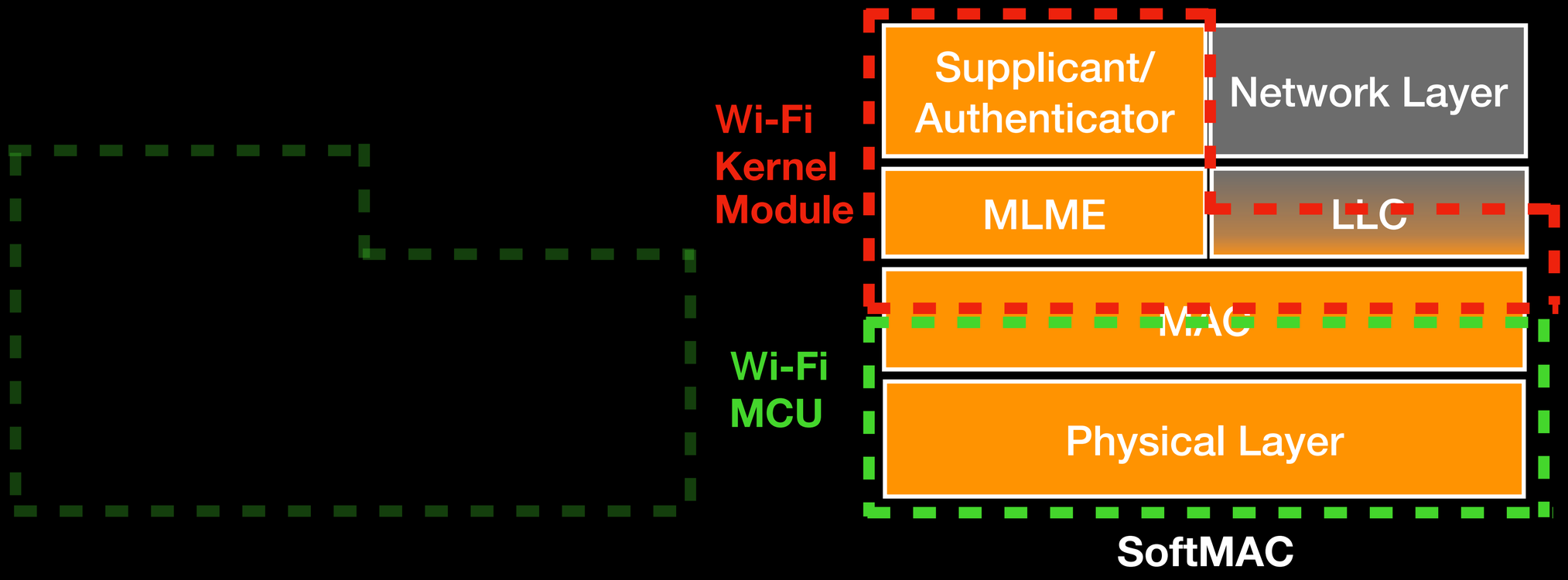
FullMAC v.s. SoftMAC

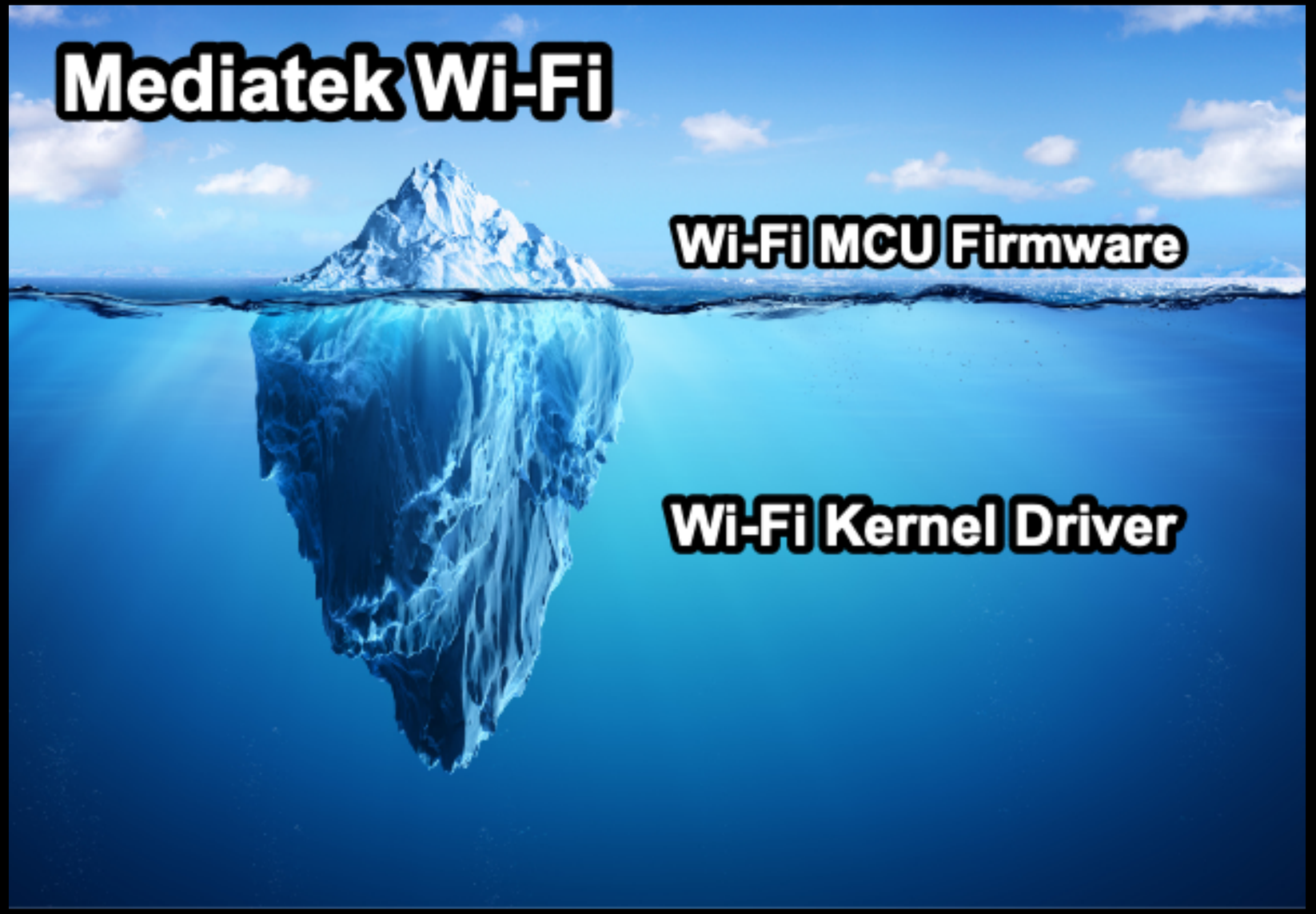
- The code related to the Wi-Fi frame handling is **minimal**



SoftMAC

- The code related to the Wi-Fi frame handling is **minimal**, so it should be **SoftMAC**




Mediatek Wi-Fi
Wi-Fi MCU Firmware
Wi-Fi Kernel Driver

The Forgotten Frame Injection

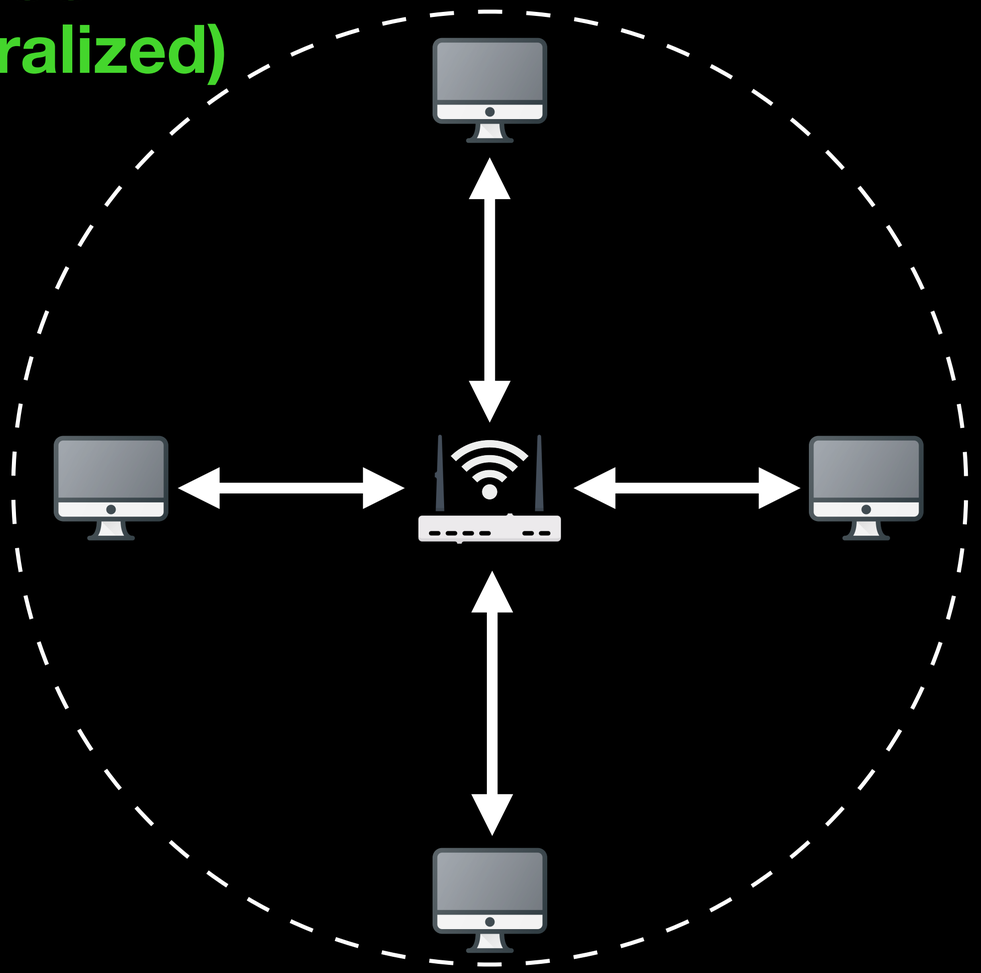
DEVCORE

Type of Wi-Fi network

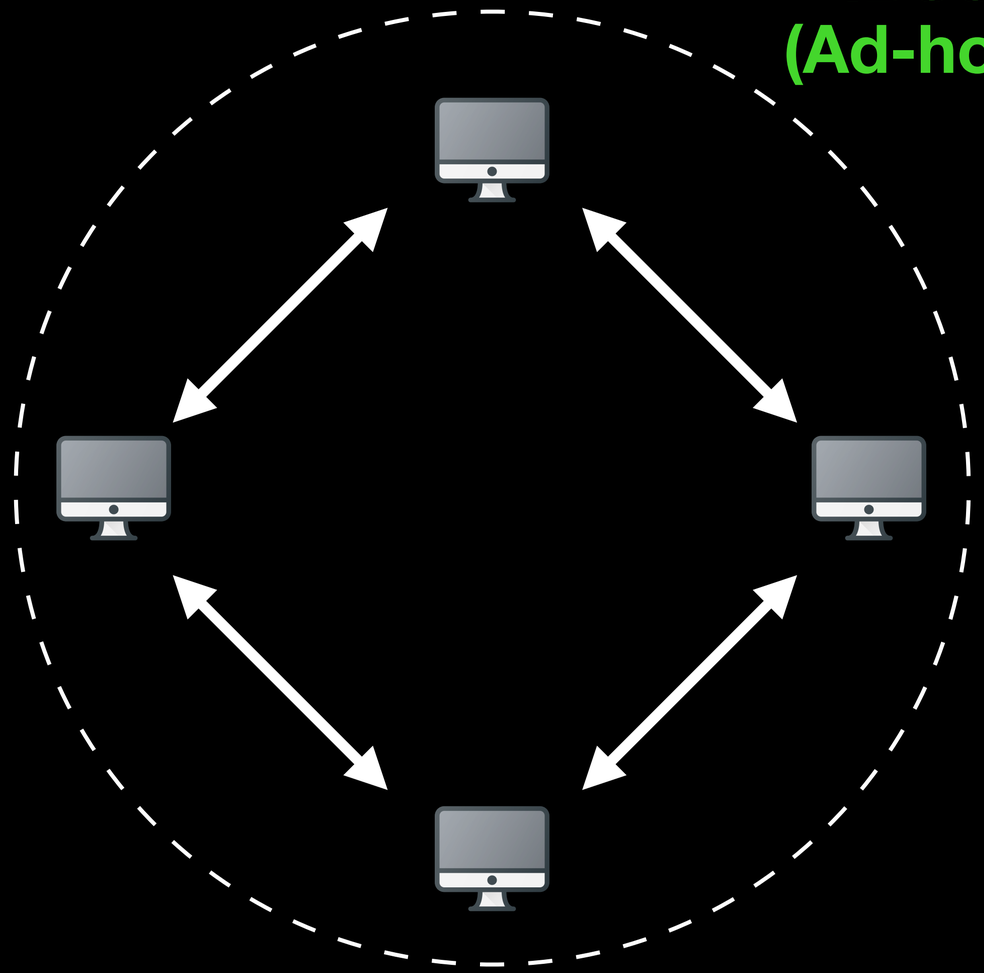
Access Point (AP)

Station (STA)

BSS
(Centralized)



IBSS
(Ad-hoc)

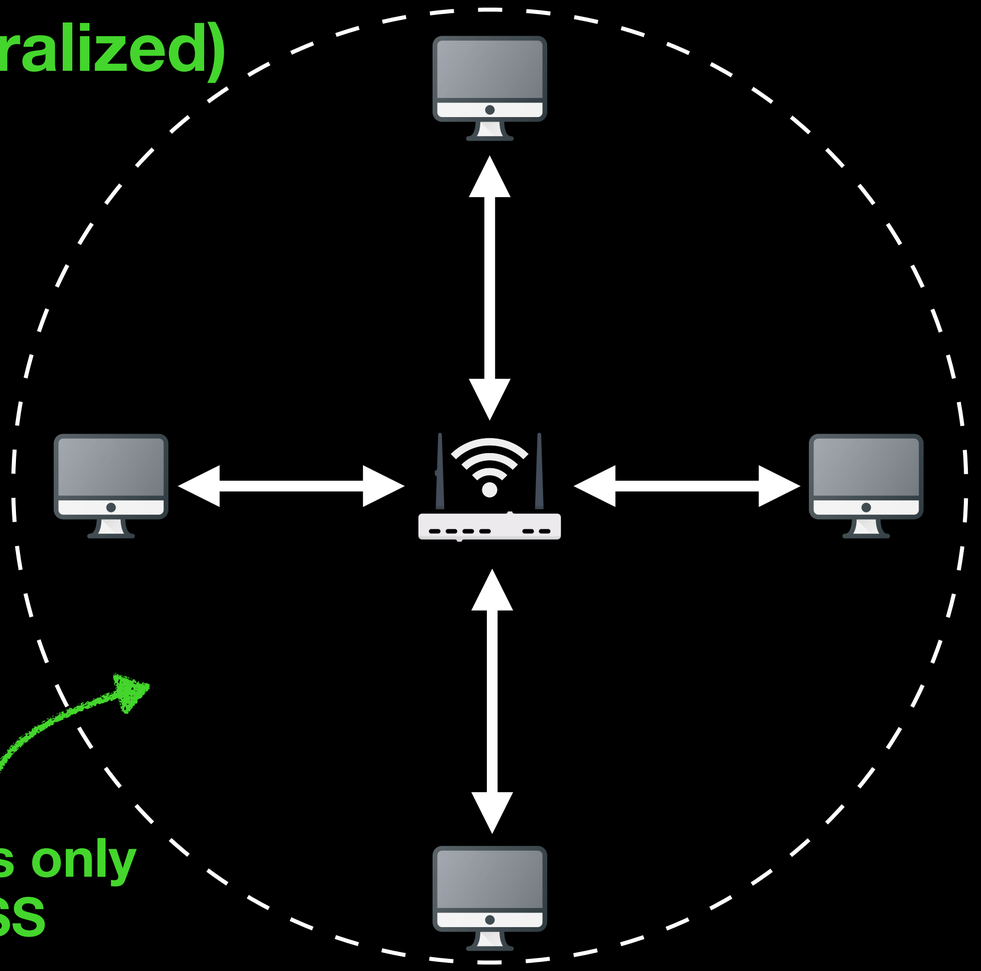


Type of Wi-Fi network

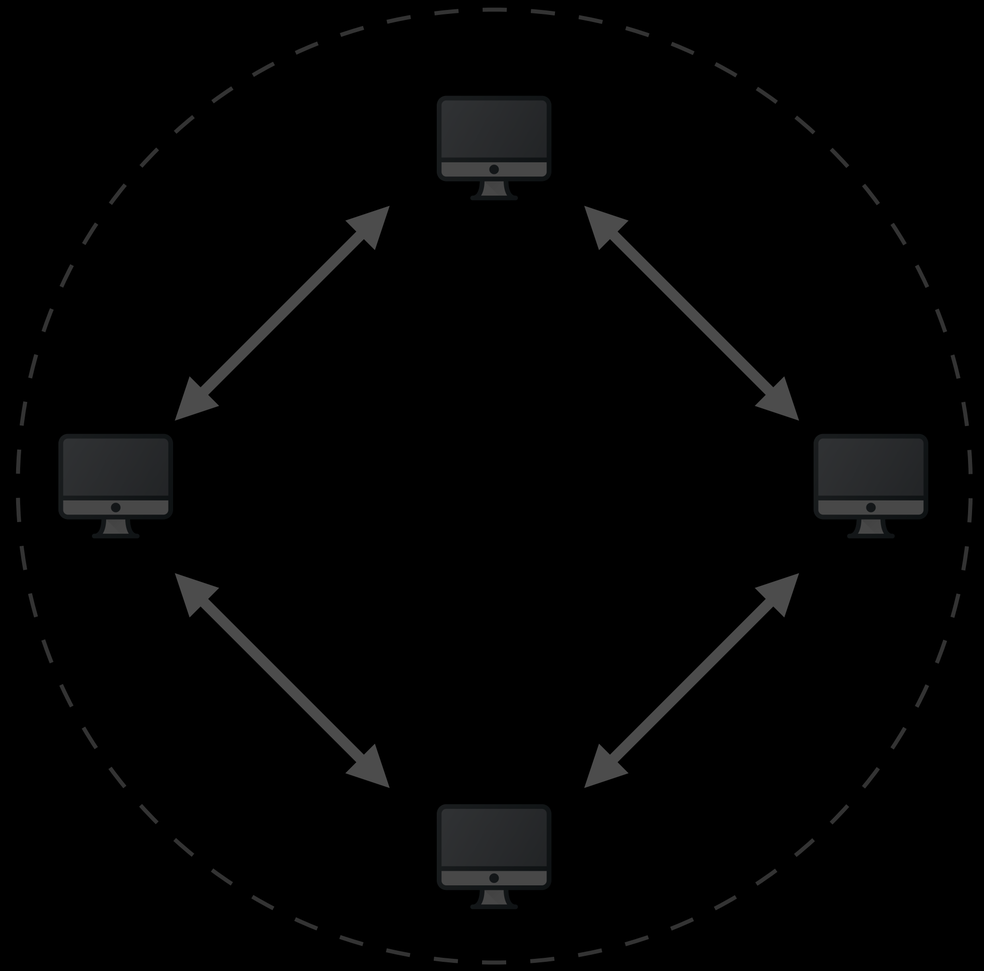
Access Point (AP)

Station (STA)

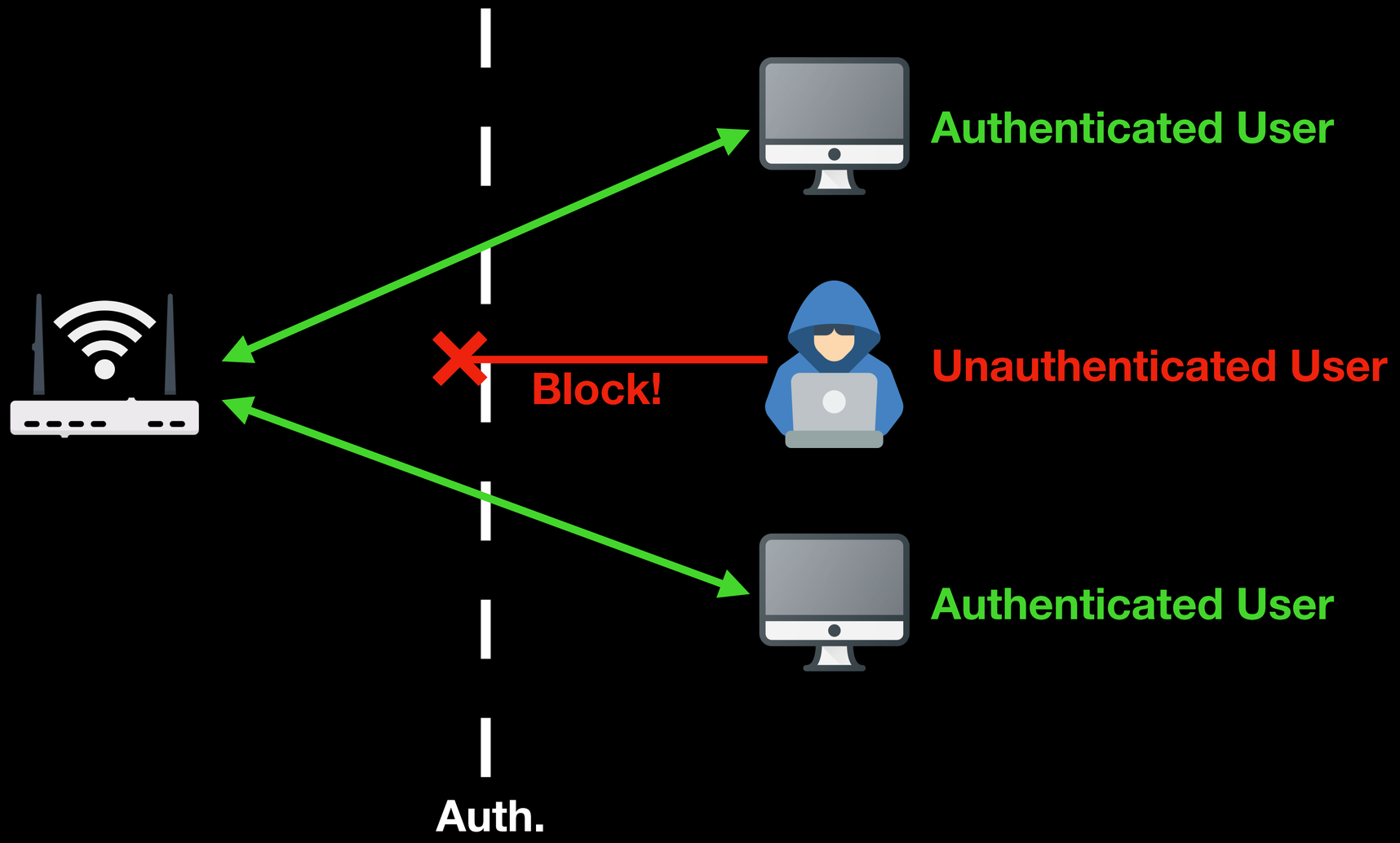
BSS (Centralized)



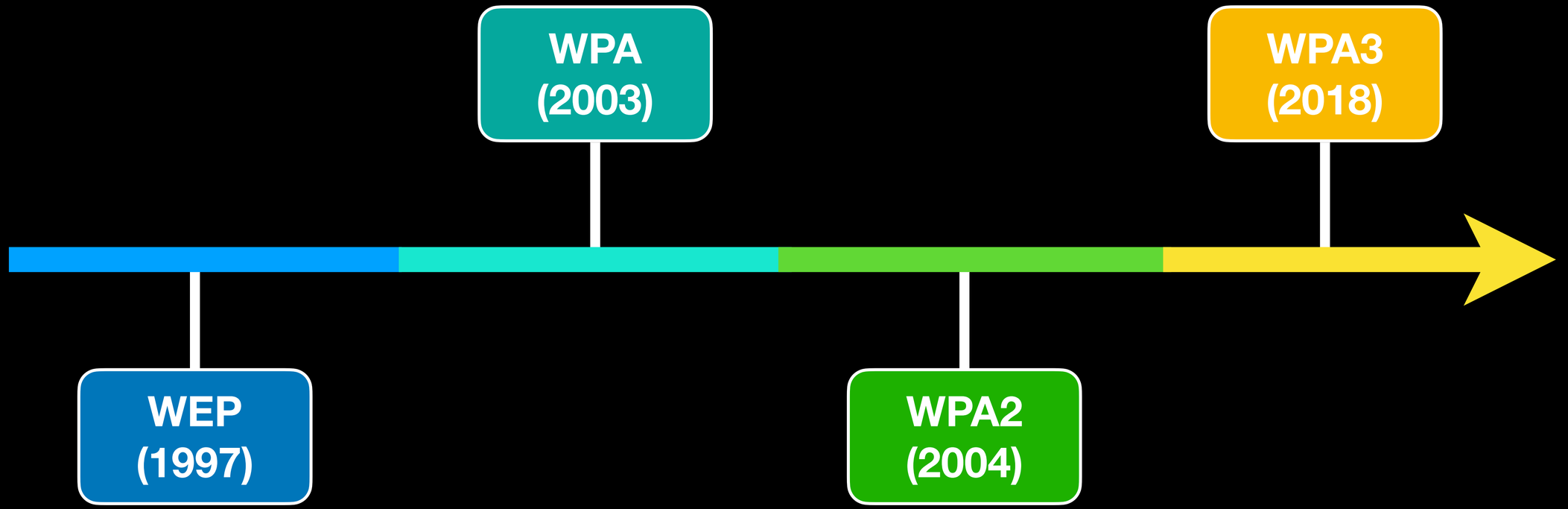
Focus only on BSS



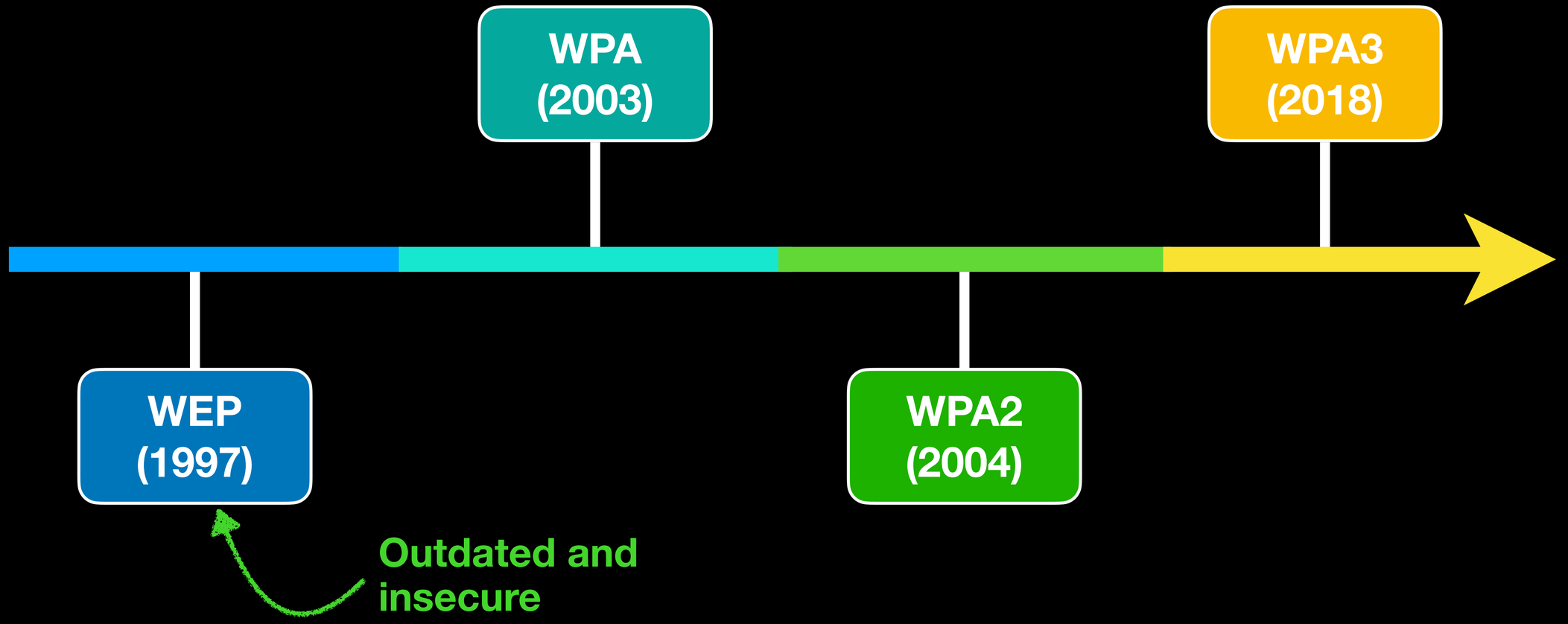
Wi-Fi Authentication



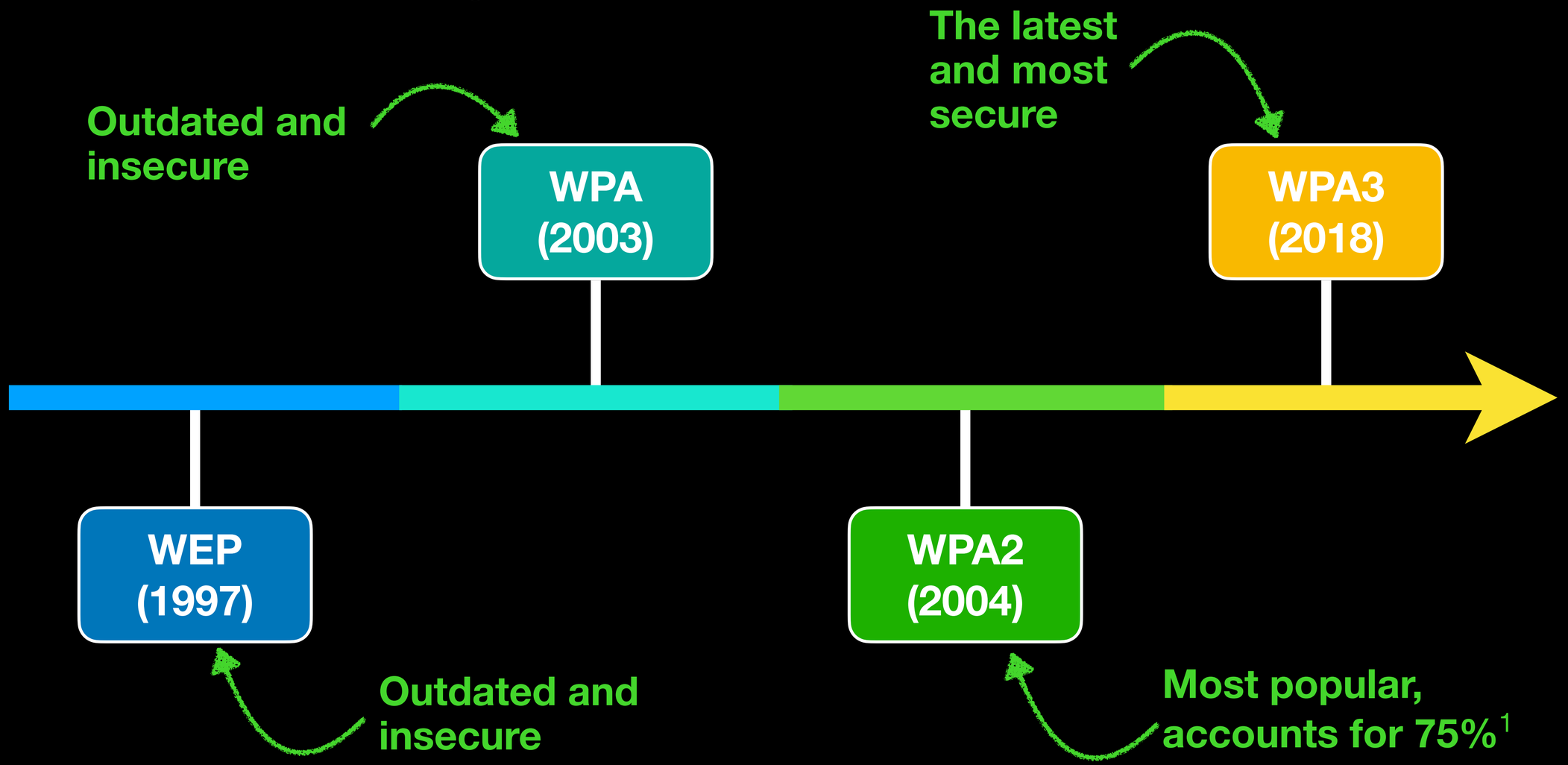
Wi-Fi Security Protocols



Wi-Fi Security Protocols

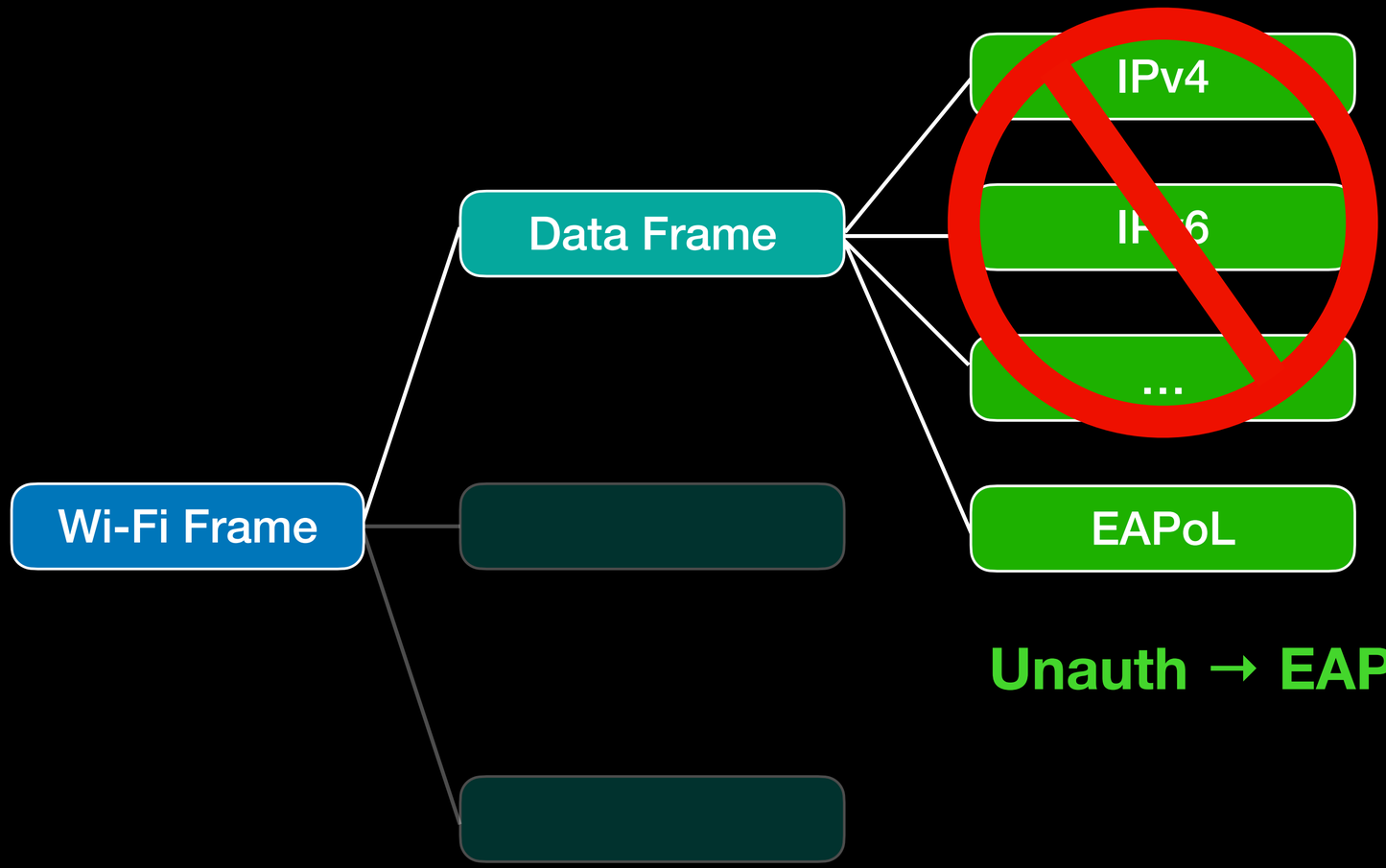


Wi-Fi Security Protocols



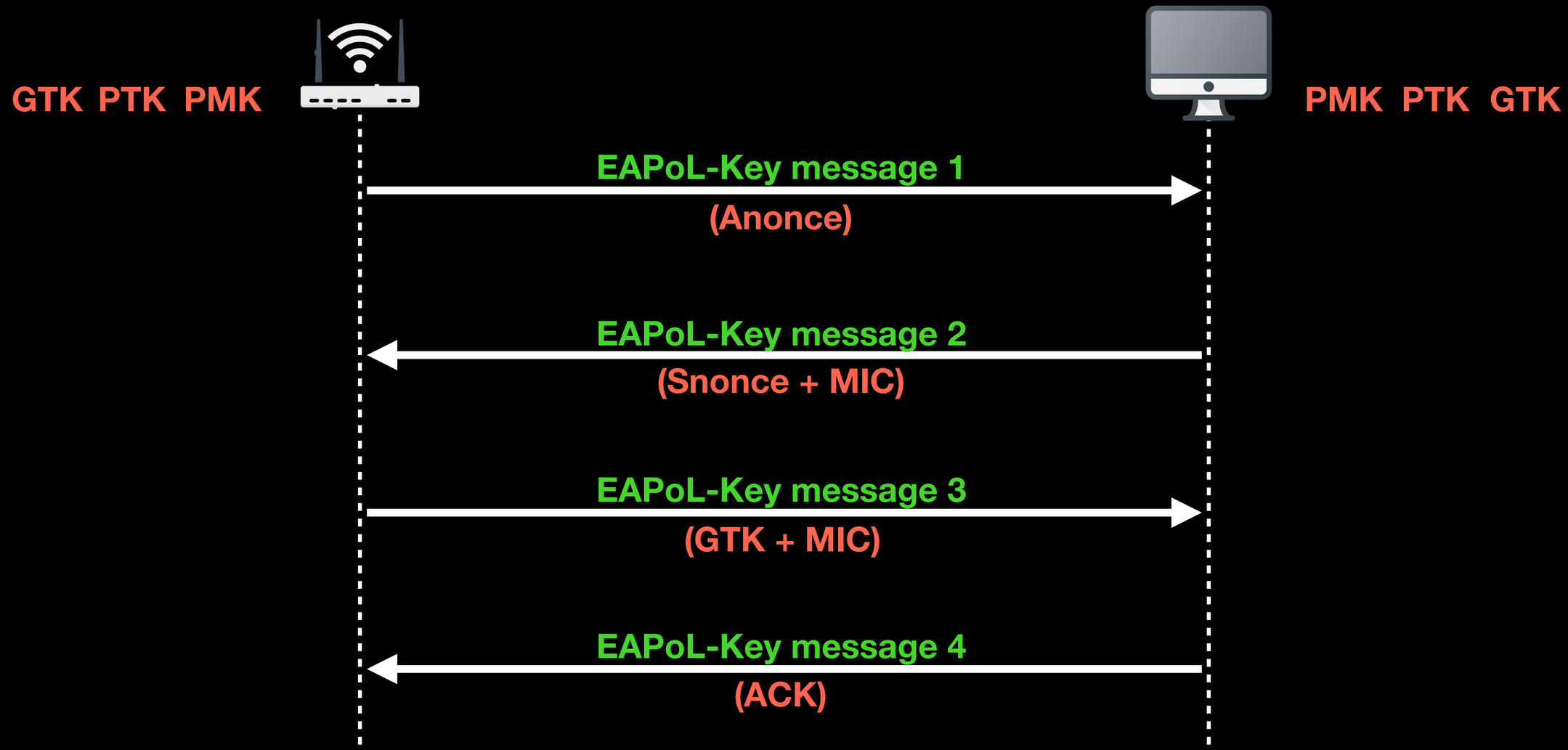
[1] https://wigle.net/stats

Unauthenticated User

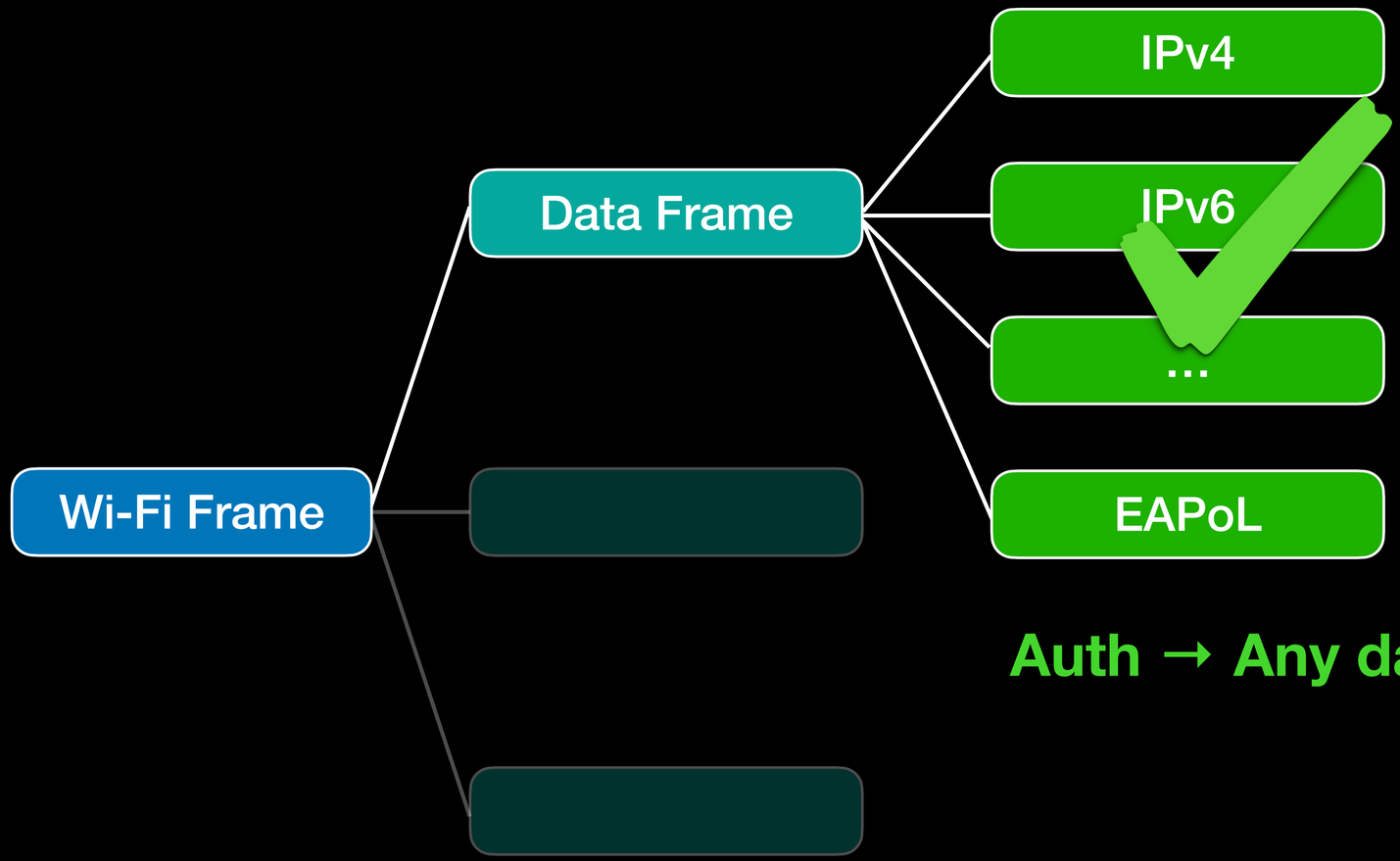


Unauth → EAPOL only

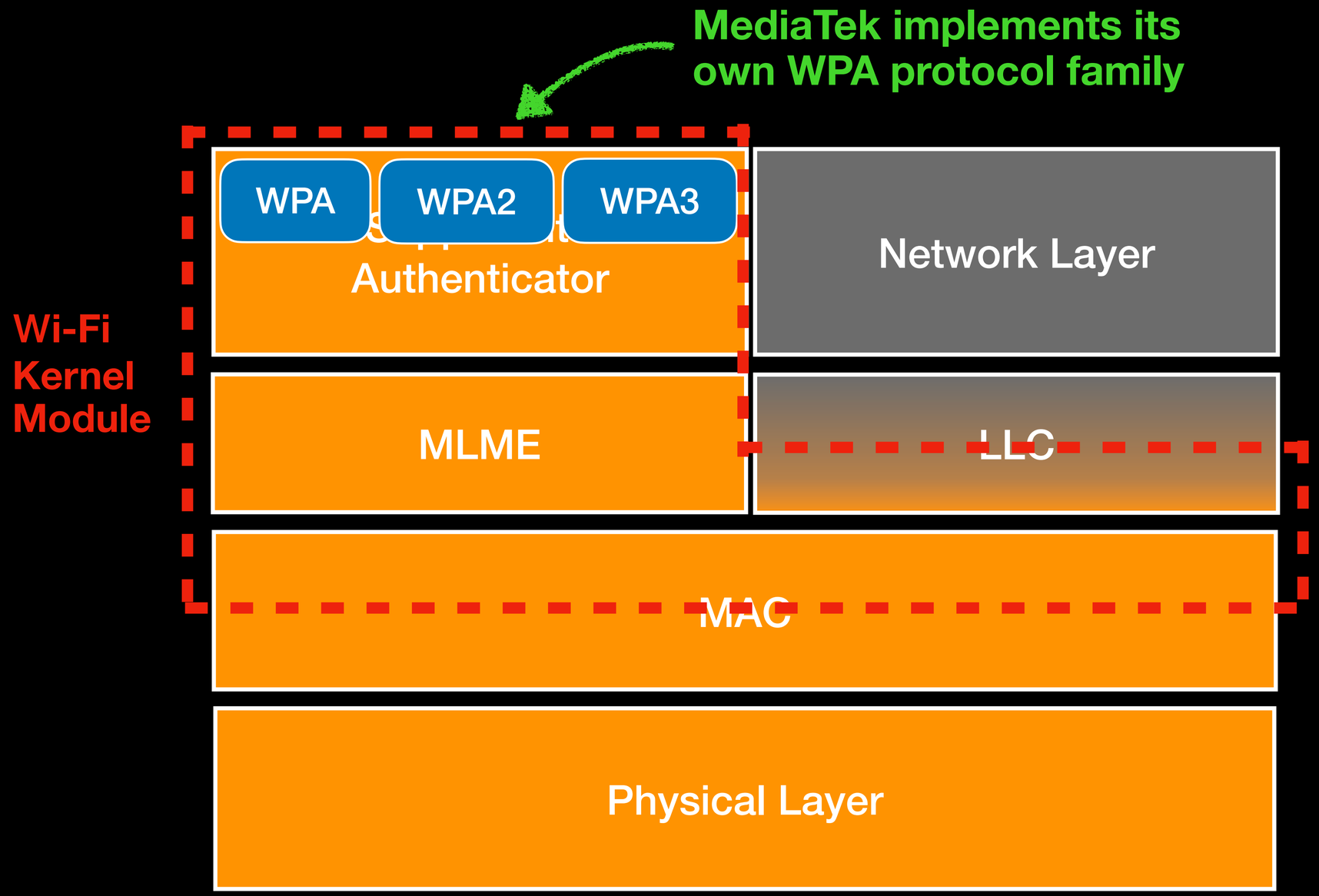
WPA2 4-way Handshake

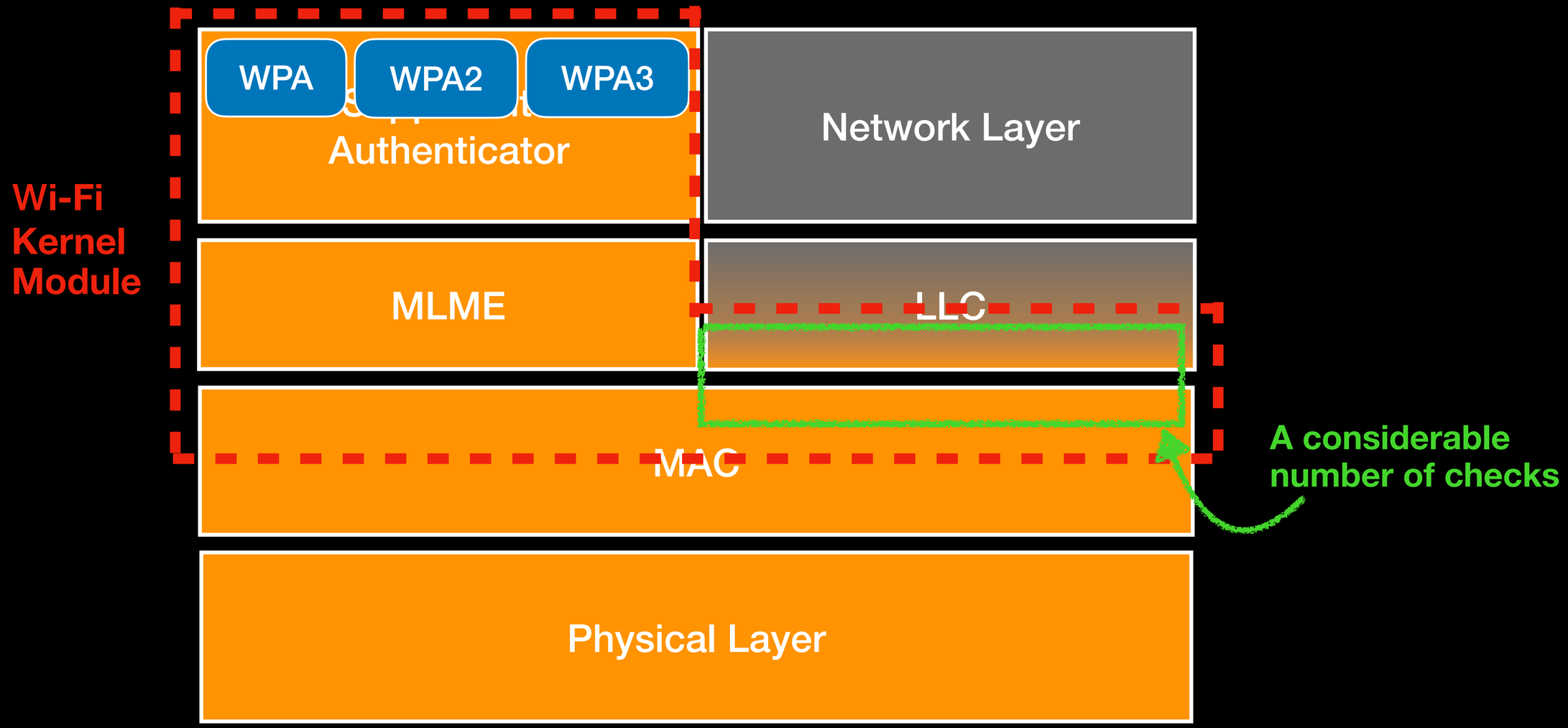


Authenticated User



Auth → Any data frame

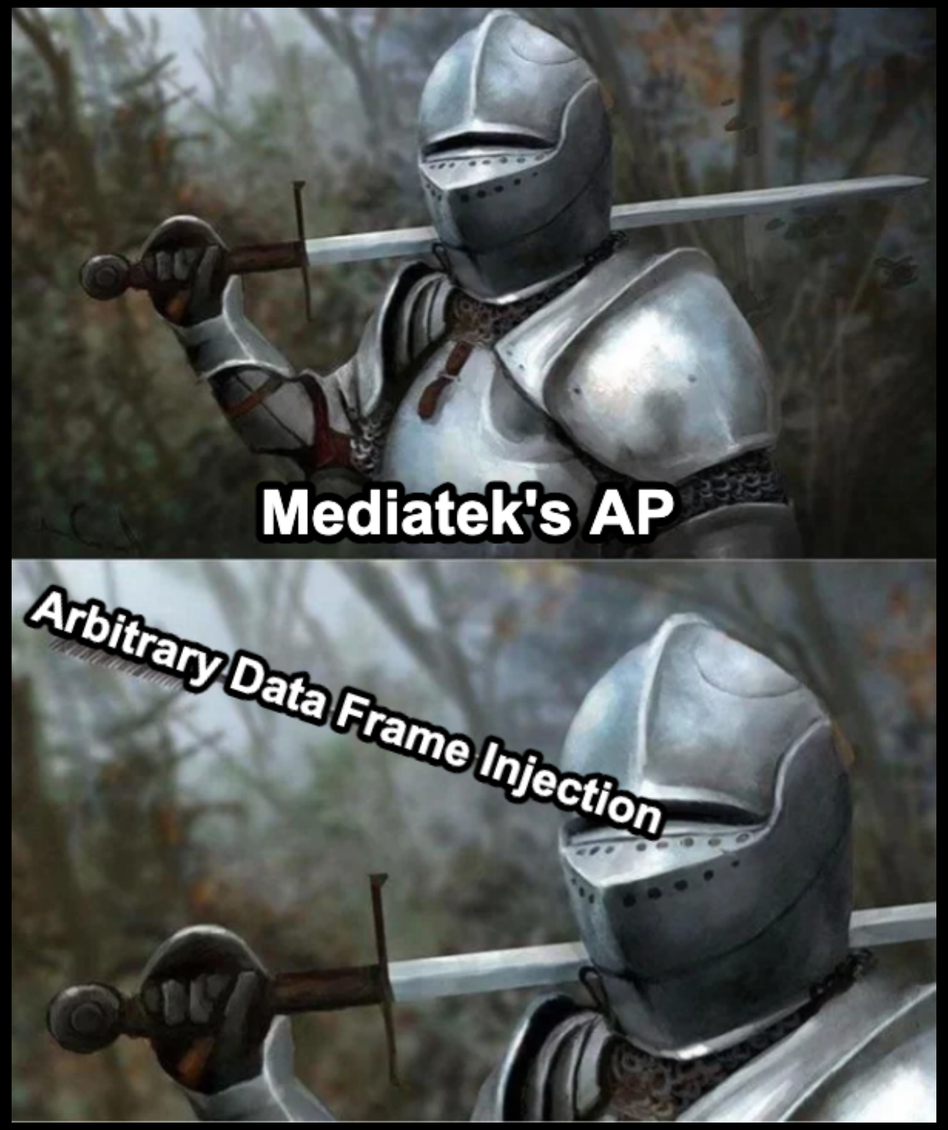

MediaTek implements its own WPA protocol family
Wi-Fi Kernel Module
WPA
WPA2
WPA3
Authenticator
Network Layer
MLME
LLC
MAC
Physical Layer


WPA
WPA2
WPA3
Authenticator
Network Layer
Wi-Fi
Kernel
Module
MLME
LLC
MAC
A considerable
number of checks
Physical Layer

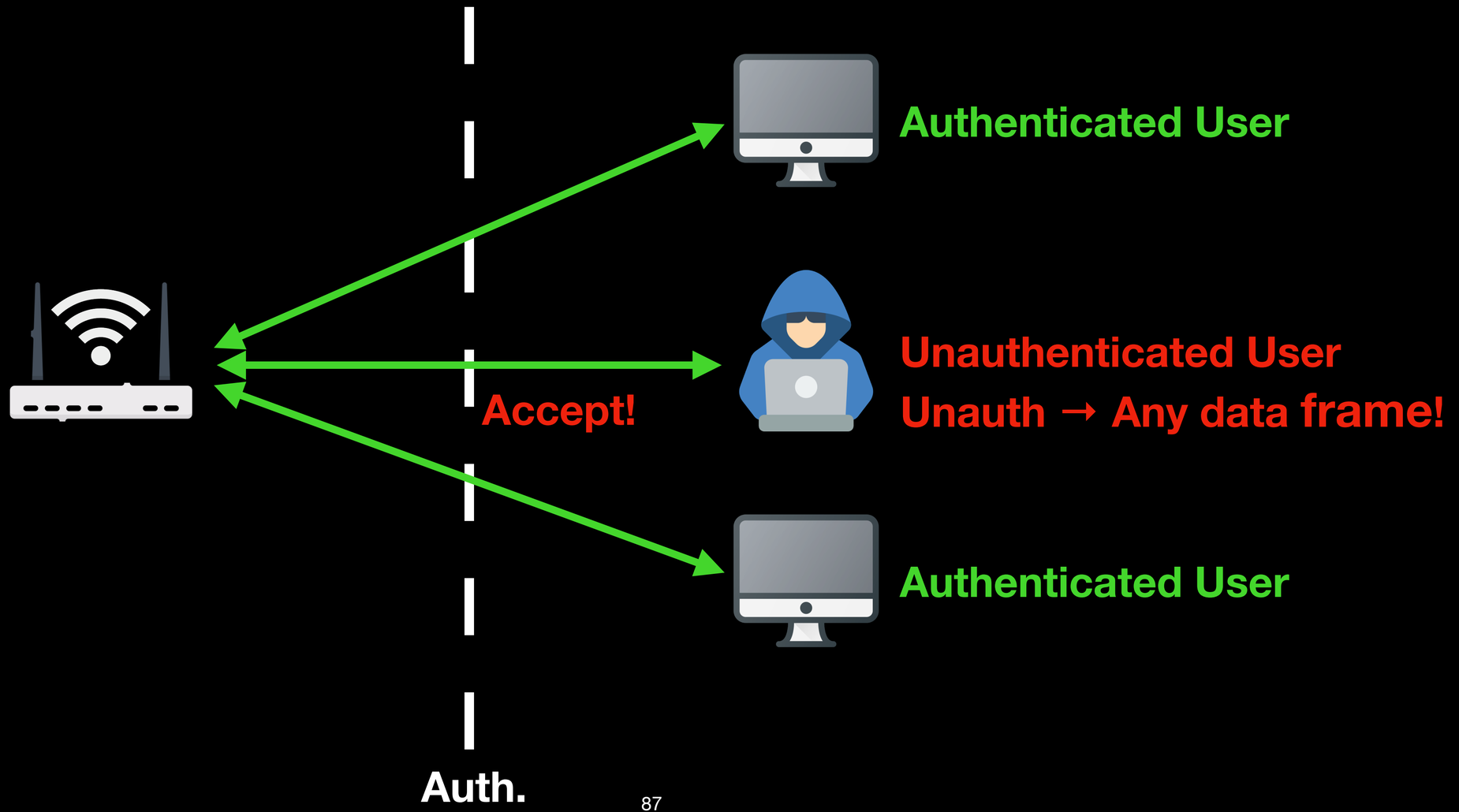
Wi-Fi
Kernel
Module

A considerable
number of checks

CVE-2025-20674

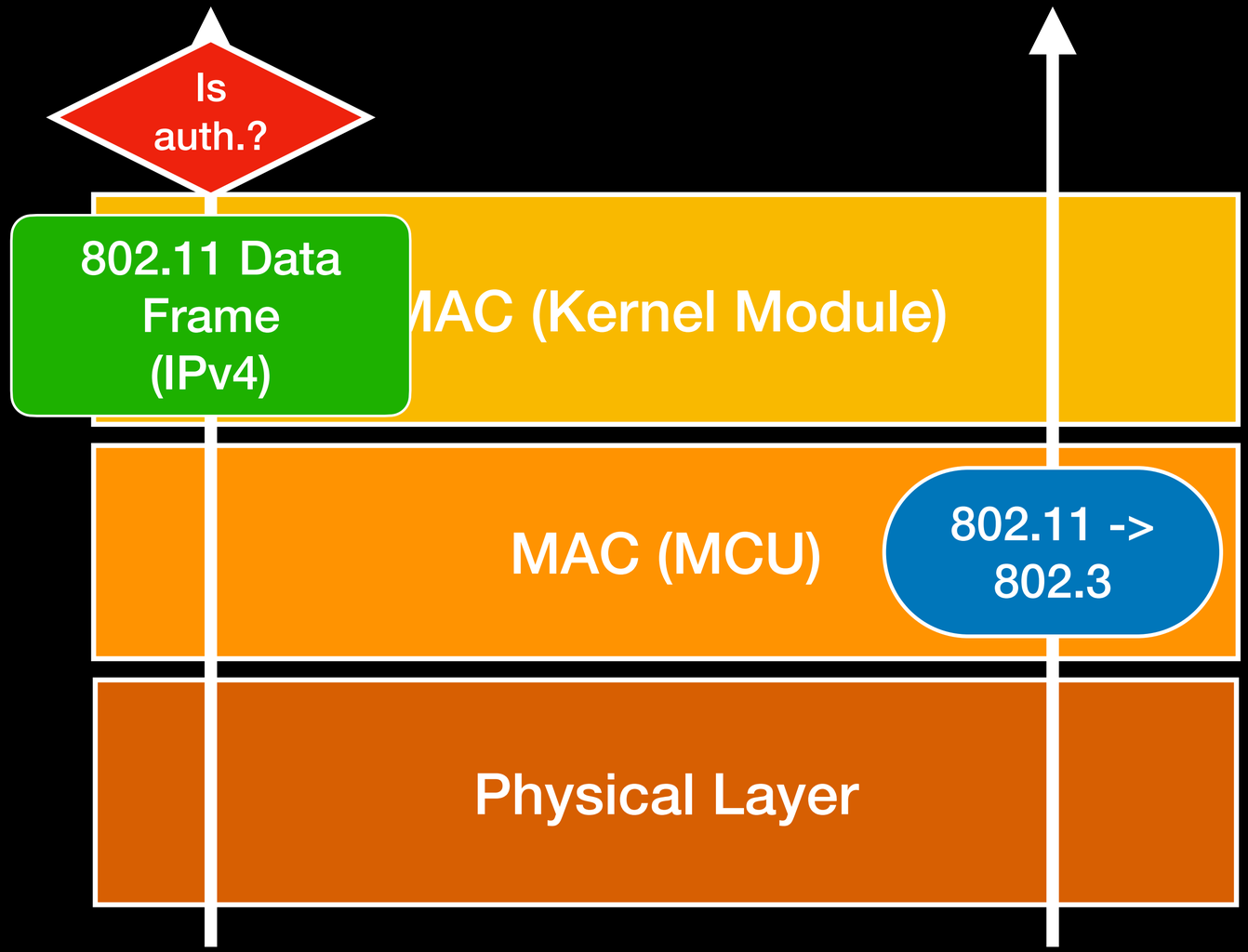


Wi-Fi ~~Authentication~~



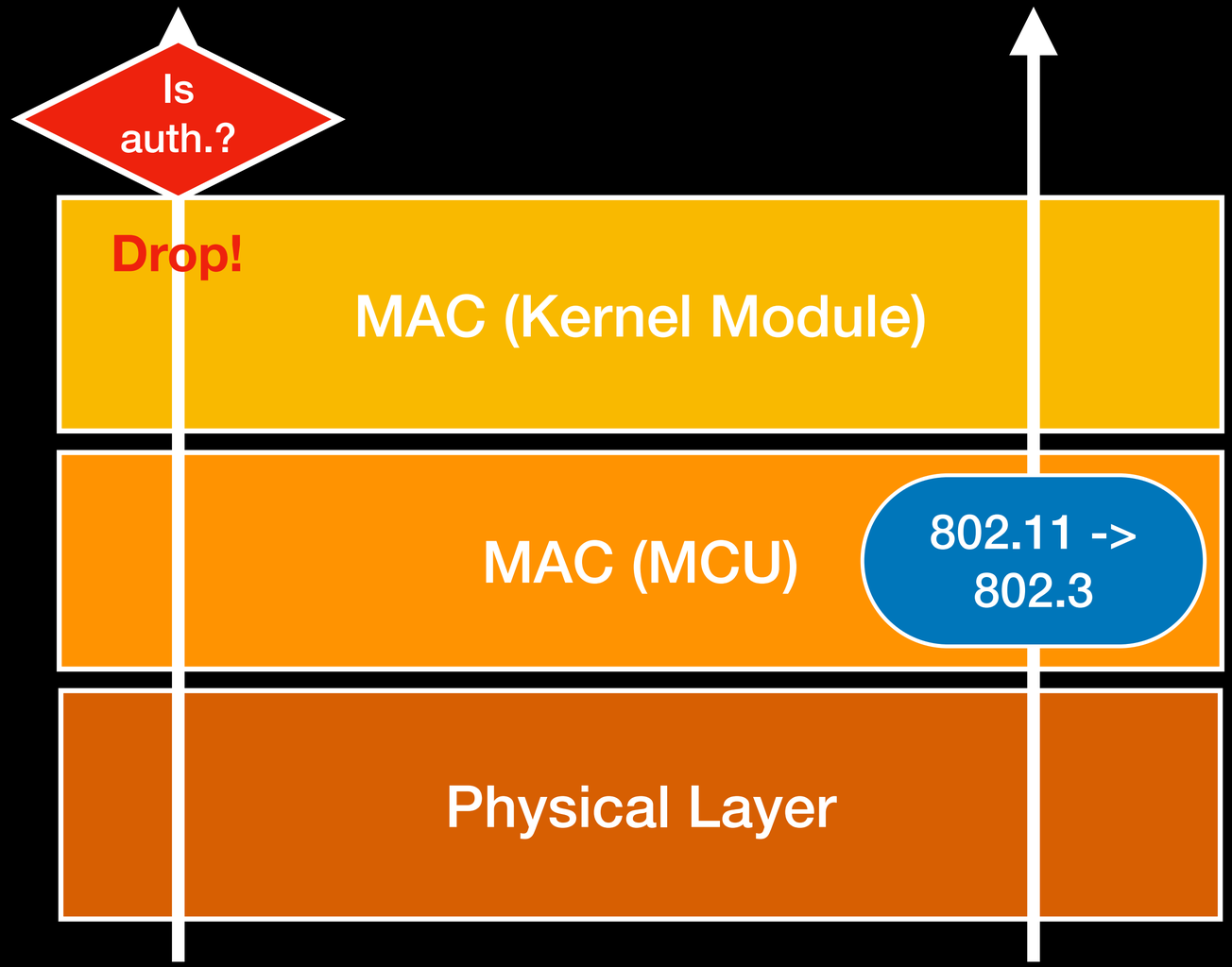
How did this happened?

From unauth. user

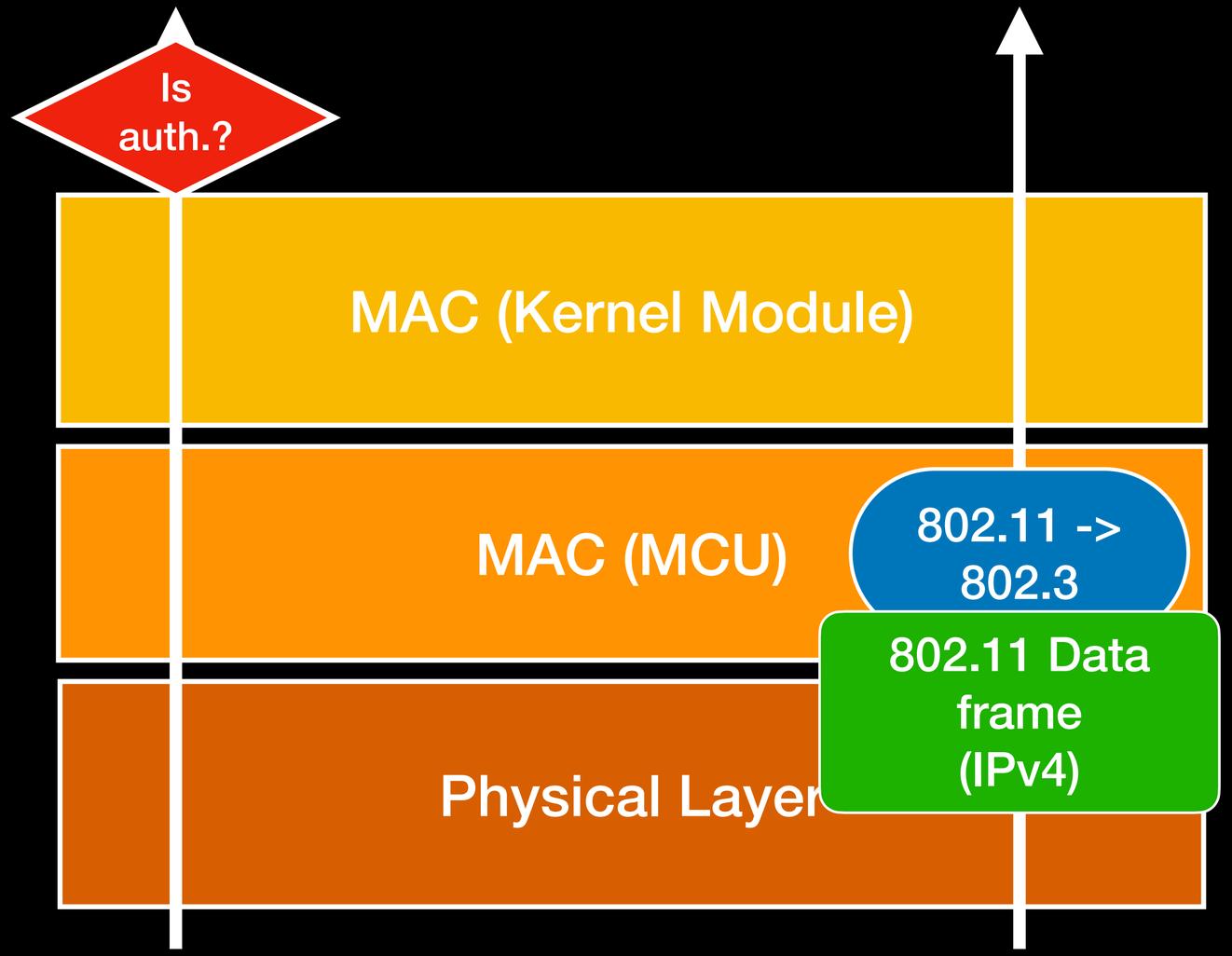


How did this happened?

Drop if source is unauth.

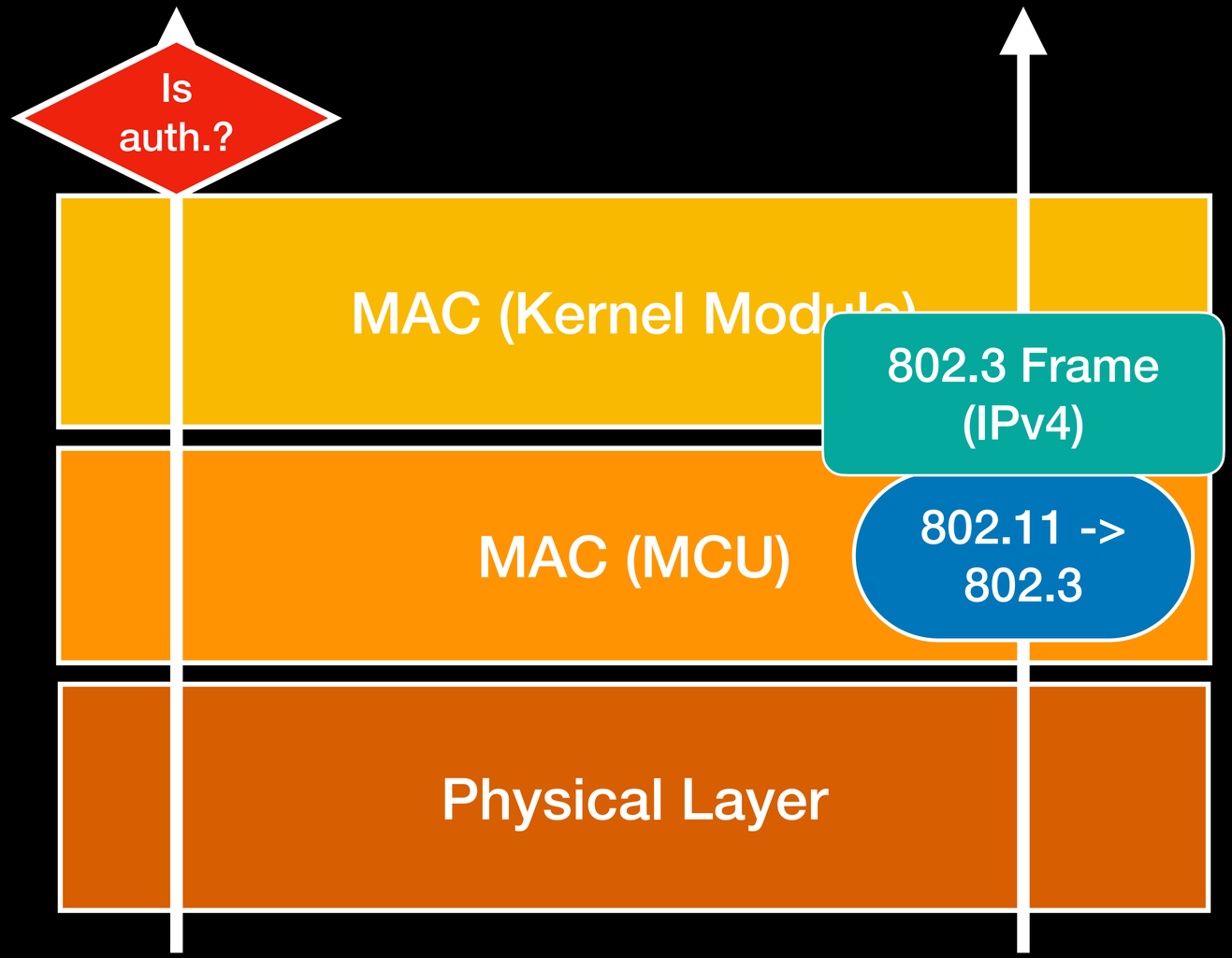


How did this happened?



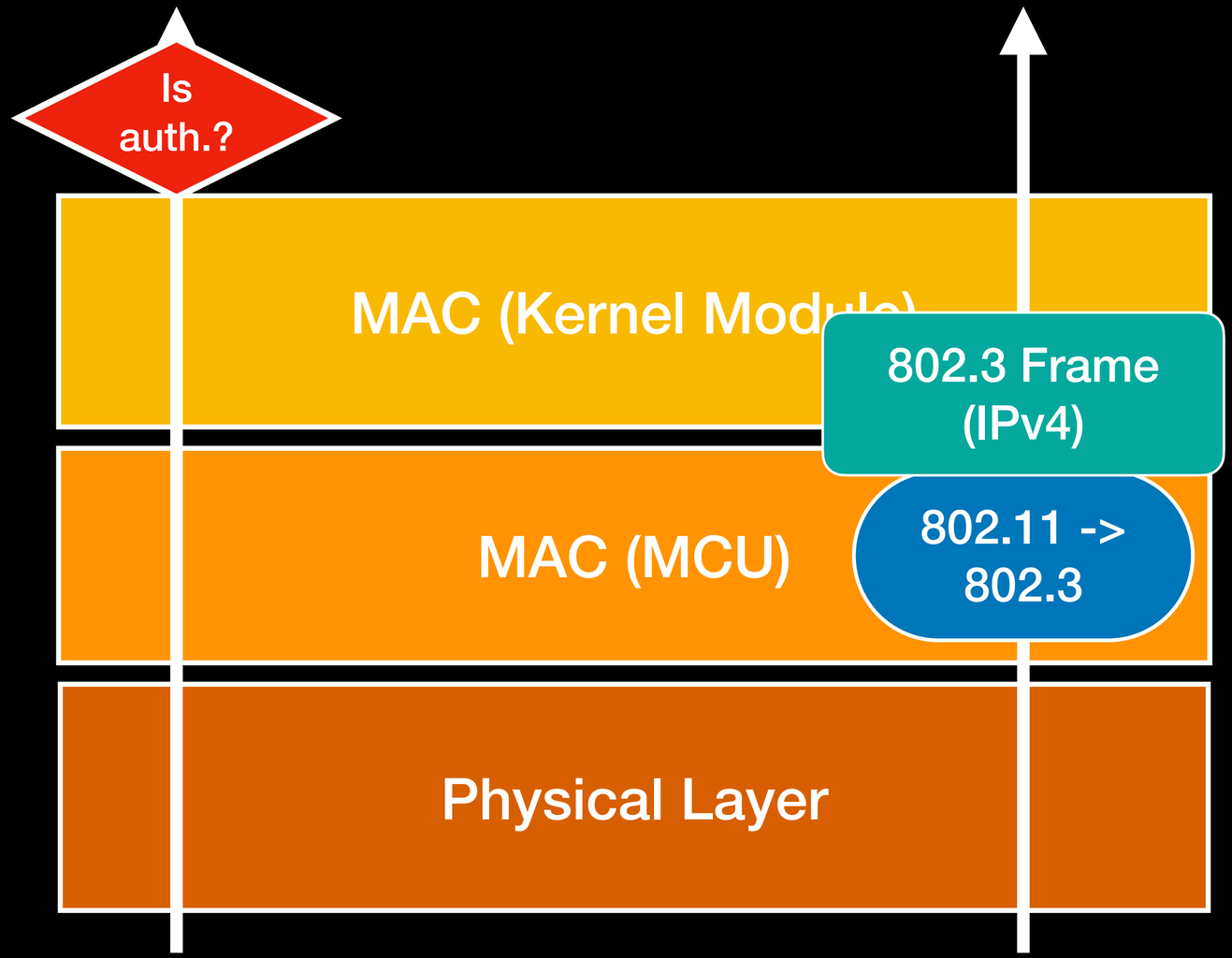
From unauth. user

How did this happened?

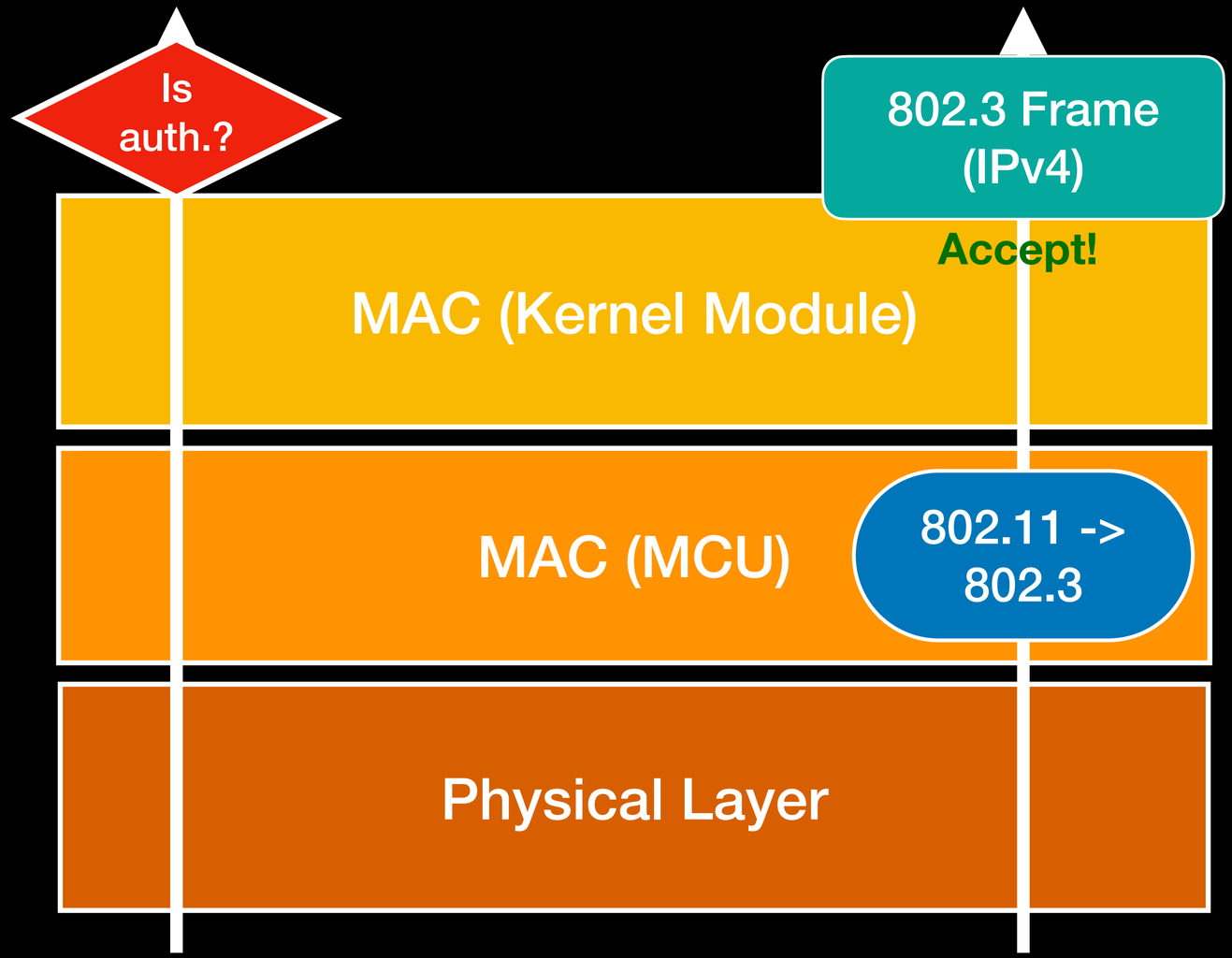


Convert Wi-Fi frame into ethernet frame

How did this happened?



How did this happened?



Missing check!

FragAttacks

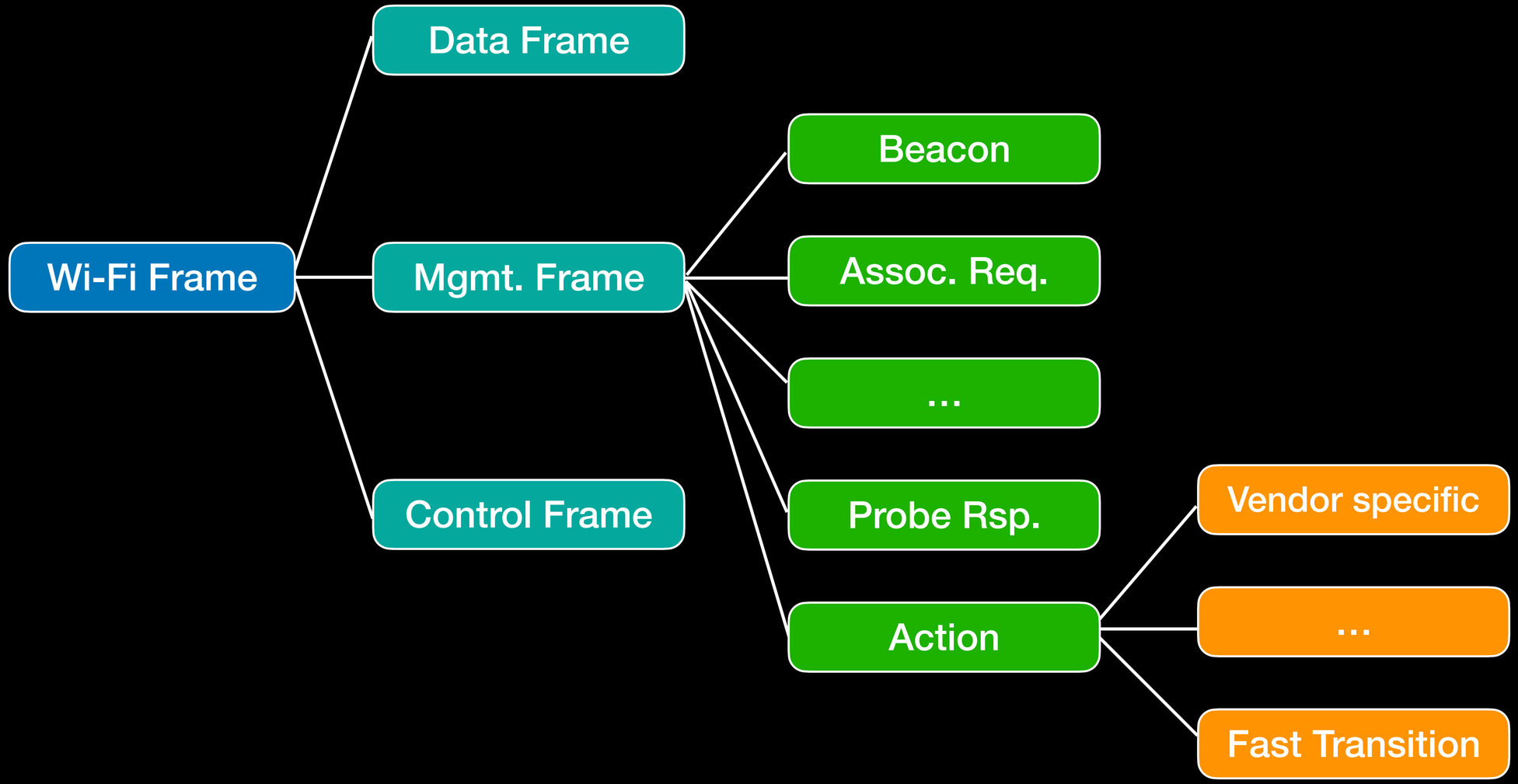
- 2017 - Broadpwn: Remotely Compromising Android and iOS via a Bug in Broadcom's Wi-Fi Chipsets
- 2018 - Researching Marvell Avastar Wi-Fi: From Zero Knowledge to Over-the-Air Zero-Touch RCE
- 2019 - Exploiting Qualcomm WLAN and Modem Over the Air
- 2021 - Fragment and Forge: Breaking Wi-Fi Through Frame Aggregation and Fragmentation
- 2021 - Qualcomm WiFi: Infinity War
- 2022 - BadMesher: New Attack Surfaces of Wi-Fi Mesh Network
- 2022 - Ghost in the Wireless, iwlwifi edition
- 2022 - WIFI Security - From 0 To 1
- 2024 - Listen-Up: Sonos Over-The-Air Remote Kernel Exploitation and Covert Wiretap
- 2025 - Unlock hidden Superpowers in MediaTek Wi-Fi Chips

It has demonstrated the exploit mechanism

DEVCORE

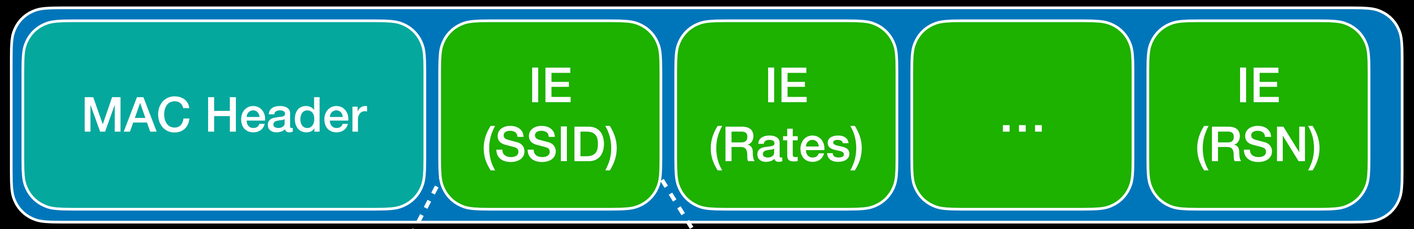
One Frame to Break Them All

DEVCORE

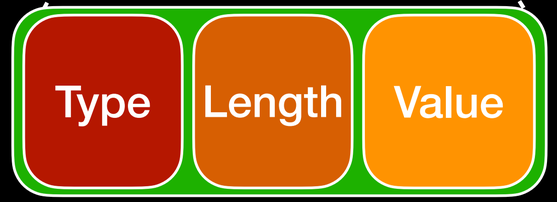

Wi-Fi Frame
Data Frame
Mgmt. Frame
Control Frame
Beacon
Assoc. Req.
...
Probe Rsp.
Action
Vendor specific
...
Fast Transition

How to Exchange Information?

Beacon frame

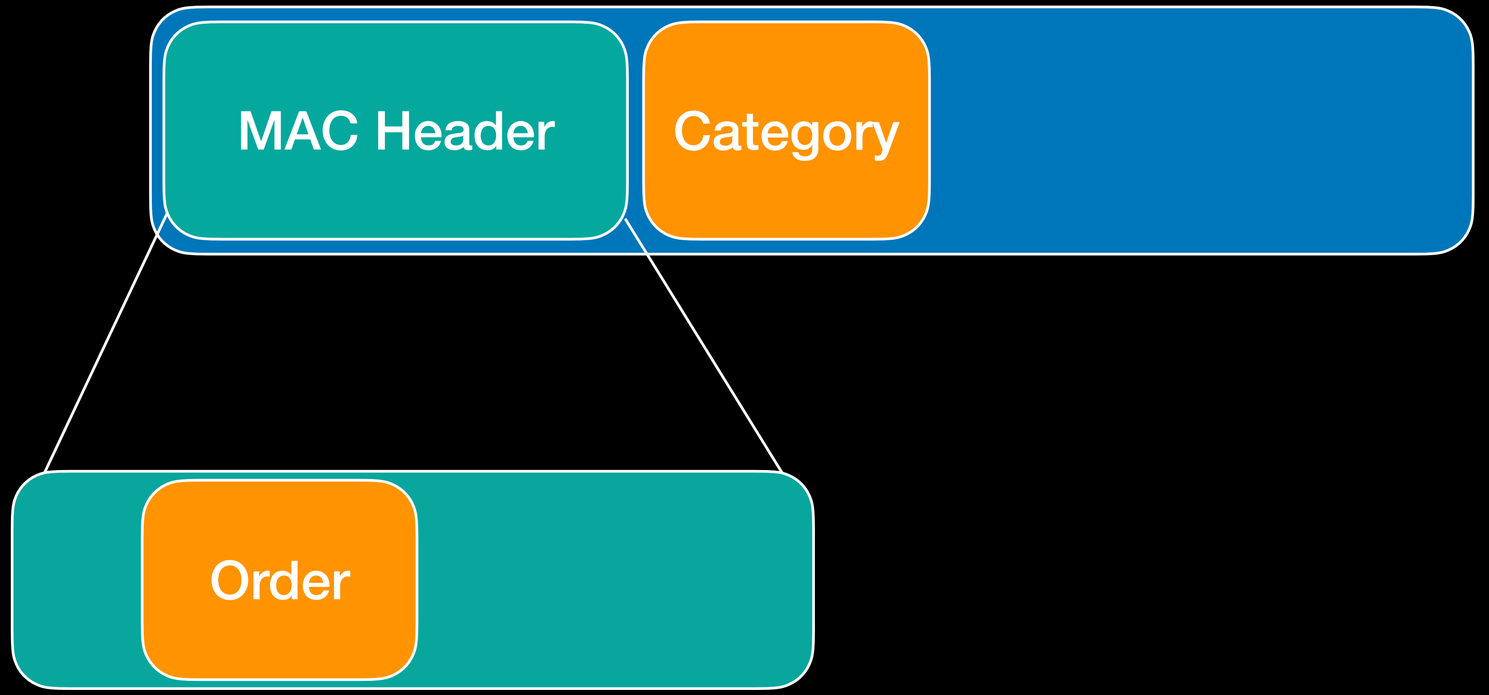


Information Element (IE)

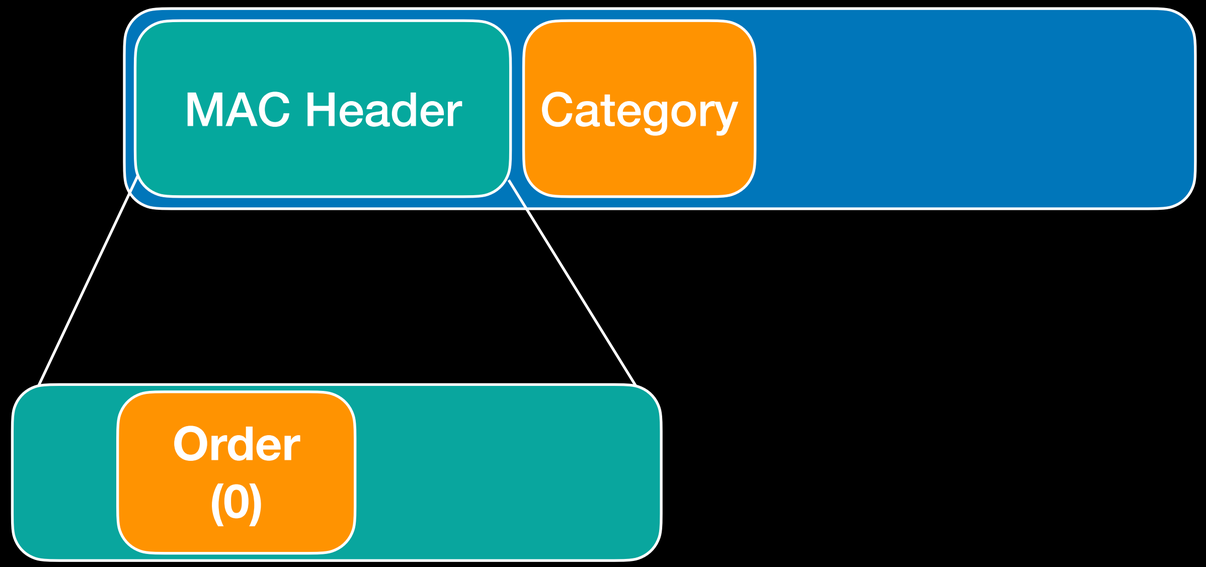


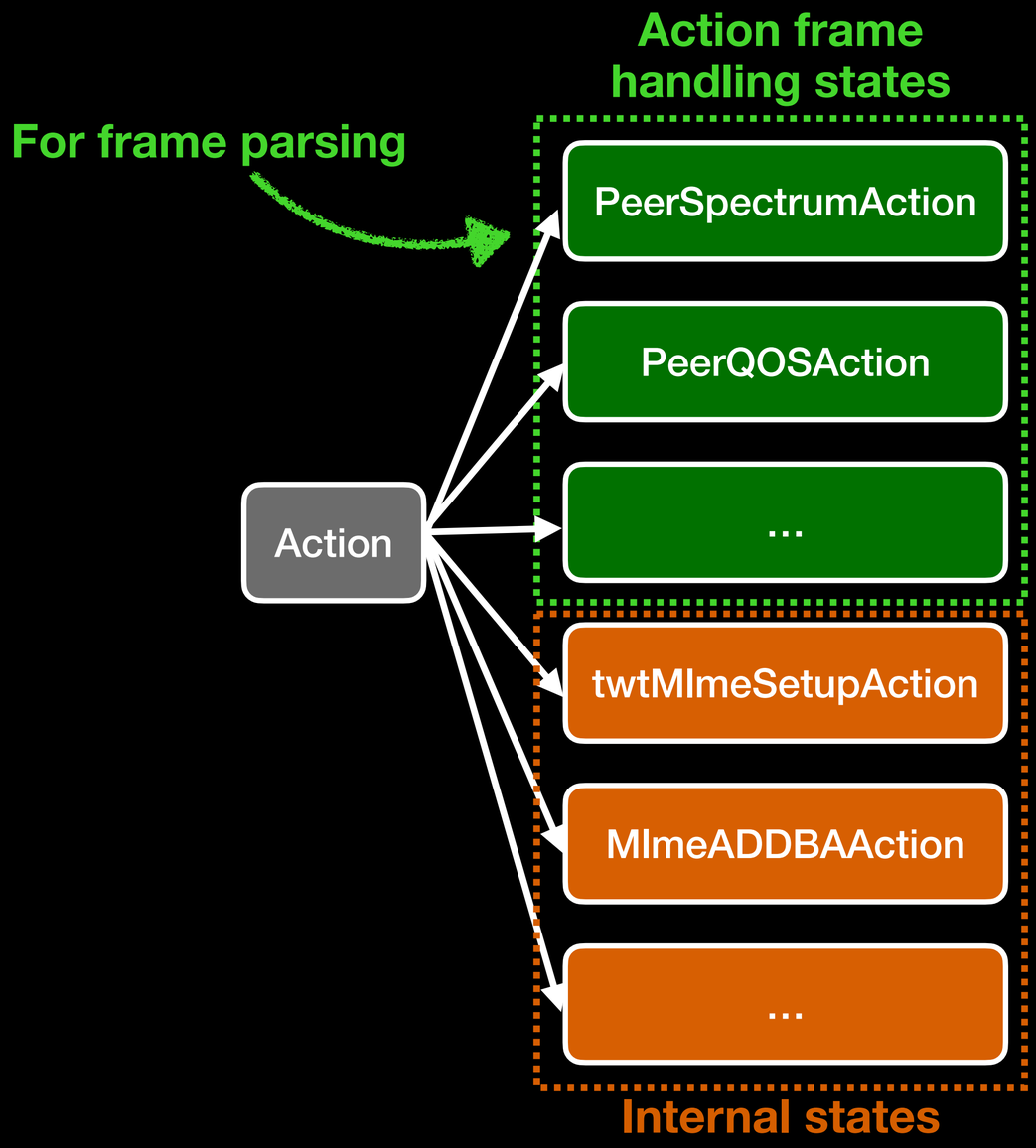
CVE-2025-20711

- In action frame, It doesn’t check the range of category when order bit in MAC header is set

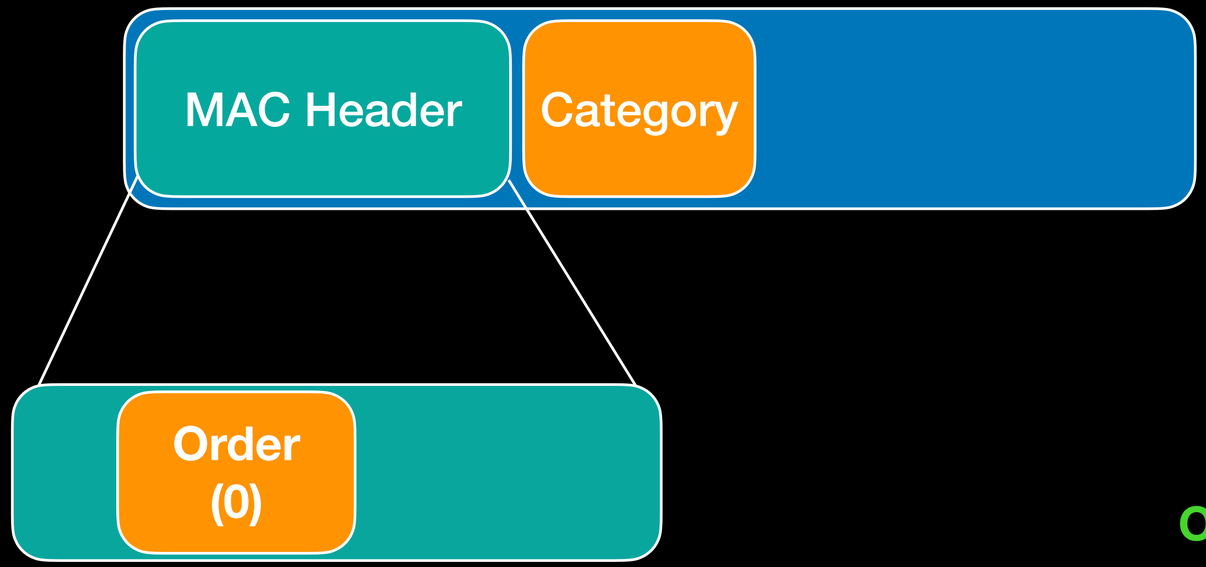


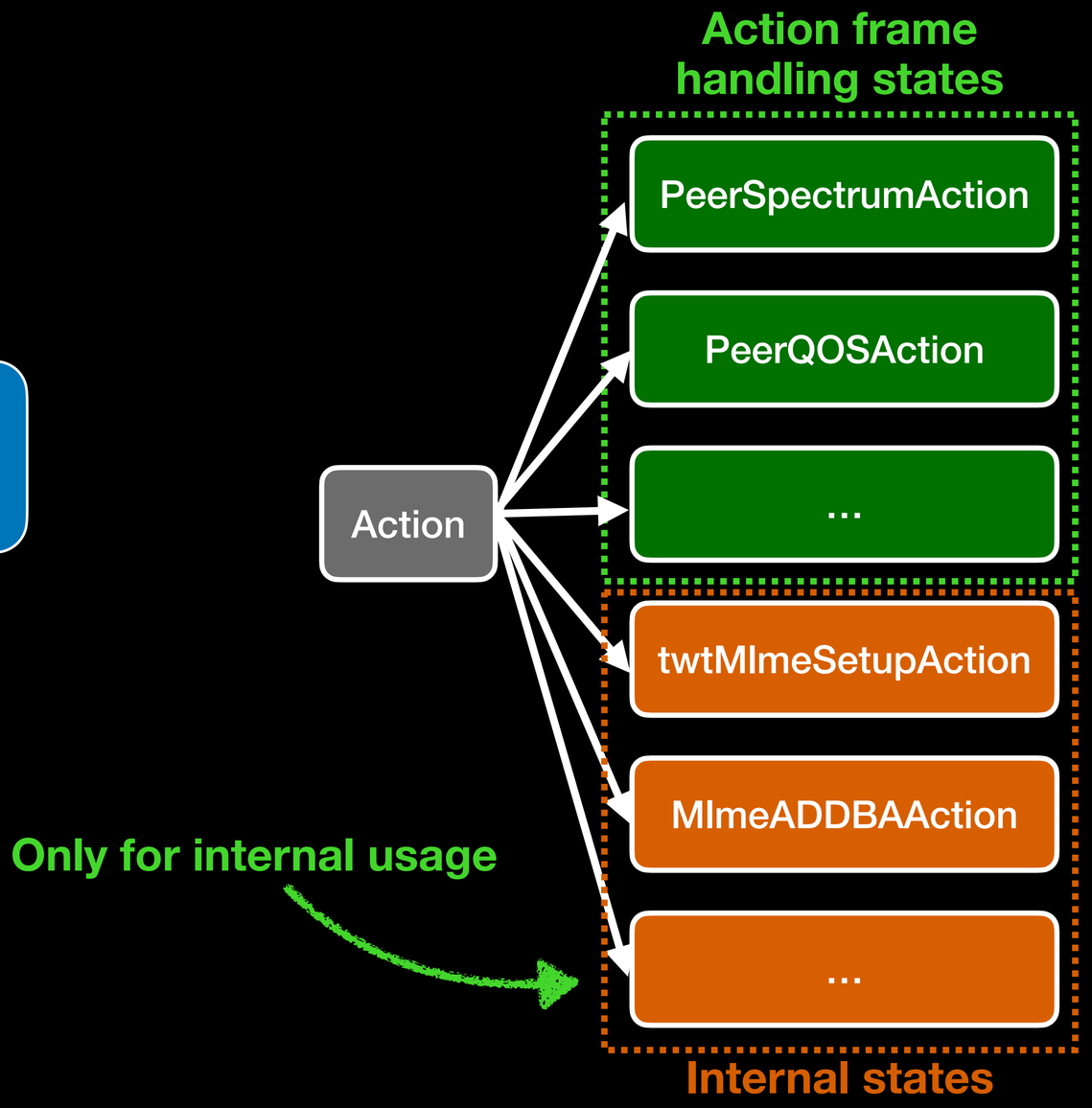
CVE-2025-20711



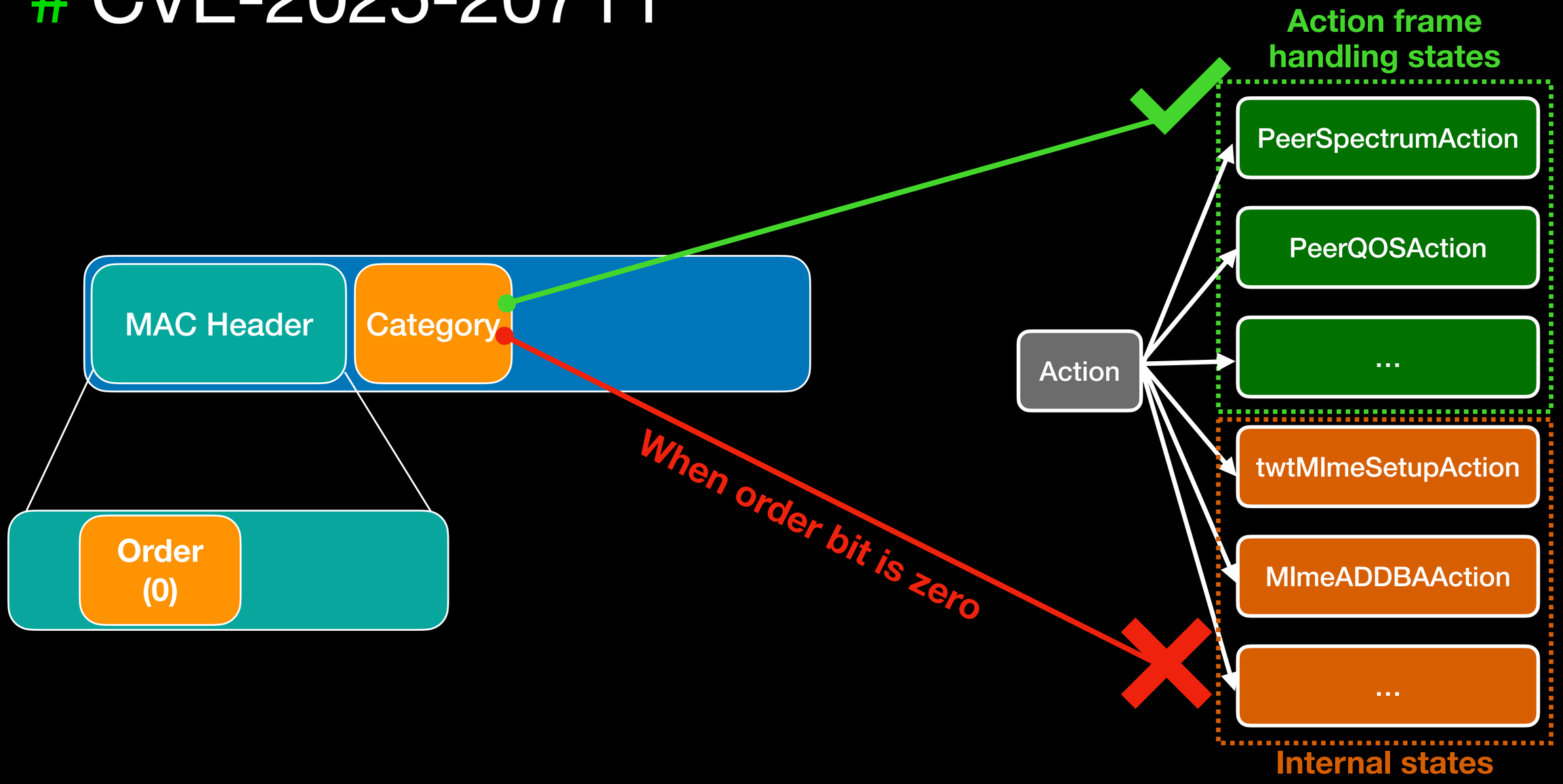


CVE-2025-20711

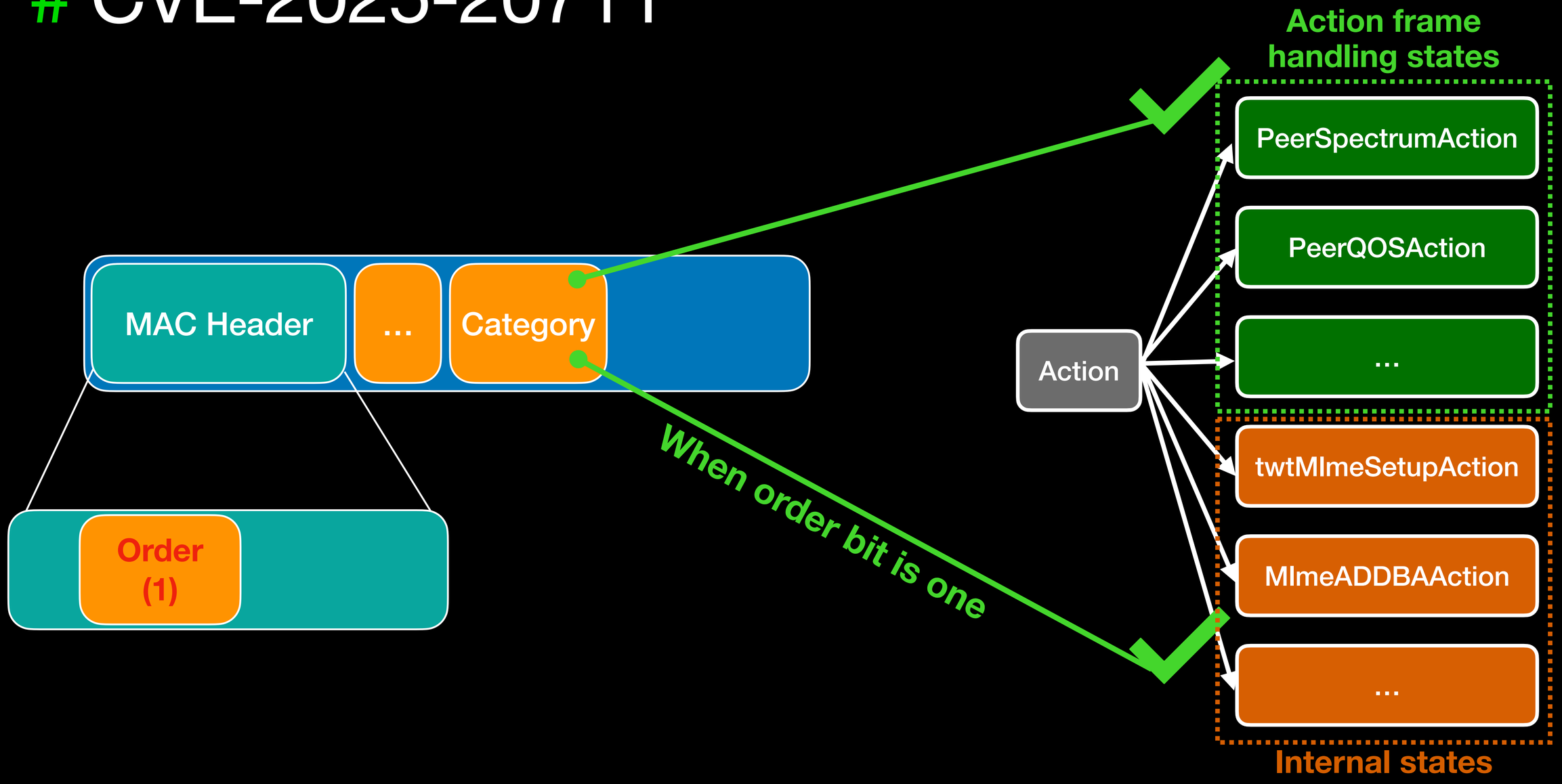




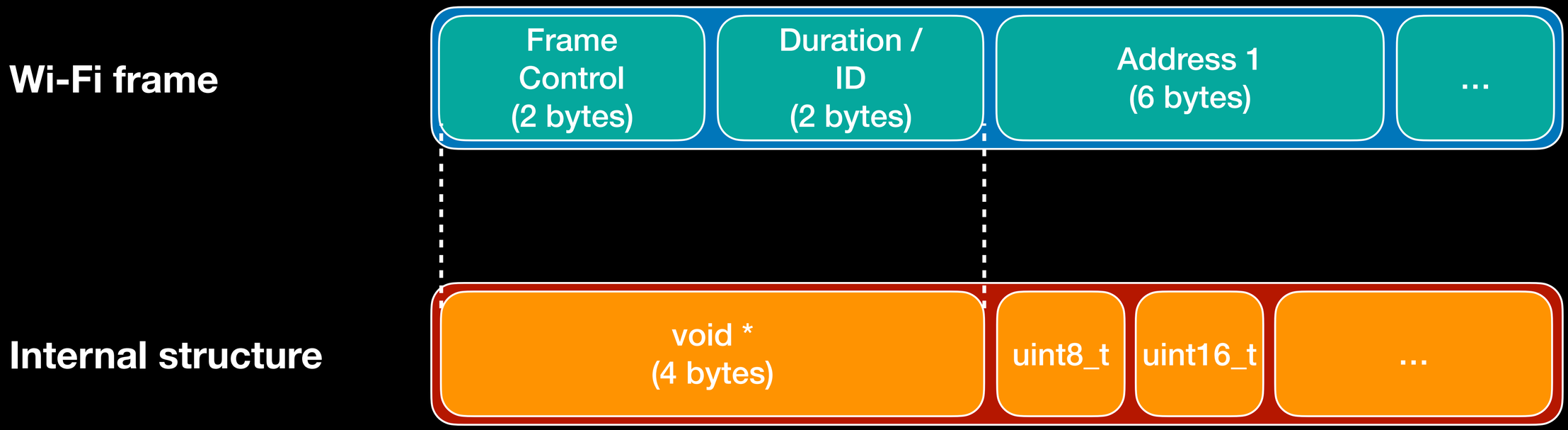
CVE-2025-20711



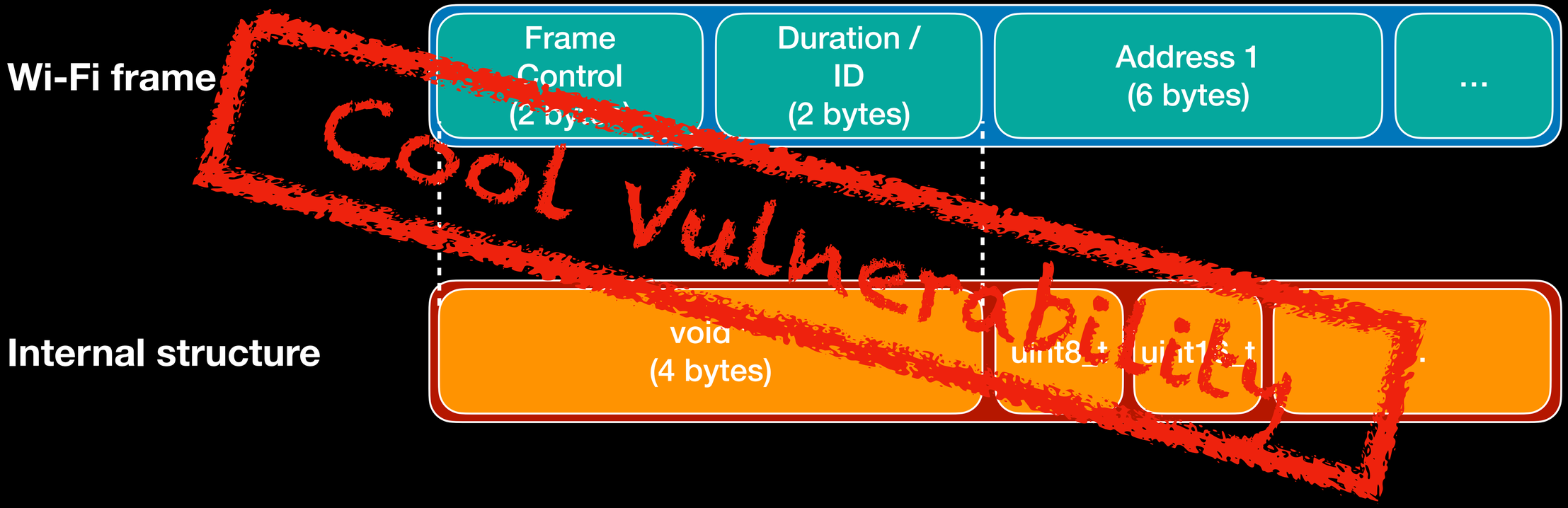
CVE-2025-20711



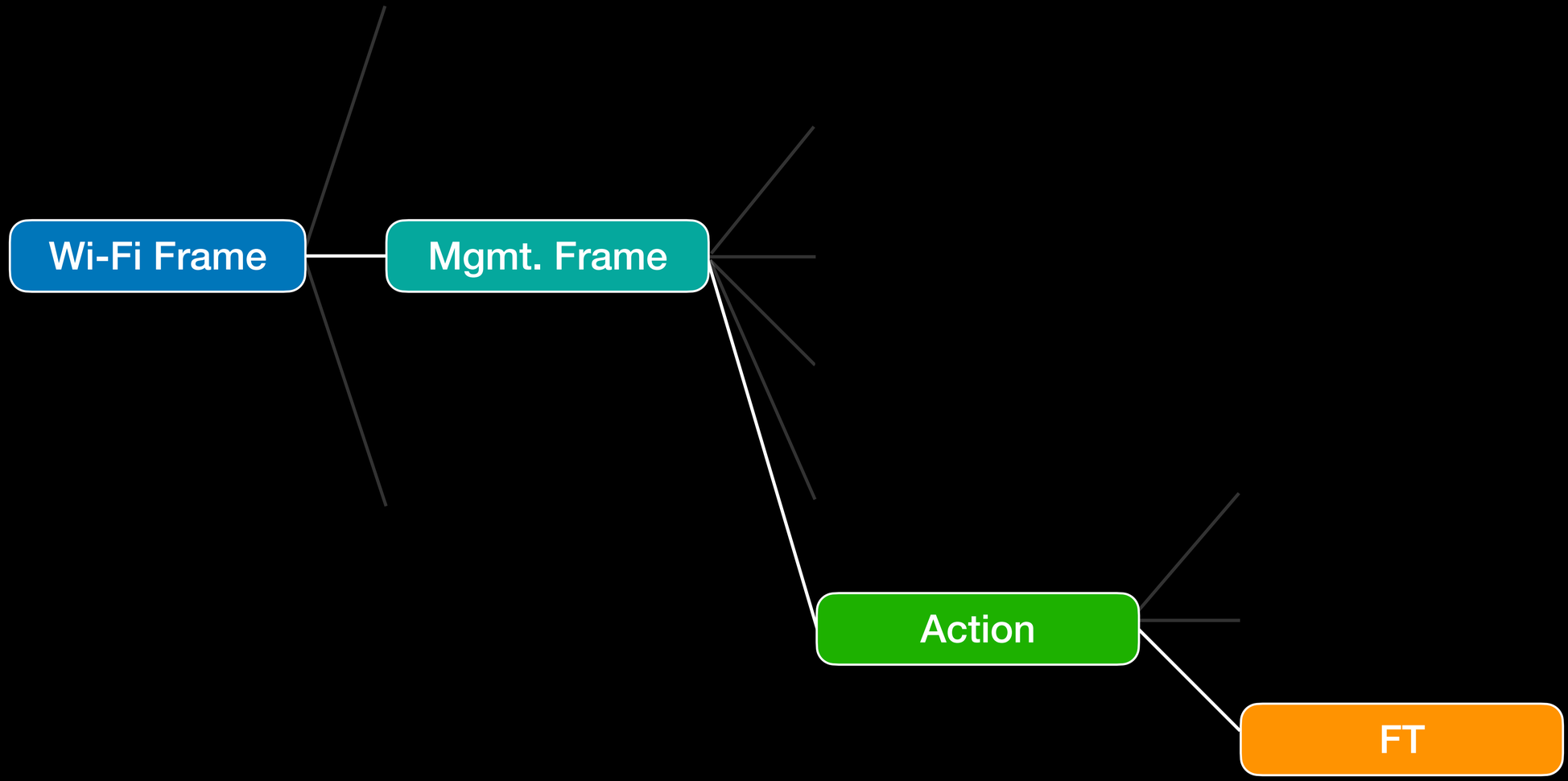
CVE-2025-20711



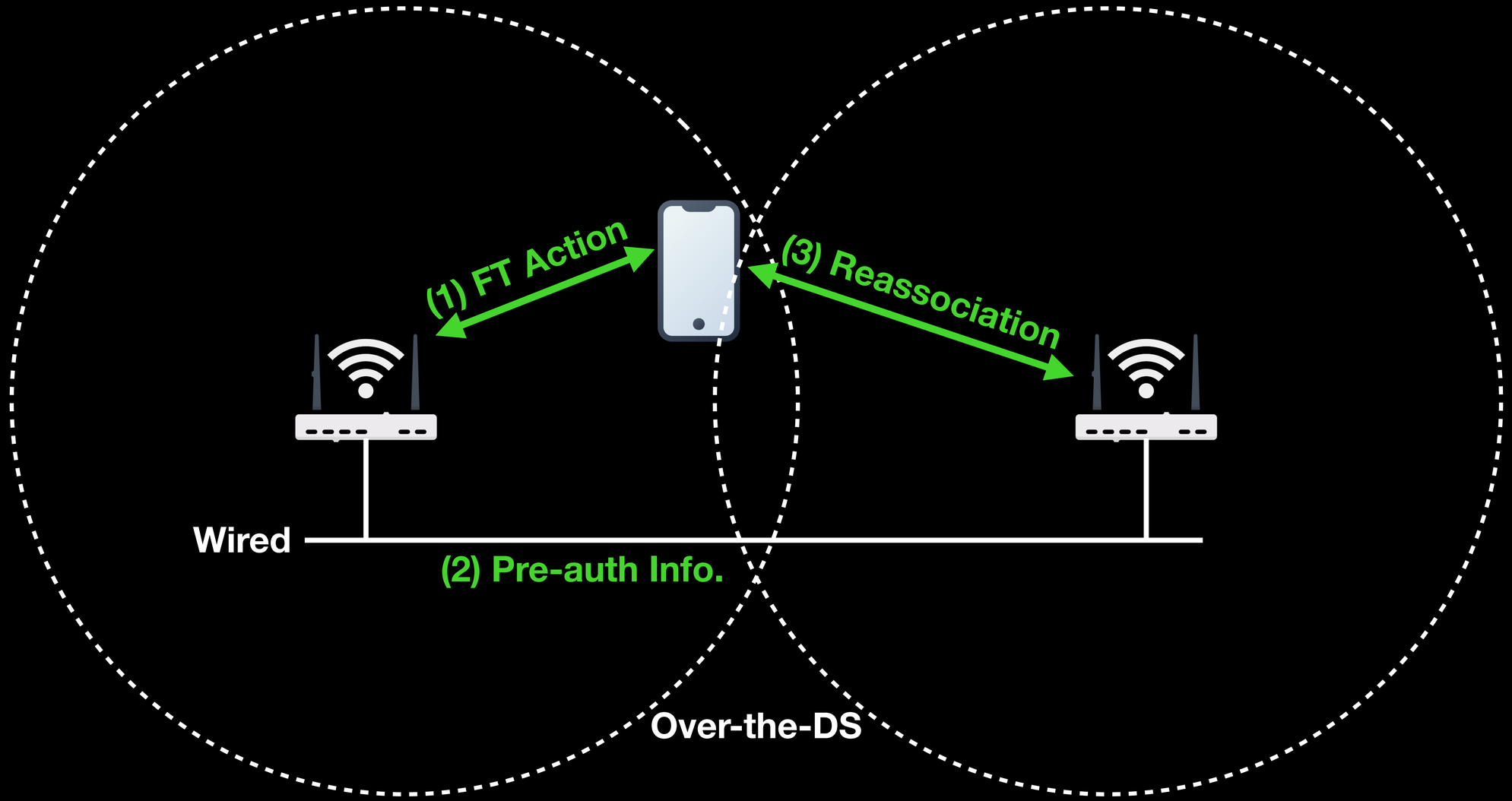
CVE-2025-20711



CVE-2025-20686



Fast BSS Transition (FT)

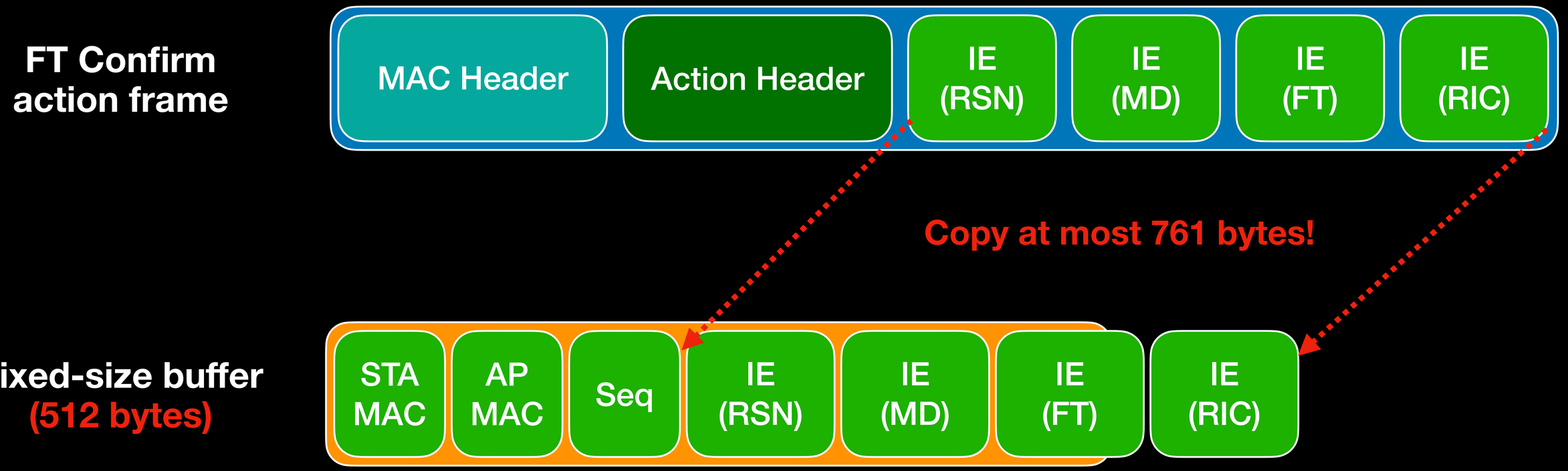


CVE-2025-20686

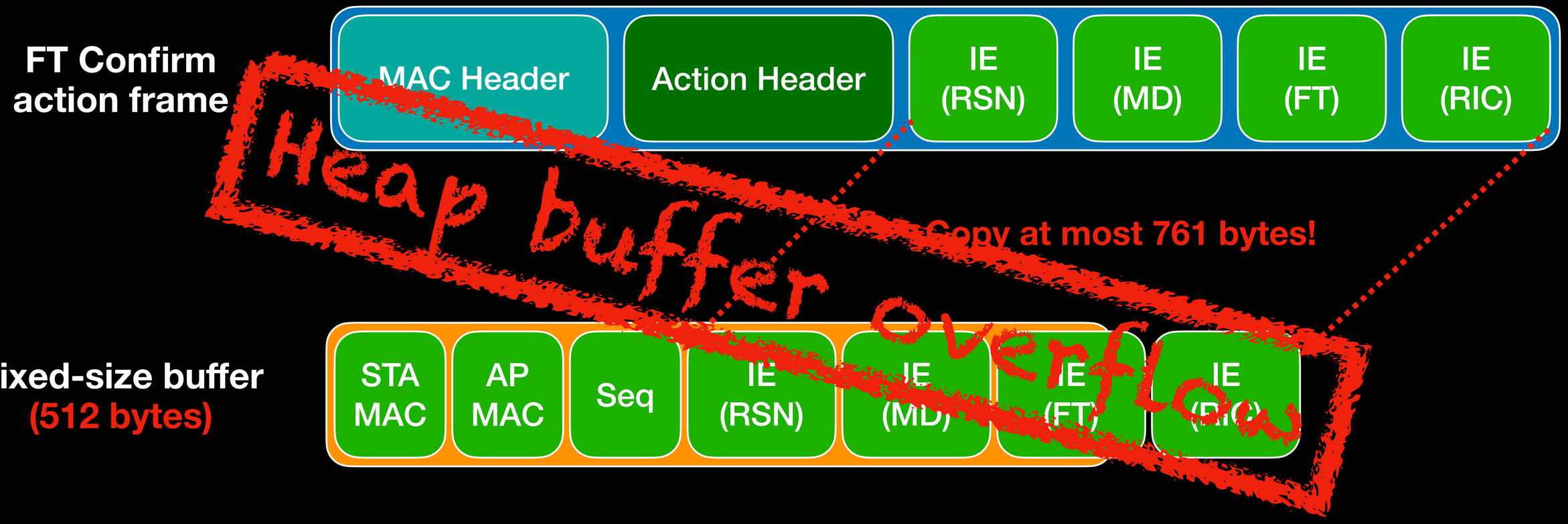
FT Confirm action frame

Fixed-size buffer (512 bytes)

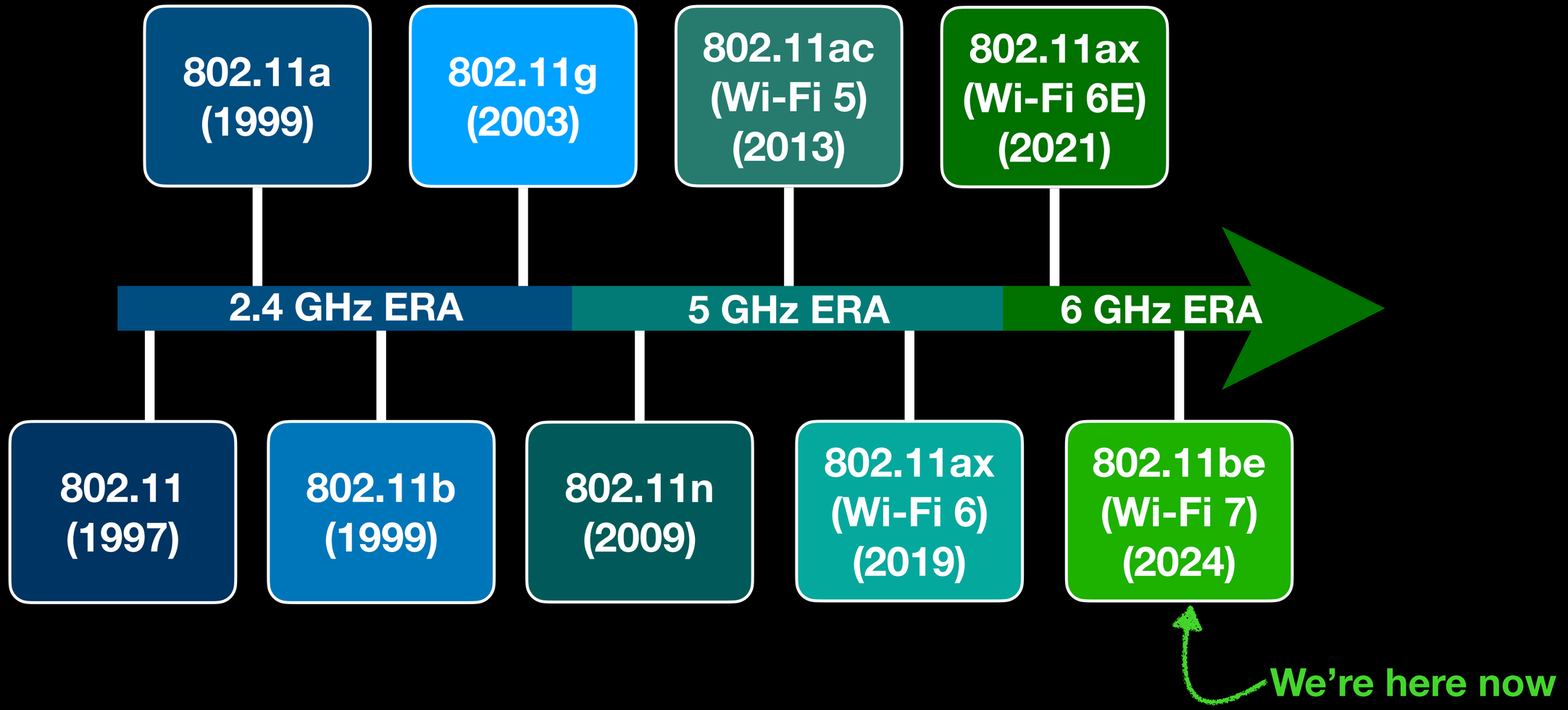
CVE-2025-20686



CVE-2025-20686

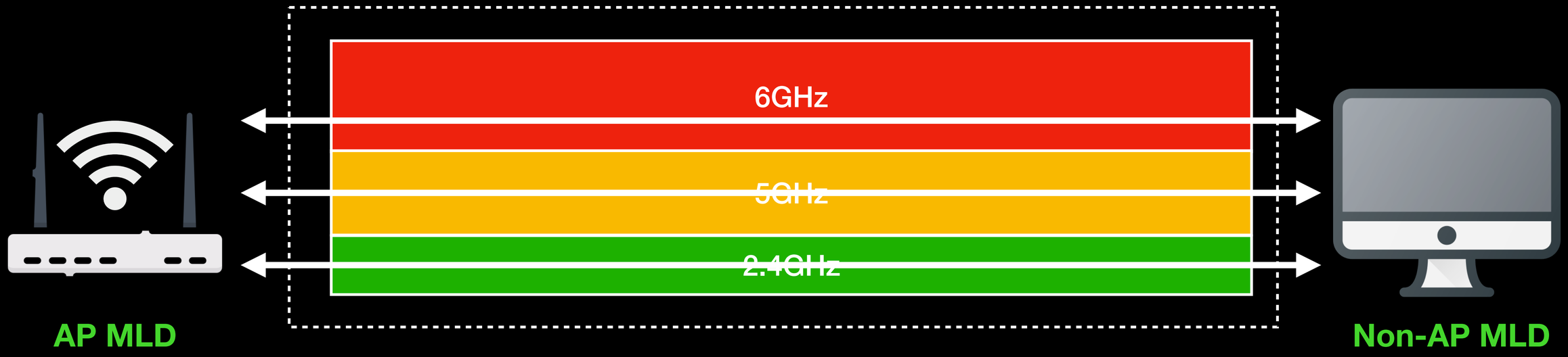


Wi-Fi 7


802.11a (1999)
802.11g (2003)
802.11ac (Wi-Fi 5) (2013)
802.11ax (Wi-Fi 6E) (2021)
2.4 GHz ERA
5 GHz ERA
6 GHz ERA
802.11 (1997)
802.11b (1999)
802.11n (2009)
802.11ax (Wi-Fi 6) (2019)
802.11be (Wi-Fi 7) (2024)
We're here now

Wi-Fi 7 - Multi-link Operation

- Multi-Link Operation, abbreviated as MLO
- It enables devices to simultaneously send and receive data across different frequency bands and channels



Wi-Fi 7 - Multi-link Element

- Multi-Link Element, abbreviated as MLE
- It is used to carry parameters related to MLO

Multi-link Element

Wi-Fi 7 - Multi-link Element

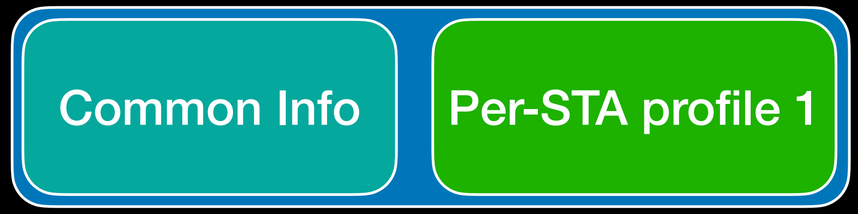
- Multi-Link Element, abbreviated as MLE
- It is used to carry parameters related to MLO
- Per-STA profile is used to carry information of each link

Multi-link Element

Per-STA profile

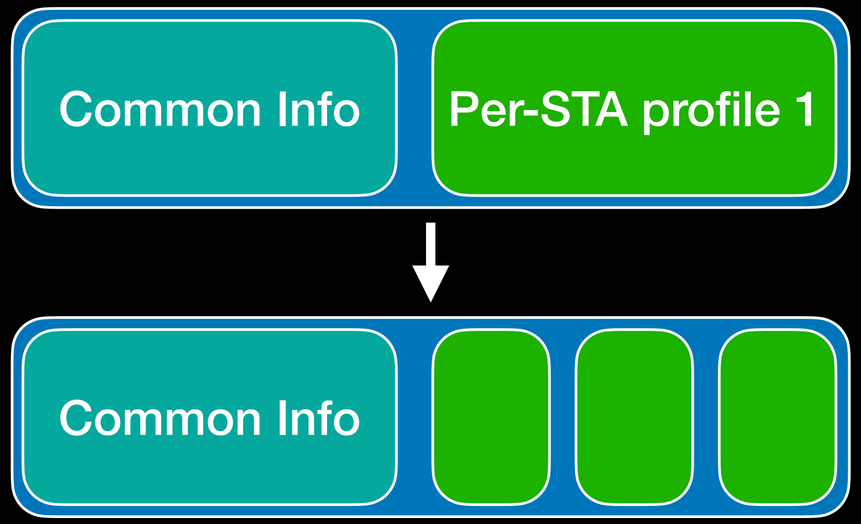
Fragmentation

- Both MLE and Per-STA profile can be fragmented



Per-STA Profile Fragmentation

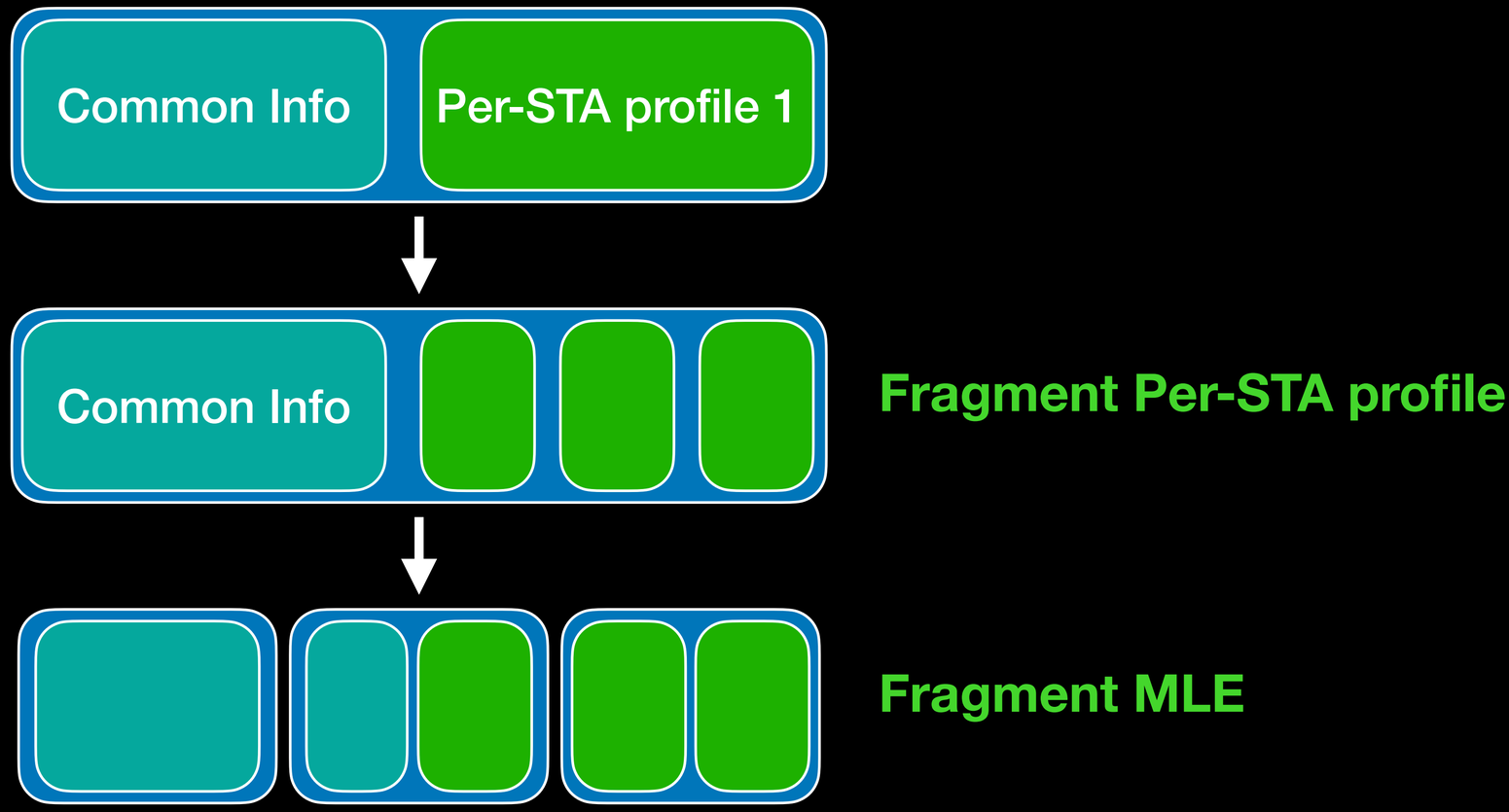
- Both MLE and Per-STA profile can be fragmented



Fragment Per-STA profile

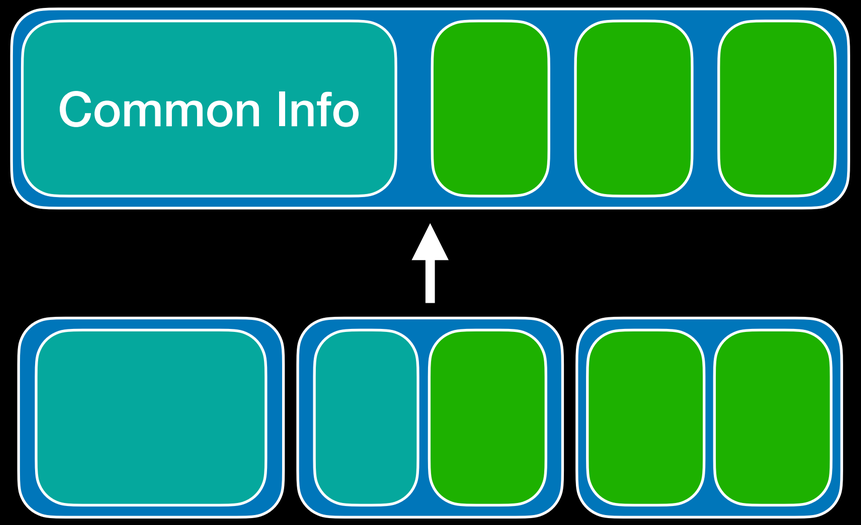
MLE Fragmentation

- Both MLE and Per-STA profile can be fragmented



CVE-2025-20712

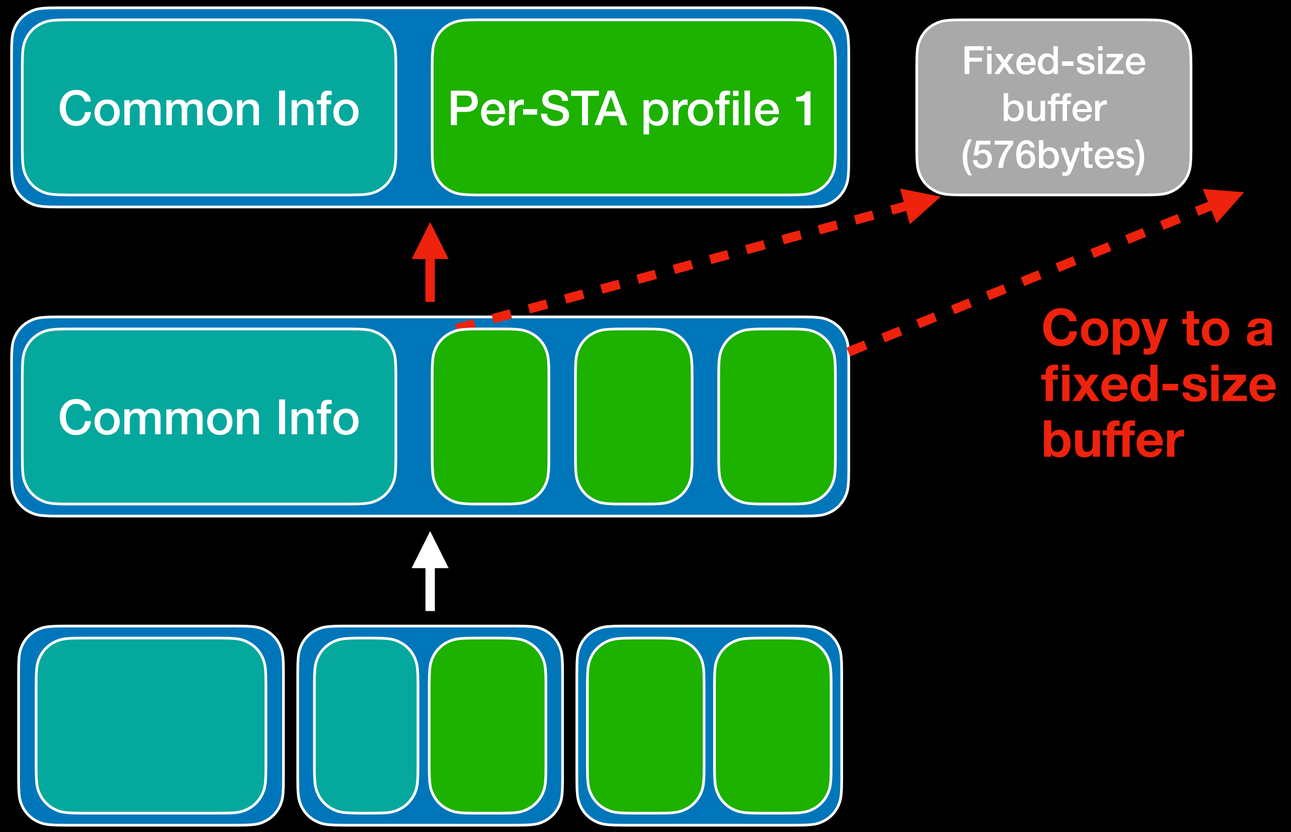
Defragment MLE



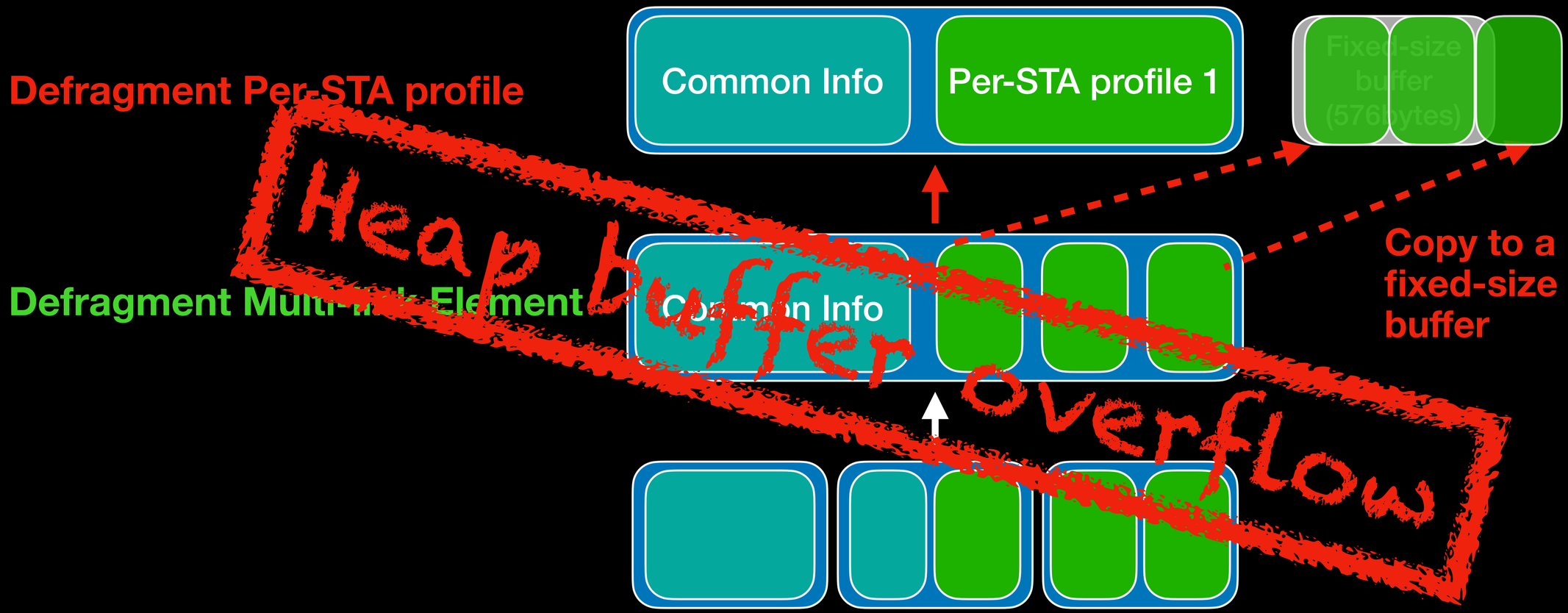
CVE-2025-20712

Defragment Per-STA profile

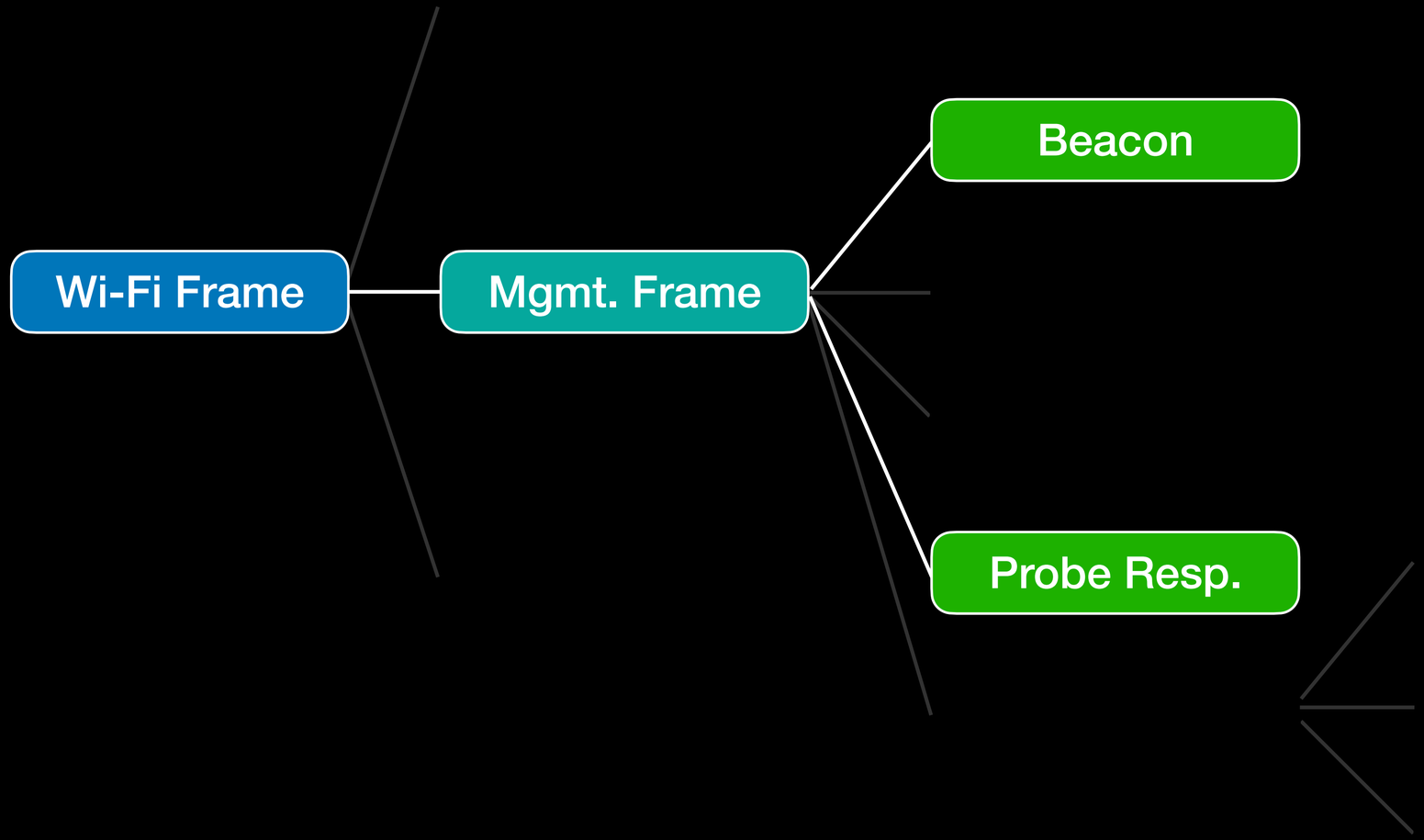
Defragment Multi-link Element



CVE-2025-20712

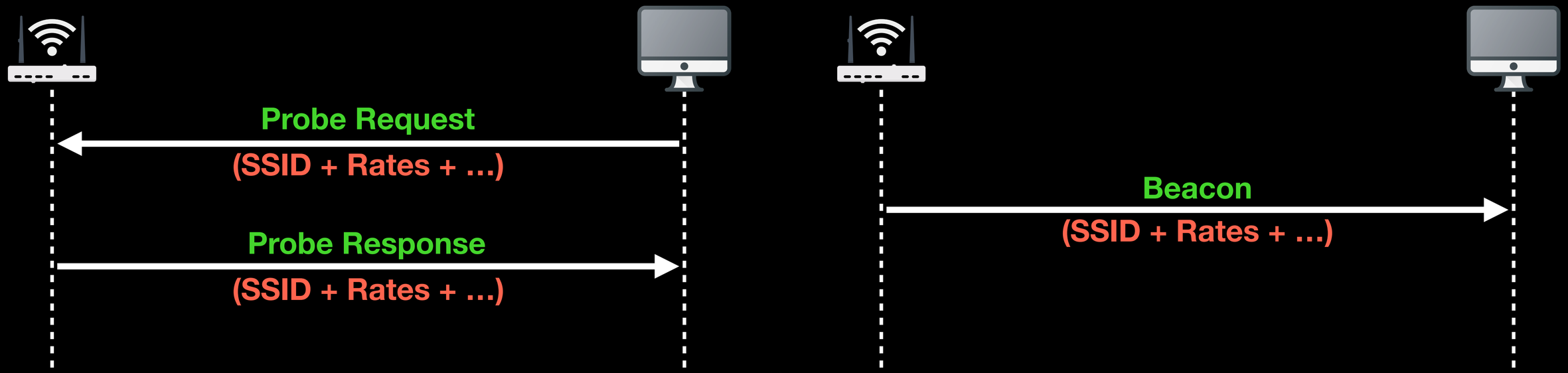


CVE-2025-20686



Wi-Fi Discovery

- Active Scanning
  - Wi-Fi device send probe frame to identify the presence of specific SSID
- Passive Scanning
  - Wi-Fi AP periodically broadcast beacon frames to announce their presence



CVE-2025-20685

```
int PeerBeaconAndProbeRspSanity(...){
    while(...){
        // Check if ie_buf can store ie
        switch(ie->type){
            case ...:
                ...
            case 244:
                // Copy ie into ie_buf
            case 221:
                // Check if wsc can store ie
                // Copy ie into wsc
        }
    }
    // Copy wsc into ie_buf
}
```

CVE-2025-20685

ie_buf: 1024 bytes

```
int PeerBeaconAndProbeRspSanity(...){
    while(...){
        // Check if ie_buf can store ie
        switch(...){
            case ...:
                ...
            case 244:
                // Copy ie into ie_buf
            case 221:
                // Check if wsc can store ie
                // Copy ie into wsc
        }
    }
    // Copy wsc into ie_buf
}
```

CVE-2025-20685

ie_buf: 1024 bytes
wsc: 512 bytes

```
int PeerBeaconAndProbeRspSanity(...){
    while(...){
        // Check if ie_buf can store ie
        switch(ie->type){
case 244:
    // Copy ie into ie_buf
case 221:
    // Check if wsc can store ie
    // Copy ie into wsc
        }
    // Copy wsc into ie_buf
}
```

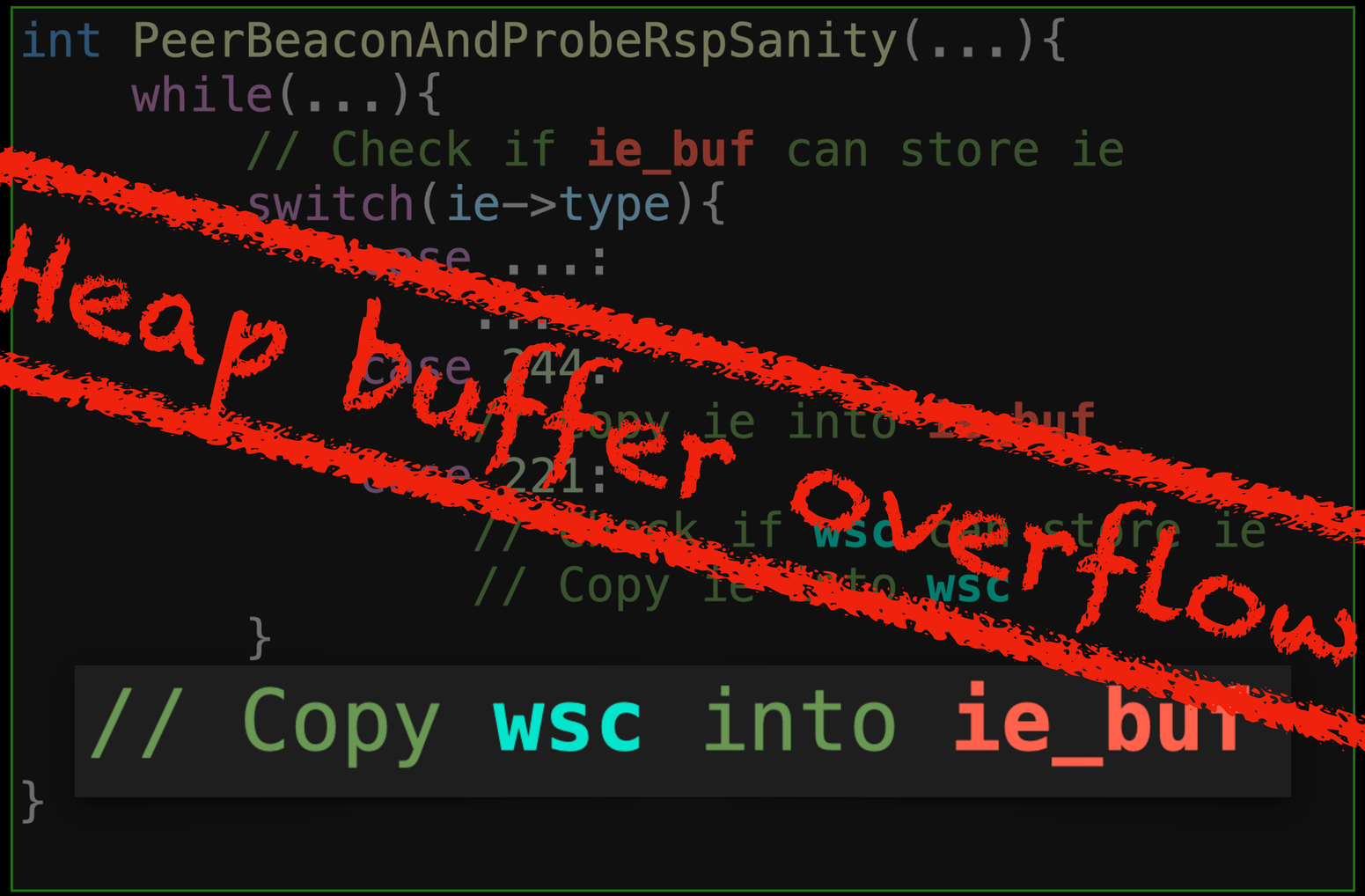
CVE-2025-20685

ie_buf: **1024 bytes**
wsc: **512 bytes**

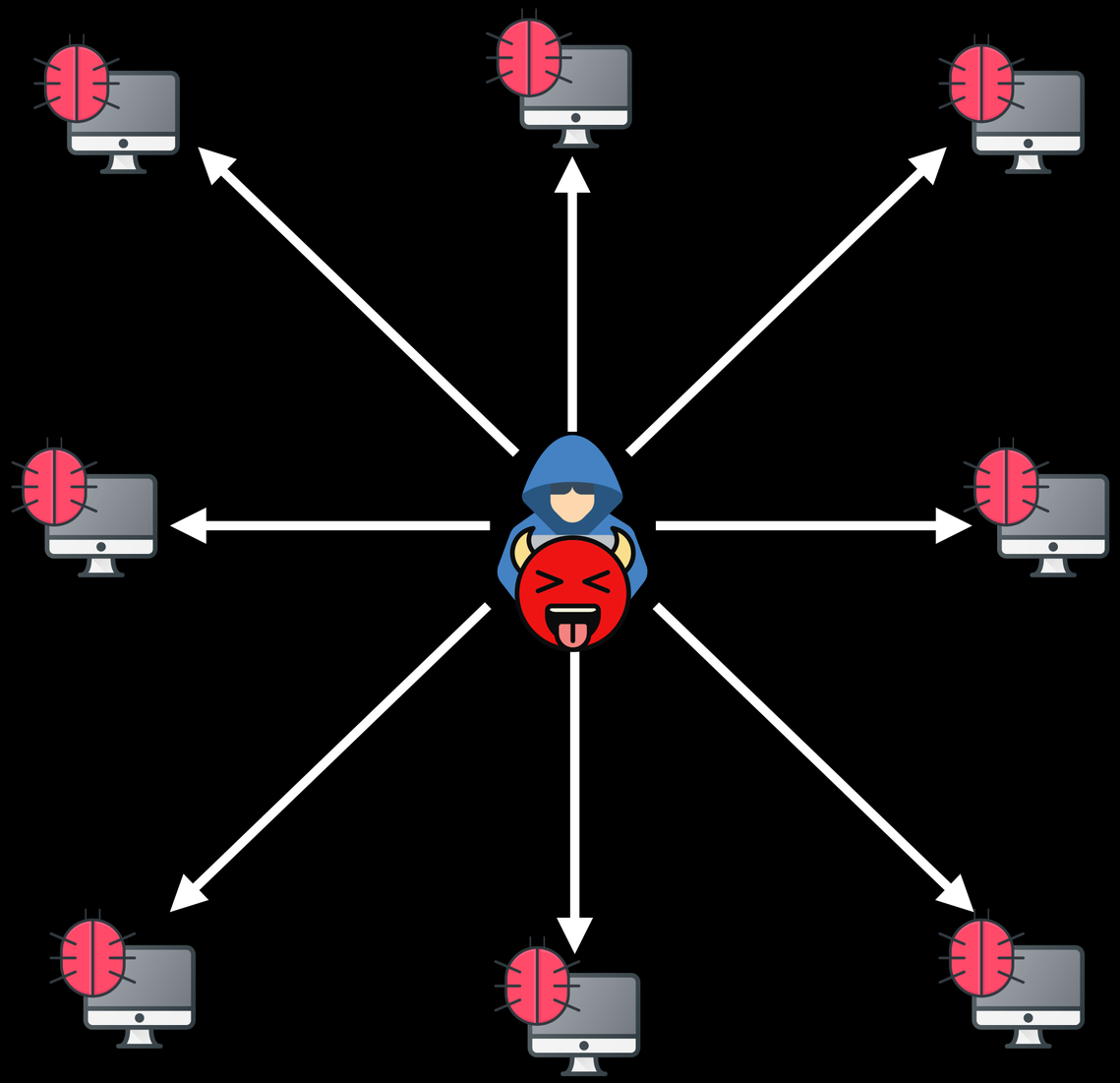
```
int PeerBeaconAndProbeRspSanity(...){
    while(...){
        // Check if ie_buf can store ie
        switch(ie->type){
            case ...:
                ...
            case 244:
                // Copy ie into ie_buf
            case 221:
                // Check if wsc can store ie
                // Copy ie into wsc
        }
    // Copy wsc into ie_buf
}
```

CVE-2025-20685

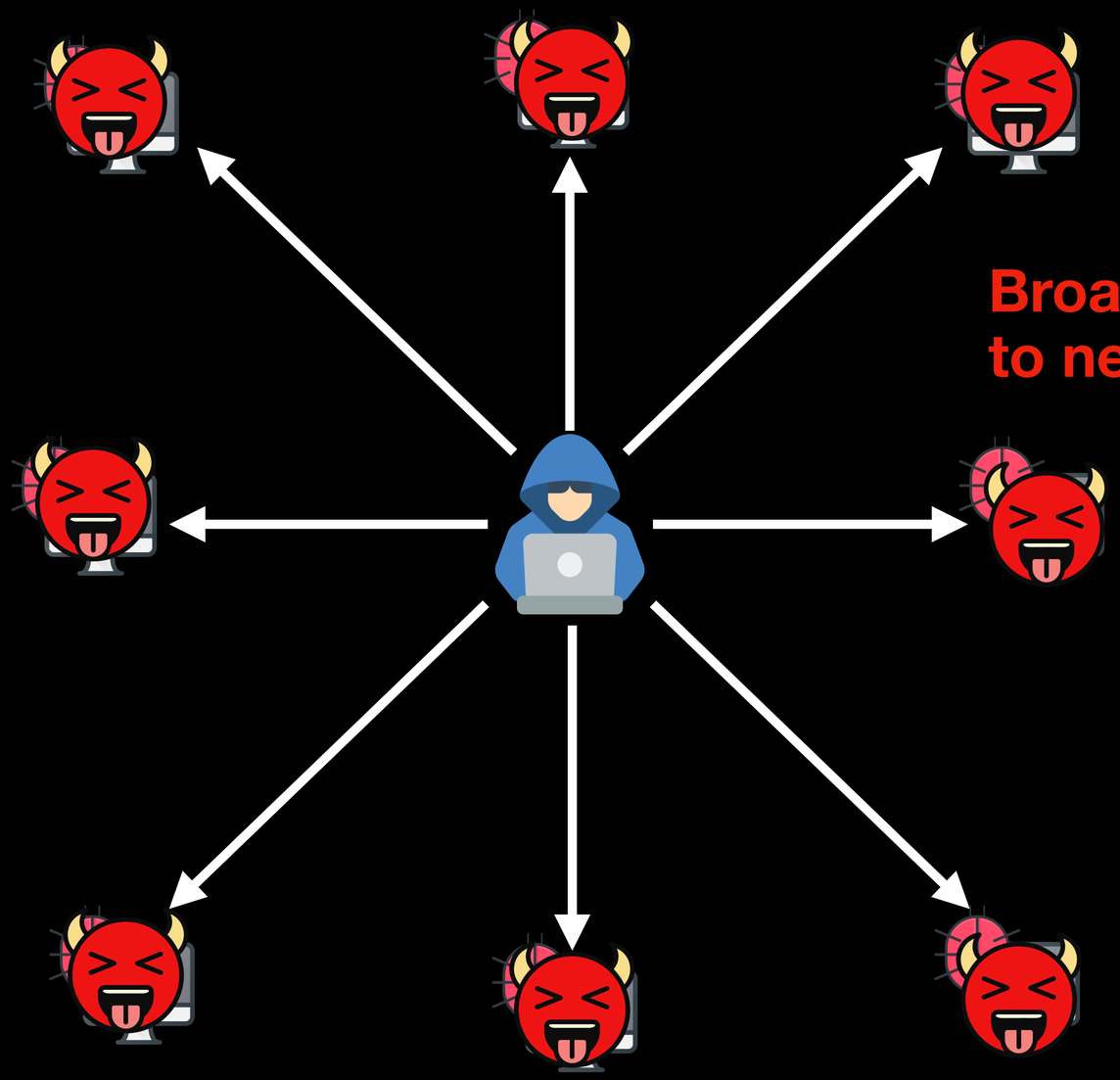
ie_buf: 1024 bytes
wsc: 512 bytes



CVE-2025-20685



CVE-2025-20685



Broadcast malicious frames to nearby devices

Internal Network Party

DEVCORE

Case Study - CVE-2025-20685

- What we have?
  - Heap-based Buffer Overflow (kmalloc-1k)
  - Overflow 508 bytes
- Kernel Protections
  - 🟢 Stack Canary
  - ❌ KASLR
  - ❌ Freelist Hardening
  - ❌ KCFI

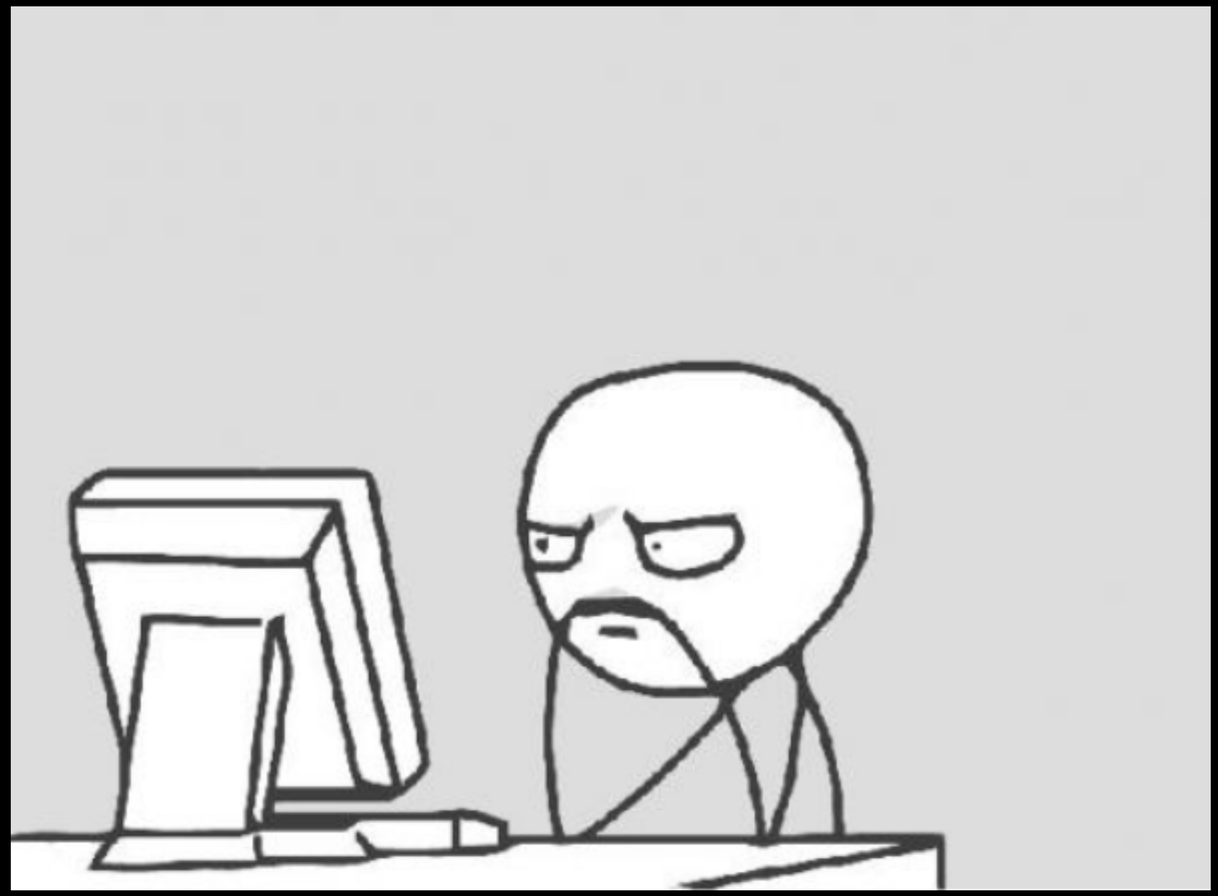
How to Debug?

- No GDB, but crash dump is enough

```
[   76.248093] CPU: 1 PID: 2875 Comm: RtmpMlmeTask Tainted: P                  5.4.171 #0
[   76.255991] Hardware name: MediaTek MT7981 RFB (DT)
[   76.260855] pstate: 20000005 (nzCv daif -PAN -UAO)
[   76.265637] pc : __kmalloc+0x84/0x208
[   76.269287] lr : __kmalloc+0x40/0x208
[   76.272935] sp : ffffffc0125c3c30
[   76.276236] x29: ffffffc0125c3c30 x28: 0000000000000000
[   76.281533] x27: ffffff80074d7a38 x26: ffffffc012c815f8
[   76.286831] x25: ffffffc008b8ff00 x24: ffffffc012daabc8
[   76.292127] x23: ffffff8007b49cc0 x22: ffffffc012690000
[   76.297425] x21: 0000000000000a20 x20: 00006f666e697265
[   76.302722] x19: ffffff800f003800 x18: 0000000000000000
[   76.308019] x17: 0000000000000000 x16: 0000000000000000
[   76.313315] x15: 0000000000000000 x14: 0000000000000000
[   76.318612] x13: 0000000000000000 x12: 0000000000000000
[   76.323909] x11: 0000000000000000 x10: 00000000000007e0
[   76.329205] x9 : 0000000000000000 x8 : ffffffc008eec2b8
[   76.334502] x7 : 0000000000000000 x6 : 000000000000003f
[   76.339799] x5 : 0000000000000040 x4 : ffffff800fcb7780
[   76.345096] x3 : ffffffbfff325000 x2 : ffffff800fb5fb80
[   76.350393] x1 : 0000000000002645 x0 : 0000000000000000
[   76.355691]
[   76.355691] PC: 0xffffffc0101c63ac:
[   76.360640] 63ac  f140081f 910003fd a90153f3 54000c88 a9025bf5 2a0103f5 97ff39e7 aa0003f3
[   76.368806] 63cc  f100401f 54000ce9 f0003401 b9499834 0a1402b4 2a1403e1 97ff39fd 35000a40
[   76.376970] 63ec  d503201f b4000a13 f9400260 910003e2 d538d081 91002000 f8616801 d538d083
[   76.385135] 640c  f9400260 8b030002 f9400844 f8636814 f100009f fa401a84 54000aa0 b9402260
[   76.393299] 642c  f8606a83 d538d082 91000424 f9400260 8b020000 f9800011 c87f1406 ca1400c6
[   76.401464] 644c  ca0100a5 aa0500c5 b5000065 c8261003 35ffff46 b5fffca5 b9402260 f8a06860
[   76.409628] 646c  d503201f d503201f d34822b5 35000515 d503201f a9425bf5 aa1403e0 a94153f3
[   76.417793] 648c  a8c37bfd d65f03c0 37b00074 b9400a60 36d7faa0 aa1303e0 94002a9f aa0003f3
[   76.425958]
[   76.425958] LR: 0xffffffc0101c6368:
[   76.430906] 6368  d4210000 b0002ba0 aa1903e2 913a2000 d2800001 9411067d f900b660 b5fffae0
[   76.439071] 6388  aa1903e0 12800174 9411045a 35fff597 17ffff8f aa1903e0 94110456 17ffff8c
[   76.447235] 63a8  a9bd7bfd f140081f 910003fd a90153f3 54000c88 a9025bf5 2a0103f5 97ff39e7
[   76.455400] 63c8  aa0003f3 f100401f 54000ce9 f0003401 b9499834 0a1402b4 2a1403e1 97ff39fd
[   76.463564] 63e8  35000a40 d503201f b4000a13 f9400260 910003e2 d538d081 91002000 f8616801
[   76.471729] 6408  d538d083 f9400260 8b030002 f9400844 f8636814 f100009f fa401a84 54000aa0
[   76.479894] 6428  b9402260 f8606a83 d538d082 91000424 f9400260 8b020000 f9800011 c87f1406
[   76.488059] 6448  ca1400c6 ca0100a5 aa0500c5 b5000065 c8261003 35ffff46 b5fffca5 b9402260
[   76.496225]
[   76.496225] SP: 0xffffffc0125c3bb0:
[   76.501173] 3bb0  12daabc8 ffffffc0 08b8ff00 ffffffc0 12c815f8 ffffffc0 074d7a38 ffffff80
[   76.509337] 3bd0  00000000 00000000 125c3c30 ffffffc0 101c63e8 ffffffc0 125c3c30 ffffffc0
[   76.517502] 3bf0  101c642c ffffffc0 20000005 00000000 12690000 ffffffc0 00000004 ff000000
[   76.525667] 3c10  ffffffff ffffffff 100d4e2c ffffffc0 125c3c30 ffffffc0 101c642c ffffffc0
[   76.533831] 3c30  125c3c60 ffffffc0 08d07764 ffffffc0 125c3cf0 ffffffc0 128b7340 ffffffc0
[   76.541996] 3c50  12d113e0 ffffffc0 12690000 ffffffc0 125c3ca0 ffffffc0 08b5a1e8 ffffffc0
[   76.550161] 3c70  12690000 ffffffc0 100d112c ffffffc0 0f0b2cc0 ffffff80 0fb5c980 ffffff80
[   76.558326] 3c90  0f0b2cd8 ffffff80 0fb5c9b8 ffffff80 125c3d30 ffffffc0 08b8f6cc ffffffc0
```

Find Useful Structures

- No Idea of what we can control…



Find Useful Structures

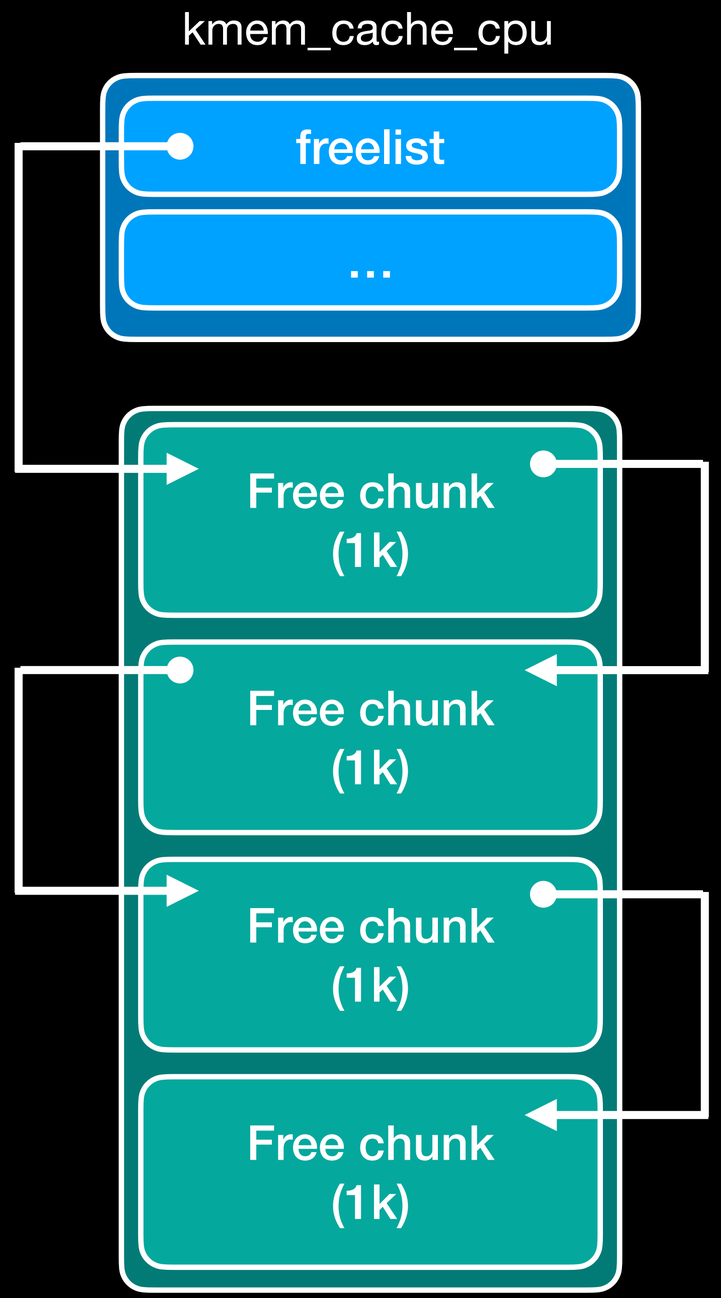
- No Idea of what we can control…
- I decided to keep crashing the devices and:
  - Collect the stack traces
  - Identify potential interesting traces

Stack Traces

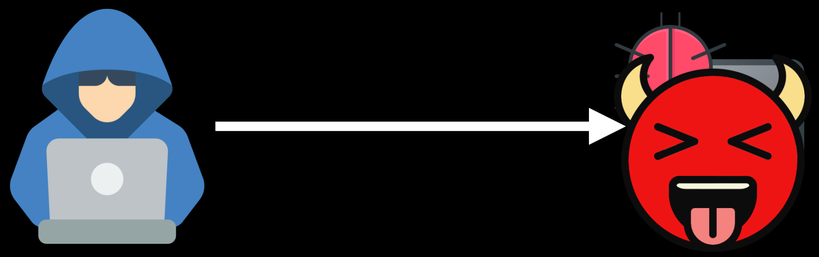
```
[ 2081.708470] bc10  ffffffff ffffffff 100d4e2c ffffffc0 1264bc30 ffffffc0 101c642c ffffffc0
[ 2081.716635] bc30  1264bc60 ffffffc0 08d02ad4 ffffffc0 1264bcf0 ffffffc0 128febd0 ffffffc0
[ 2081.724799] bc50  12d2d5c0 ffffffc0 126c1000 ffffffc0 1264bca0 ffffffc0 08b55340 ffffffc0
[ 2081.732963] bc70  126c1000 ffffffc0 10638450 ffffffc0 03f9a100 ffffff80 1264bd88 ffffffc0
[ 2081.741127] bc90  0fb5c980 ffffff80 0fb5c940 ffffff80 1264bd30 ffffffc0 08b8a824 ffffffc0
[ 2081.749293] Call trace:
__kmalloc+0x84/0x208
os_alloc_mem+0x1c/0x38 [mt_wifi]
wlan_operate_get_vht_ldpc+0x1e20/0x26a8 [mt_wifi]
StateMachinePerformAction+0x6c/0x80 [mt_wifi]
MlmeHandler+0x6fc/0x948 [mt_wifi]
[ 2081.823806] SMP: stopping secondary CPUs
[ 2081.827718] Kernel Offset: disabled
[ 2081.831194] CPU features: 0x0002,20002008
[ 2081.835188] Memory Limit: none
[ 2081.858882] Rebooting in 3 seconds..
```

Freelist Hijack

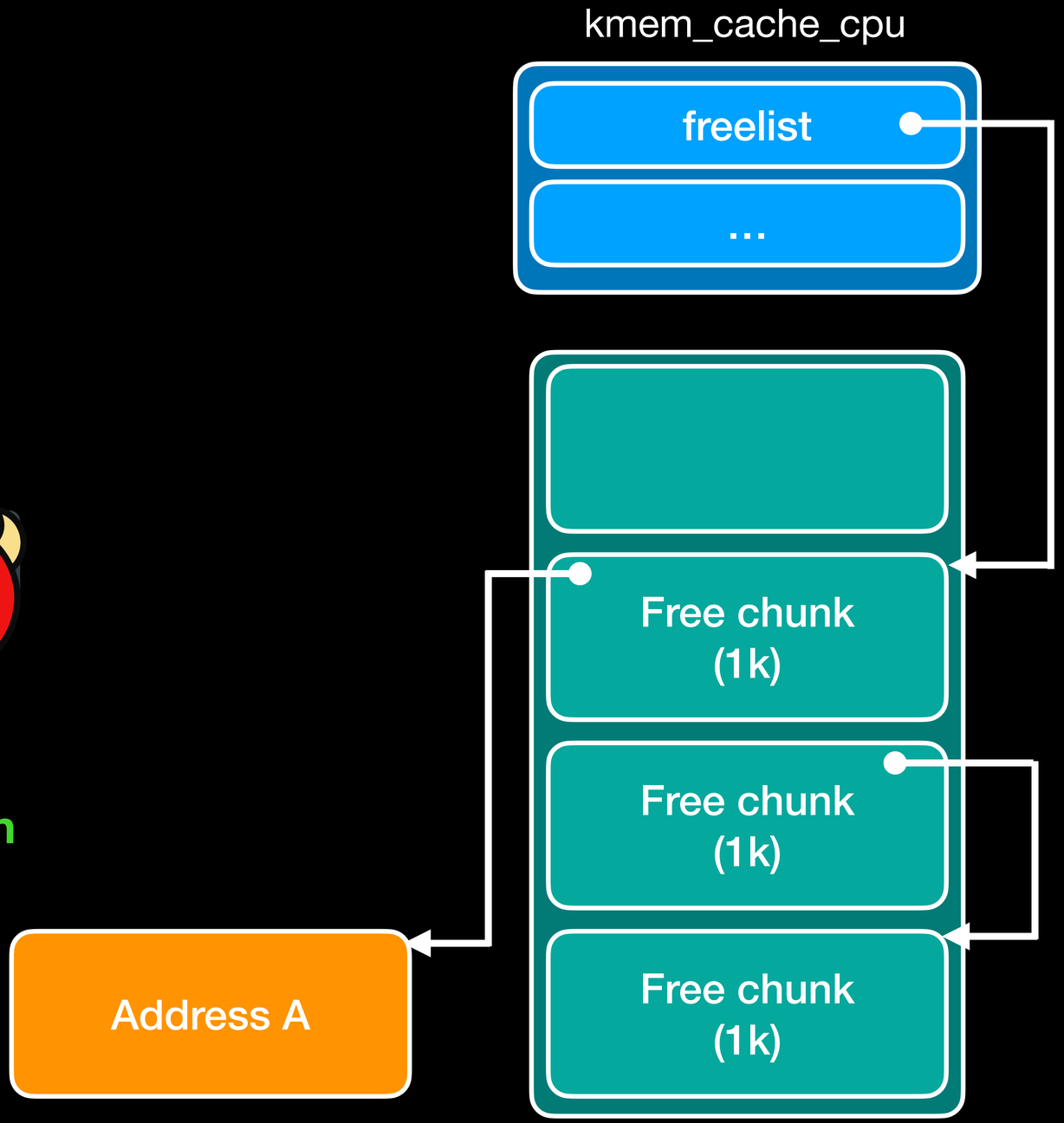
Attacker prepares first beacon to hijack freelist



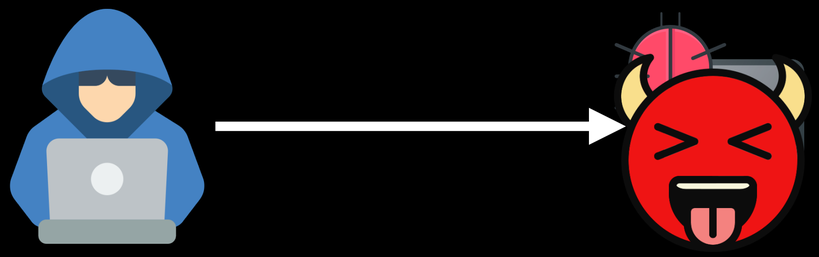
Freelist Hijack



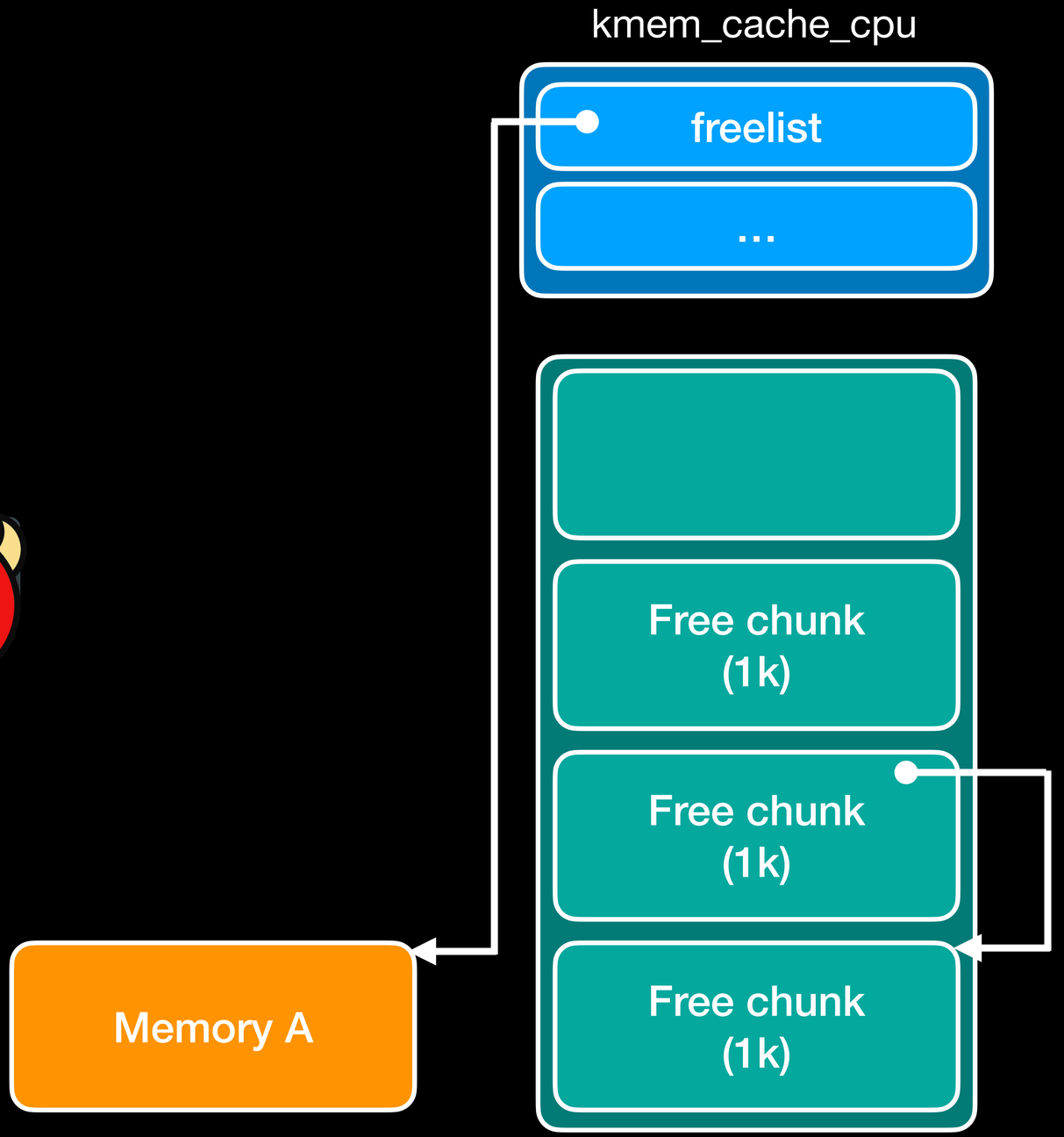
Attacker sends the first beacon



Freelist Hijack

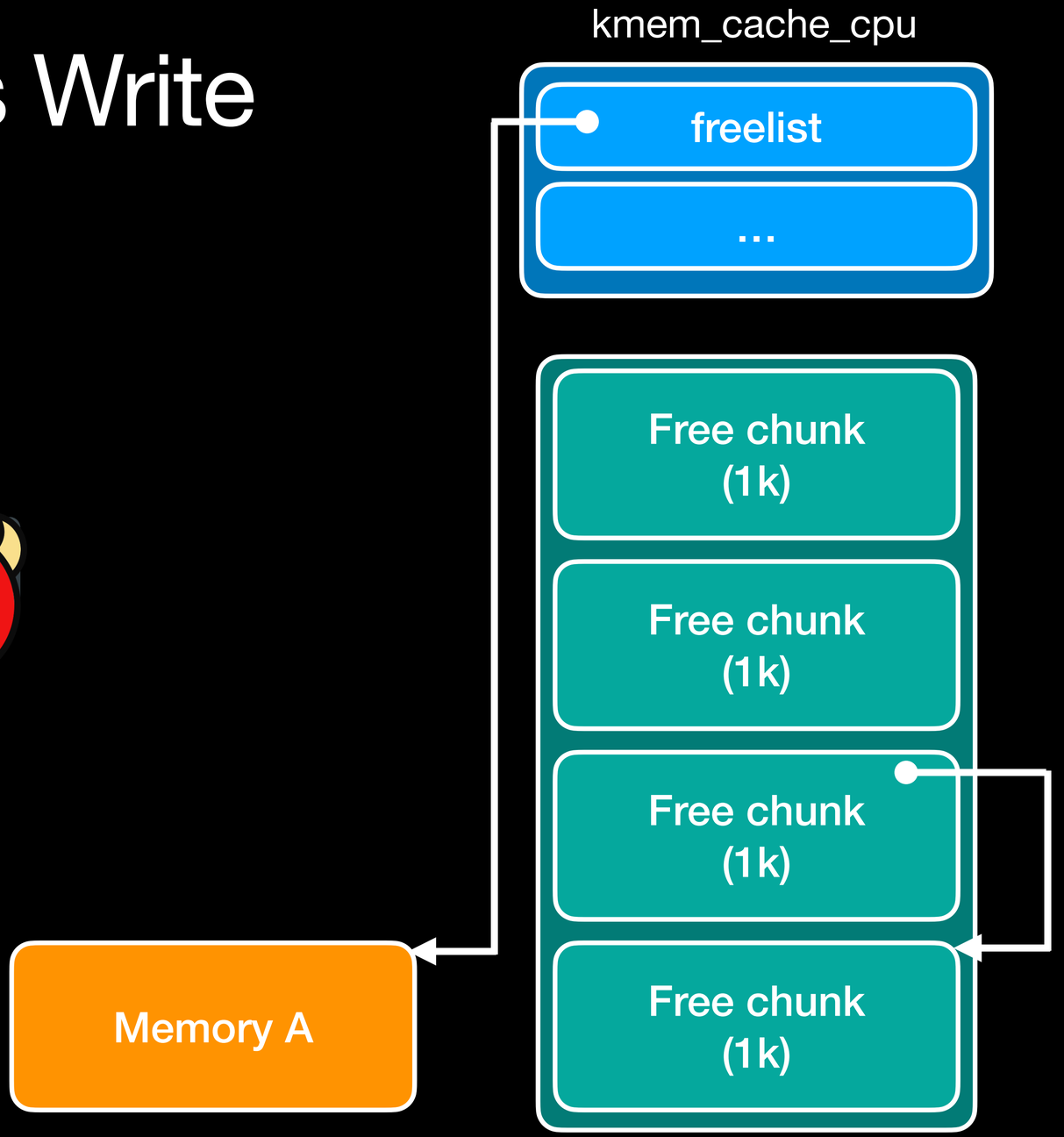


Someone Take one free chunk

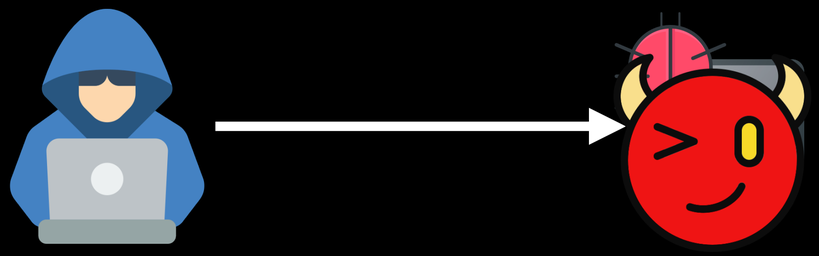


Arbitrary Address Write

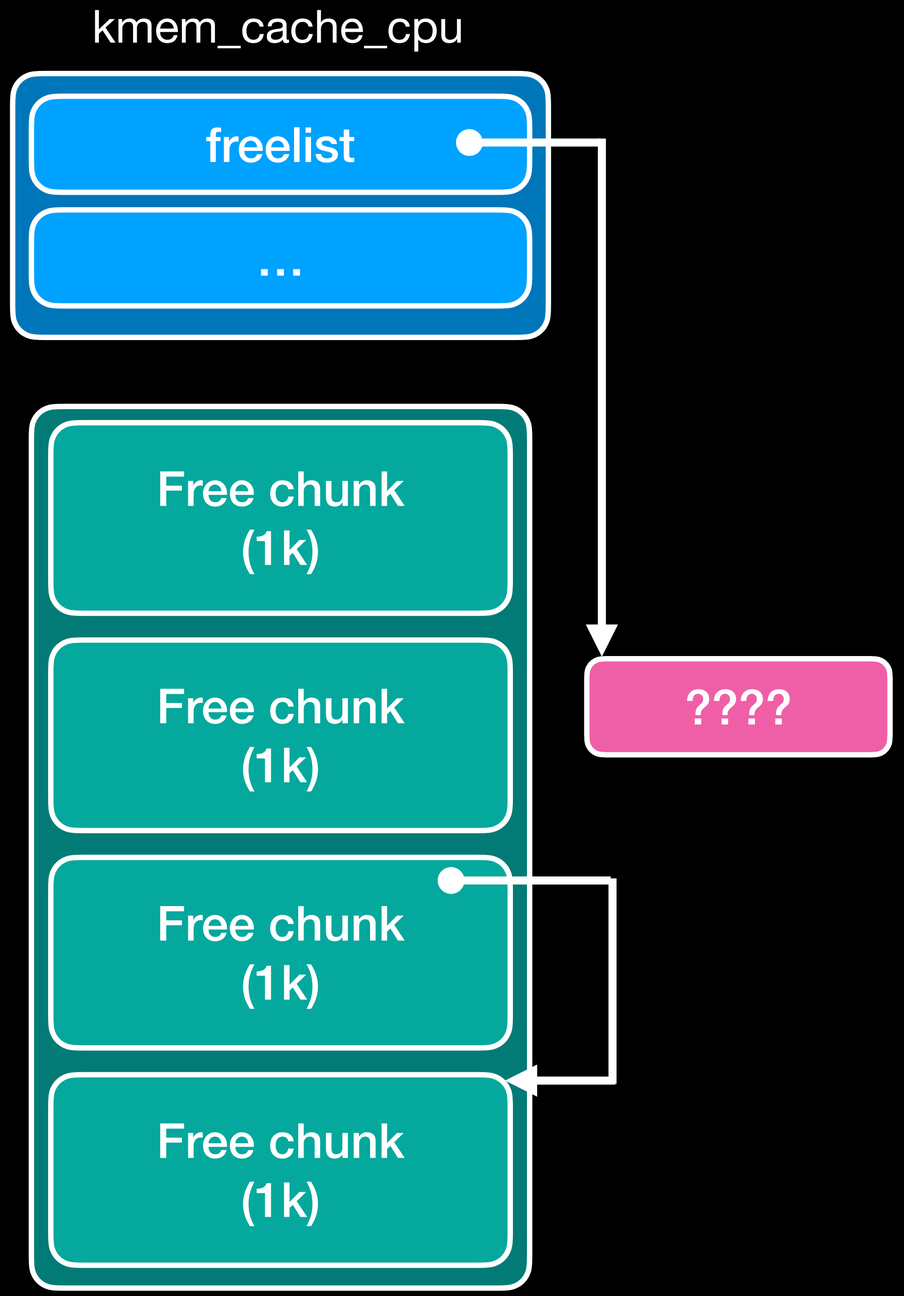
Attacker prepares second beacon to achieve AAW



Arbitrary Address Write



Attacker sends second beacon



Exploitation Plan

- We can achieve one-shot arbitrary address write
  - Place the shellcode in our payload
  - Overwrite the function pointer of Wi-Fi kernel module to cpu_switch_to

```
[    77.443850] Hardware name: MediaTek MT7981 RFB (DT)
[    77.448714] pstate: 00000005 (nzcv daif -PAN -UAO)
[    77.453494] pc : 0xaaaaaaaaaaaaaaaa
…
[    77.519602] x9 : 8a8a8a8a8a8a8a8a x8 : 9191919191919191
[    77.524898] x7 : 9090909090909090 x6 : 0000000000000000
[    77.530195] x5 : 0000000000000000 x4 : aaaaaaaaaaaaaaaa
[    77.535492] x3 : ffffffc012cad028 x2 : ffffff80038c6000
[    77.540789] x1 : ffffffc0128c5bd0 x0 : ffffffc012688000
```

```
LDP     X19, X20, [X8],#0x10
LDP     X21, X22, [X8],#0x10
LDP     X23, X24, [X8],#0x10
LDP     X25, X26, [X8],#0x10
LDP     X27, X28, [X8],#0x10
LDP     X29, X9,  [X8],#0x10
LDR     X30, [X8]
MOV     SP, X9
MSR     SP_EL0, X1
RET
```

We can control x8

cpu_switch_to

Exploitation Plan

- We can achieve one-shot arbitrary address write
  - Place the shellcode in our payload
  - Overwrite the function pointer of Wi-Fi kernel module to cpu_switch_to
- Execute three gadgets to
  - Make the shellcode executable
  - Return to the shellcode

Three gadgets that I used

```
mov x19, x1
mov x1, x22
  blr x23
    …

mov x0, x20
  blr x25
    …

  blr x21
ldp x21, x22, [sp, #0x20]
ldp x29, x30, [sp], #0x30
    ret
```

Limitations in Exploitation

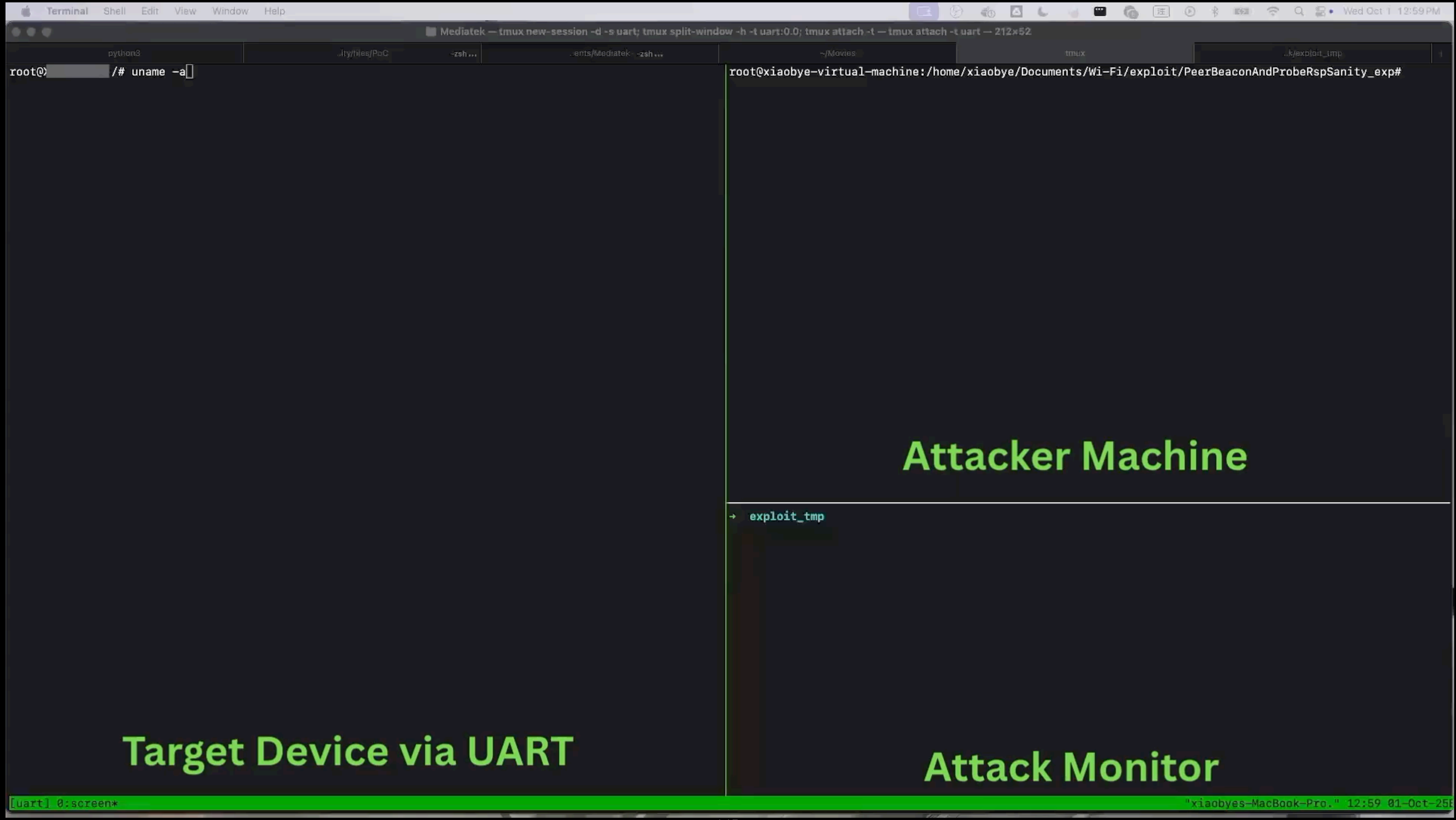
- Problems related to probability:
  - Kernel modules have partially randomized load addresses
  - The freelist order is not always as expected

Limitations in Exploitation

- Problems related to probability:
  - Kernel modules have partially randomized load addresses
  - The freelist order is not always as expected
- In some cases, kernel modules are compiled into kernel:
  - Exploitation requires less time

Demo

DEVCORE


Terminal Shell Edit View Window Help
Wed Oct 1 12:59 PM
Mediatek — tmux new-session -d -s uart; tmux split-window -h -t uart:0.0; tmux attach -t — tmux attach -t uart — 212×52
python3
-zsh...
-zsh...
~/Movies
tmux
root@ /# uname -a
root@xiaobye-virtual-machine:/home/xiaobye/Documents/Wi-Fi/exploit/PeerBeaconAndProbeRspSanity_exp#
Attacker Machine
→ exploit_tmp
Target Device via UART
Attack Monitor
[uart] 0:screen*
"xiaobyes-MacBook-Pro." 12:59 01-Oct-25

Conclusion

DEVCORE

Advices

- For device vendor
  - Enabling KASLR and freelist hardening increases the difficulty of remote heap exploitation
  - Continuously track chipset vendor updates and keep the SDK updated
- For chipset vendor
  - Share testing builds or hotfixes with researchers under NDA when necessary to validate patches

DEVCORE

Advices

- For researcher
  - There is a large attack surface in the Wi-Fi kernel module and vendor-specific protocols based on Wi-Fi
  - The wf_rom.axd in the MediaTek Wi-Fi SDK really speeds up reverse-engineering MCU firmware
- For user
  - Strong passwords alone aren’t enough. Limit Wi-Fi signal leakage

DEVCORE

Mediatek’s PSIRT

- Mediatek has a complete vulnerability reporting and disclosure process
- All CVEs and vulnerabilities mentioned in this slide deck have already been fixed
  - CVE-2025-20674 has already been fixed in the June 2025 MediaTek Security Bulletin
  - CVE-2025-20685 and CVE-2025-20686 have already been fixed in the July 2025 MediaTek Security Bulletin
  - CVE-2025-20711 and CVE-2025-20712 have already been fixed in the October 2025 MediaTek Security Bulletin
- Mediatek first discloses the patches to OEM partners, giving them two months to apply them

DEVCORE

End

Attribution

- https://www.flaticon.com/free-icons/wifi
  - Wifi signal icons created by DinosoftLabs - Flaticon
- https://www.flaticon.com/free-icons/computer
  - Computer icons created by Those Icons - Flaticon
- https://www.flaticon.com/free-icons/hacker
  - Hacker icons created by Smashicons - Flaticon
- https://www.flaticon.com/free-icons/malicious-app
  - Malicious app icons created by Hopstarter - Flaticon
- https://www.flaticon.com/free-icons/evil
  - Evil icons created by muhammad atho - Flaticon
- https://www.flaticon.com/free-icons/electronics
  - Electronics icons created by kumakamu - Flaticon

DEVCORE