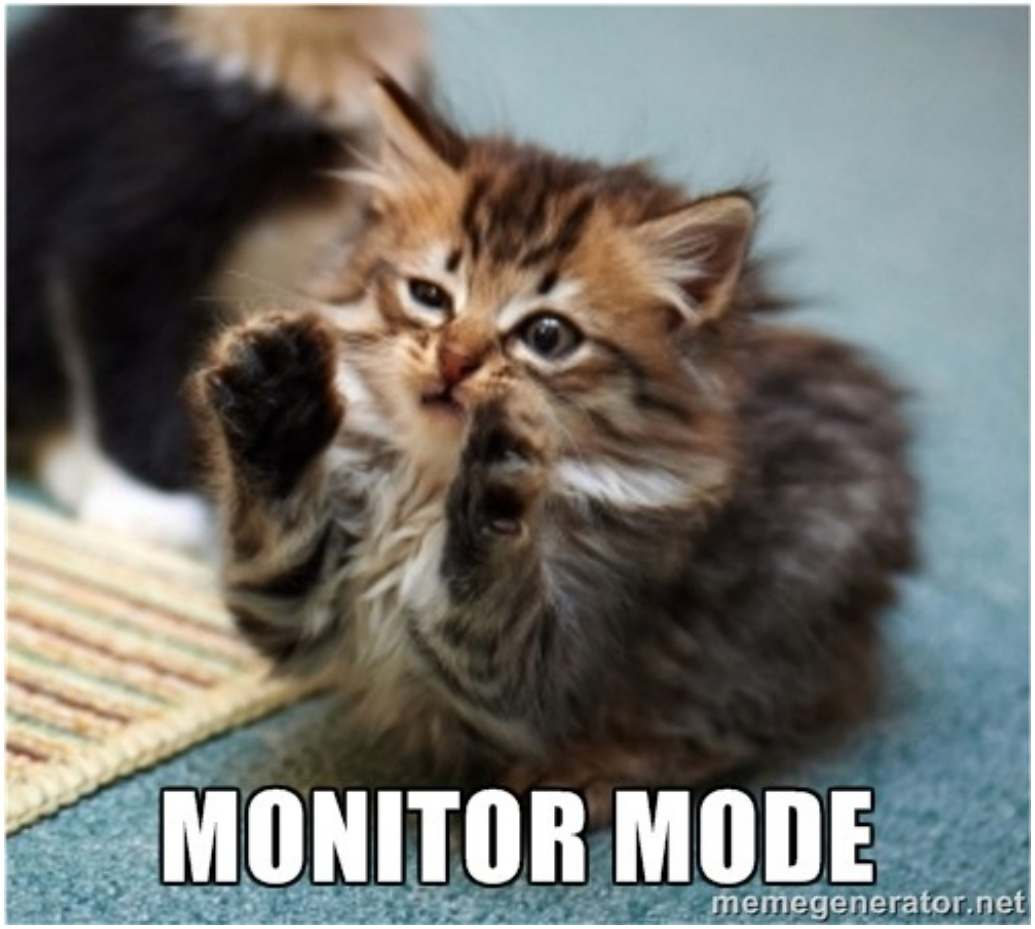
nexmon

MAKE WI-FI HACKING ON SMARTPHONES GREAT AGAIN!

Overview

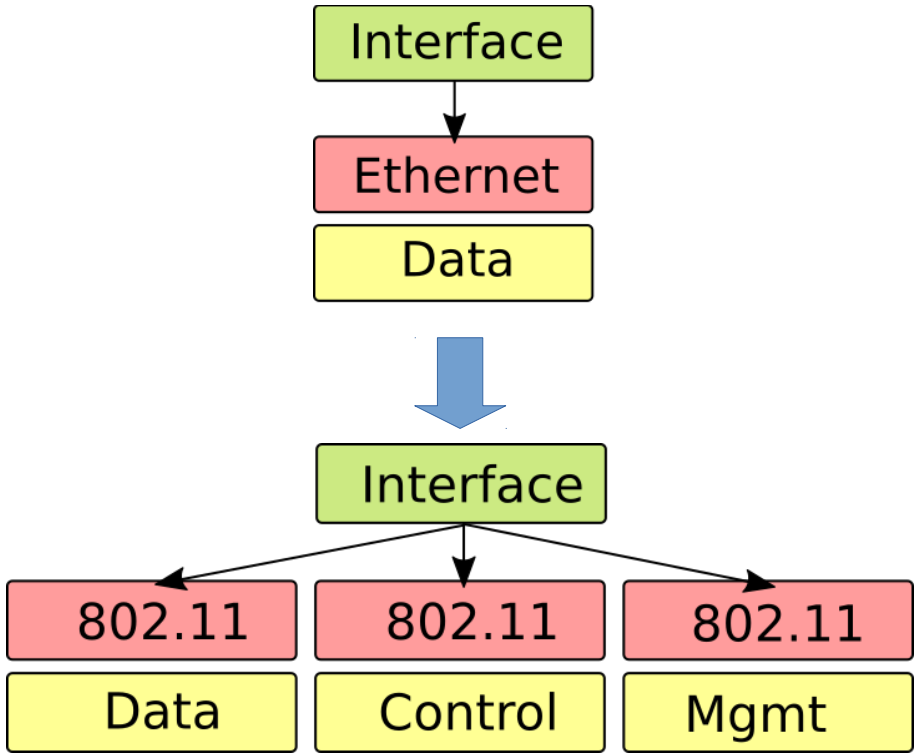
1. Monitor Mode
    1. Motivation
    2. Code Extraction and Examination
    3. Patching Framework
    4. Demo
2. Native Monitor Mode
3. Related Projects
4. Future Work

Motivation: Monitor Mode



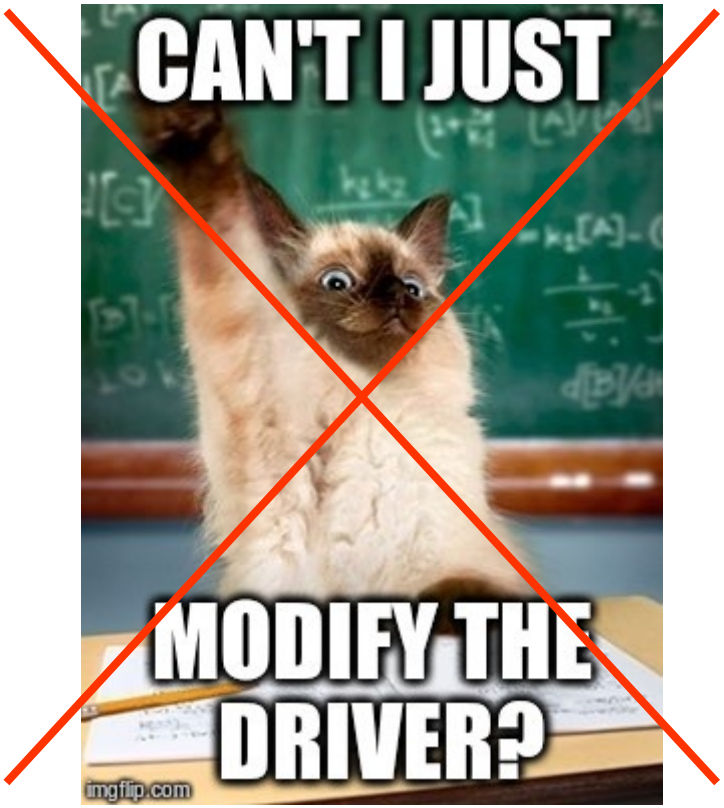
Motivation: Monitor Mode on Smartphones

- Receive **arbitrary frames** (incl. mgmt + ctl frames)
- Receive frames **form all stations** (promiscuous mode)
- **Inject** custom packets
- Run **legacy tools** like the aircrack-ng suite



Motivation:
Driver Modifications

- It's open source, right?



Motivation: Wi-Fi Chip Types

SoftMAC:
MAC Layer handled
in the **driver**

FullMAC:
MAC Layer handled
in the **Wi-Fi chip**

Motivation: FullMAC vs SoftMAC

- From the firmware point of view:

Full MAC  **Soft MAC**

Firmware needs to be modified!

Motivation:
Prior Projects

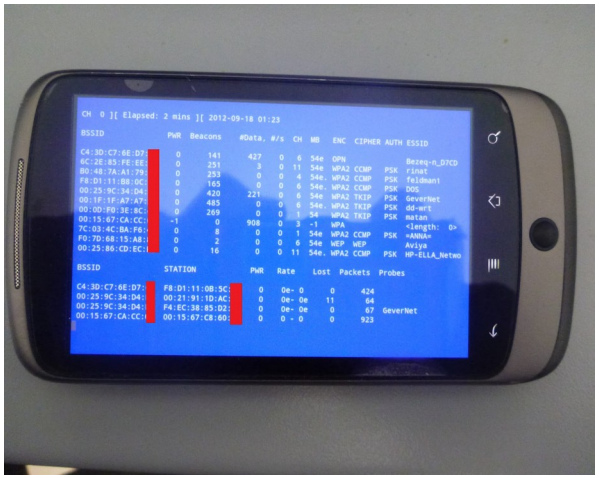
- BCMON
- MONMOB

Source: ifixit.com

Source: ifixit.com

But:

- **No** source code
- **No** new hardware supported



Source: bcmon.blogspot.com

Motivation:
Nexus 5 Wi-Fi Chip

- Google Nexus 5
- Also a Broadcom chip: **bcm4339**
- Supports **802.11n + ac** (incl. 5GHz)
- Capable of **40 and 80MHz** wide channels
- Only 1 antenna (1x1 MIMO)

Code Extraction: Firmware file

- Firmware file in Android file system
  /system/vendor/firmware/fw_bcmdhd.bin

- Lets load it into IDA Pro! But where?

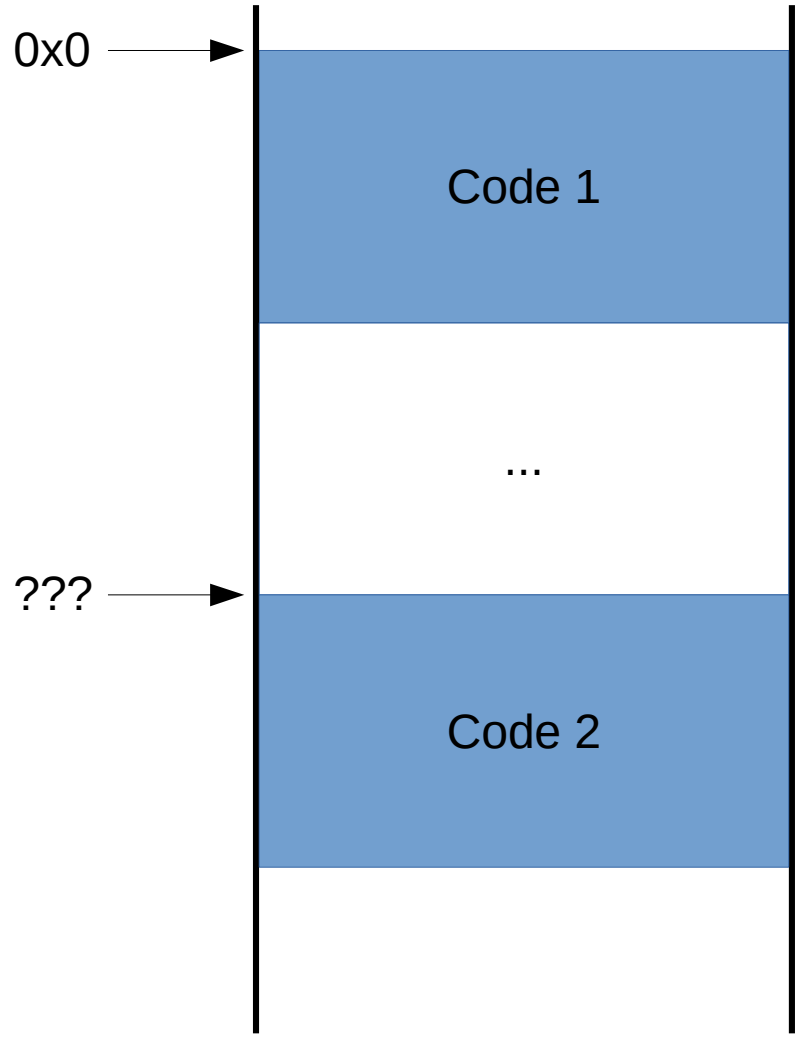
Code Extraction: RAM location

Problem:

Branch (jump) commands are relative:

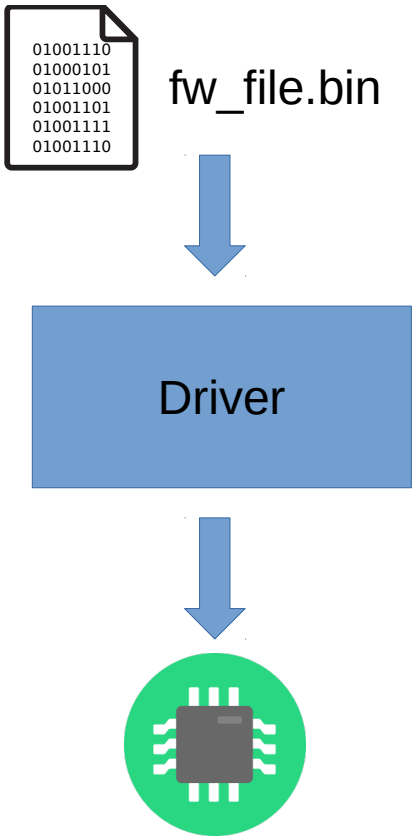
Destination =
Current Location + Offset

=> Code **offset** is important!



Code Extraction: ROM offset

The **driver** must know!




```
static u32 brcmf_chip_tcm_rambase(struct brcmf_chip_priv *ci)
{
        switch (ci->pub.chip) {
        case BRCM_CC_4345_CHIP_ID:
                return 0x198000;
        case BRCM_CC_4335_CHIP_ID:
        case BRCM_CC_4339_CHIP_ID:
        case BRCM_CC_4350_CHIP_ID:
        case BRCM_CC_4354_CHIP_ID:
        case BRCM_CC_4356_CHIP_ID:
        case BRCM_CC_43567_CHIP_ID:
        case BRCM_CC_43569_CHIP_ID:
        case BRCM_CC_43570_CHIP_ID:
        case BRCM_CC_4358_CHIP_ID:
        case BRCM_CC_43602_CHIP_ID:
        case BRCM_CC_4371_CHIP_ID:
                return 0x180000;
        case BRCM_CC_4365_CHIP_ID:
        case BRCM_CC_4366_CHIP_ID:
                return 0x200000;
        default:
                brcmf_err("unknown chip: %s\n", ci->pub.name);
                break;
        }
        return 0;
}
```

Source: http://lxr.free-electrons.com/source/drivers/net/wireless/brcm80211/brcmfmac/chip.c?v=4.4#L670

Code Extraction: Missing Code

- >3000 unknown jump destinations

IDA View-A | Occurrences of strings with error | Hex View-1

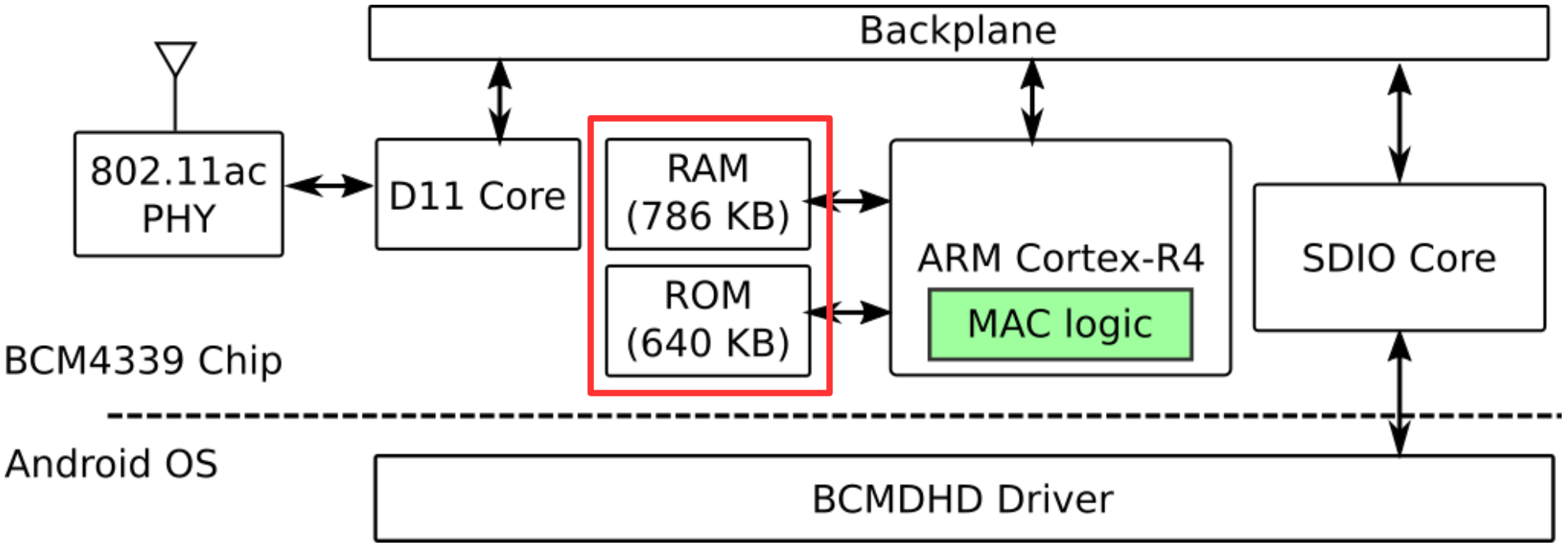
Address	Function	Instruction	
ROM:00181638	sub_181628	BL	0x1269C
ROM:00181644	sub_181628	B.W	0x16578
ROM:0018167C	sub_181674	BL	0x1269C
ROM:001816FE	sub_1816E4	BL	0x1269C
ROM:00181706	sub_1816E4	BL	0x16578
ROM:00181712	sub_1816E4	BL	0x126F0
ROM:00181734	sub_1816E4	BL	0x131E0
ROM:00181750	sub_1816E4	BL	0x1269C
ROM:00181758	sub_1816E4	BL	0x16578
ROM:00181760	sub_1816E4	BL	0x126F0
ROM:00181814	sub_1817A4	BL	0x126F0
ROM:00181842	sub_1817A4	BL	0x126F0
ROM:00181880	sub_1817A4	BL	0x126F0
ROM:0018189E	sub_1817A4	BL	0x126F0
ROM:001818A8	sub_1817A4	BL	0x126F0
ROM:001818BC	sub_1817A4	BL	0x126F0
ROM:00181906	sub_1817A4	BL	0x126F0
ROM:0018191C	sub_1817A4	BL	0x130E8
ROM:00181966	sub_1817A4	BL	0x126F0
ROM:00181988	sub_1817A4	BL	0x126F0
ROM:00181A24		BL	0x16500
ROM:00181A8E	sub_181A88	BL	0x1D9B4
ROM:00181AC0	sub_181A88	BL	0x164BC
ROM:00181AFE	sub_181AF8	BL	0x126F0
ROM:00181B16		BL	0x126F0
ROM:00181B4E	sub_181B28	BL	0x1269C
ROM:00181BB4	sub_181BA0	BL	0x1269C
ROM:00181C1E	sub_181BE0	B.W	0x16620
ROM:00181C2C	sub_181BE0	BL	0x16620
ROM:00181C8E	sub_181C88	BL	0x1DA4C
ROM:00181C9A	sub_181C88	BL	0x1D474
ROM:00181CA8	sub_181C88	BL	0x1DCBC
ROM:00181CFE	sub_181C88	BL	0x1DCDC
ROM:00181D22	sub_181D14	BL	0x168E8

Line 1 of 3982

We are missing some code!

Motivation: FullMAC vs SoftMAC

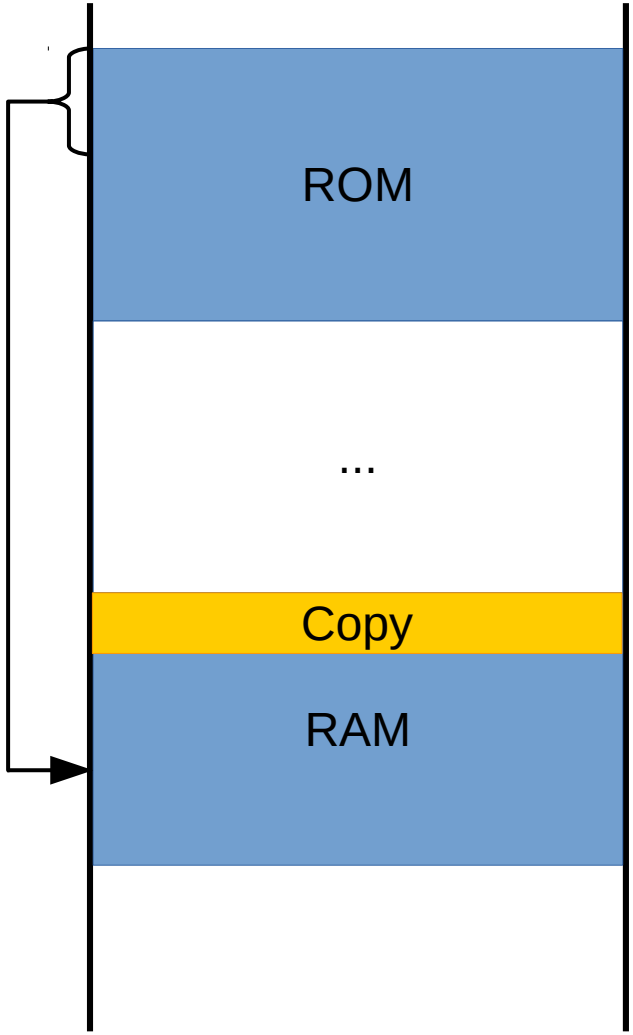
- On the Nexus 5 Wi-Fi chip:



Code Extraction: ROM

Multiple possible ways:

1. Via the Driver: **membytes()** function
2. No ROM access?
   Copy ROM to RAM first, then use membytes()



Code Extraction: FW Structure

- **Bare metal** (but with heap and stack)
- Printf() **=>** built-in **console**!
- **Wrapper** functions:
  - Use **Pointer Table** at the beginning of the RAM
  - Points to functions in ROM
  - Thereby, calls to **ROM functions can be modified** via the Pointers in the RAM!

Code Extraction: Problems

- Lots of code
- **No** function names
- **No** variable names
- Looks ugly, **even decompiled**

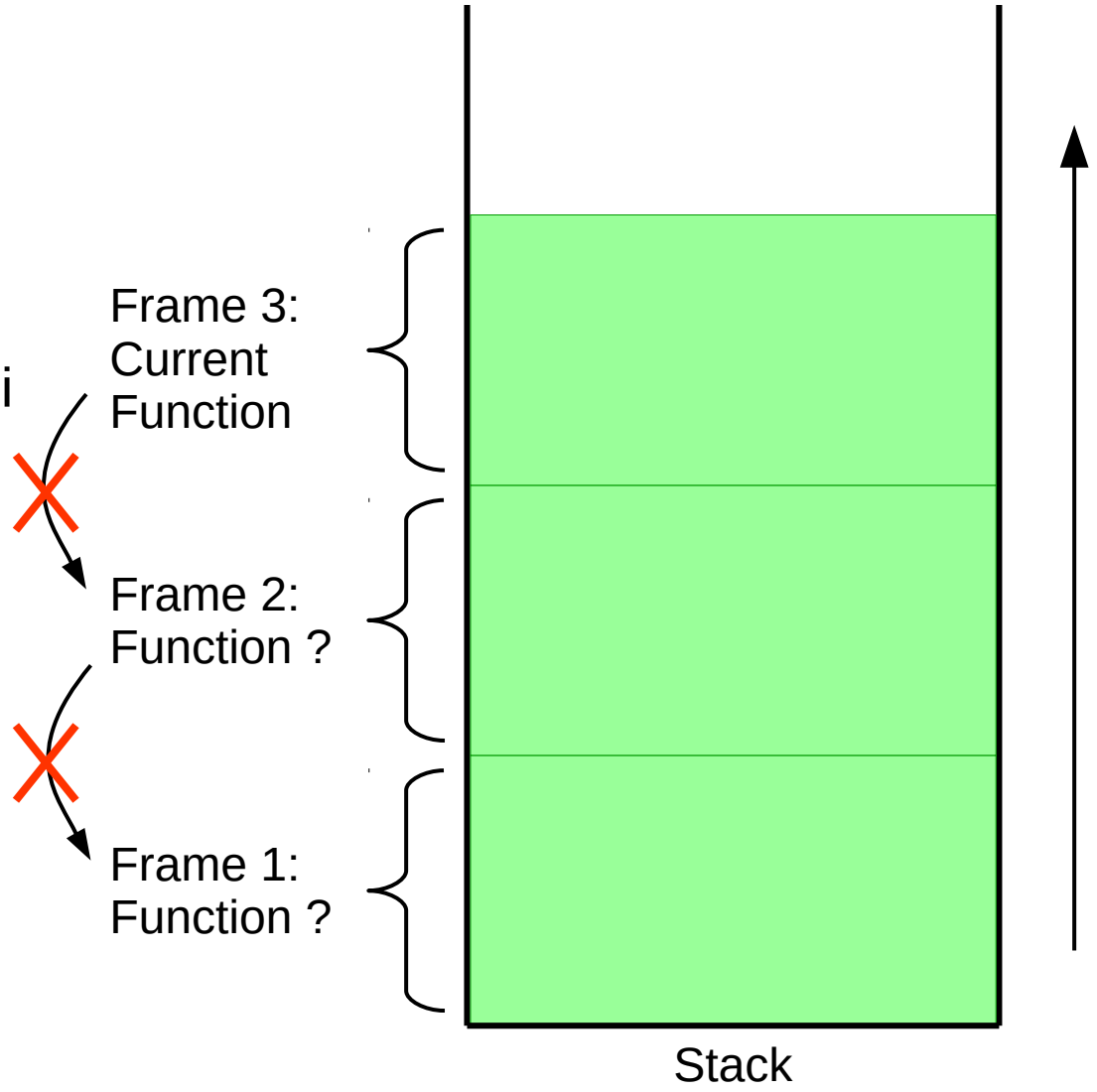
```
int __fastcall sub_18E1E8(int result, unsigned int a2, int a3)
{
  int v3; // r6@1
  unsigned int v4; // r4@1
  int v5; // r7@3
  int v6; // r0@3
  int v7; // r0@3
  int v8; // r3@4
  int v9; // r2@4
  __int16 v10; // r1@4
  unsigned int v11; // r5@11

  v3 = result;
  v4 = a2;
  if ( *(_BYTE *)(result + 672) && a2 )
  {
    v5 = *(_DWORD *)(a2 + 788);
    v6 = sub_6417C(result, a3);
    sub_641F4(v4, v6);
    sub_640C8(v4);
    v7 = sub_18981E();
    result = sub_1897D4(v4, v7);
    if ( !*(_BYTE *)(v4 + 6) )
    {
      v8 = *(_DWORD *)(v4 + 780);
      v9 = *(_DWORD *)(v4 + 792);
      v10 = *(_WORD *)(v9 + 16);
      LOWORD(v9) = *(_WORD *)(v9 + 20);
      *(_WORD *)(v8 + 42) = v10;
      *(_WORD *)(v8 + 44) = v9;
      if ( *(_BYTE *)(v4 + 22) )
      {
        if ( !(*(_DWORD *)(v4 + 256) & 0x2000) && !*(_BYTE *)(v5 + 5) )
          result = sub_1A9EEA(v4, 1);
      }
    }
    *(_BYTE *)(v5 + 6) = 0;
    *(_DWORD *)(v5 + 56) = 0;
    if ( *(_BYTE *)(v5 + 60) && *(_DWORD *)(v5 + 52) == 4 )
      result = sub_1A8338((int *)v4);
    v11 = *(_BYTE *)(v4 + 22);
    if ( !*(_BYTE *)(v4 + 22) && *(_BYTE *)(v5 + 137) == 1 )
    {
      *(_BYTE *)(v5 + 137) = v11;
      sub_3CE84(v3, v4);
      result = sub_32474(v3, v4, 15, v11, v11, v11, v11, v11, v11);
    }
  }
  return result;
}
```

Code Examination: Some tips

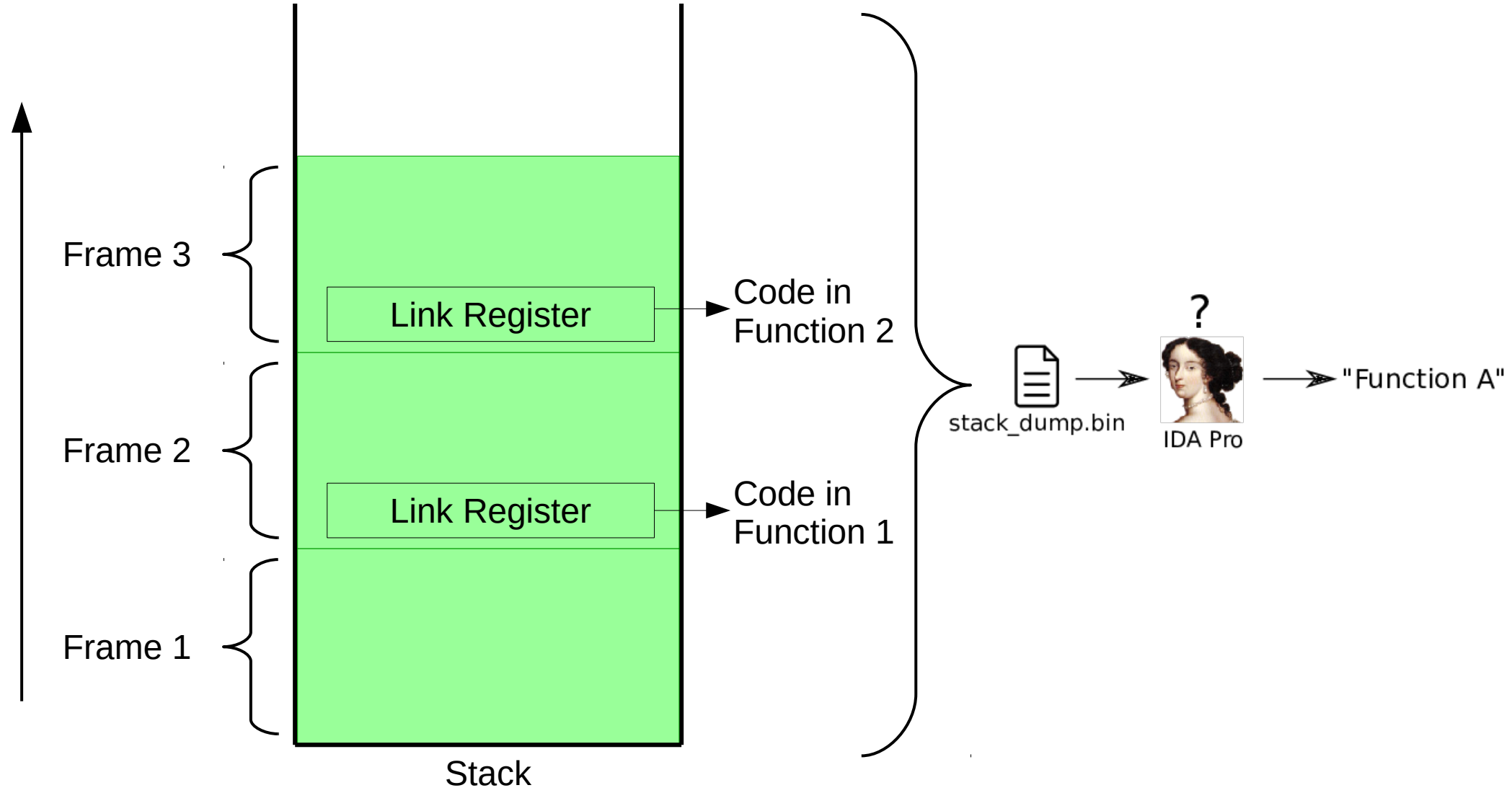
- Search for strings: Debug output via printf()
  - e.g.: *printf("wlc_tpc_get_current: 20in80 clm_limits failed\n");*
- Many similarities to SoftMAC driver (brcmsmac)
  - Where is a function called?
  - What other functions does this function call?
- Look at known byte sequences, e.g. LLC header

Code Examination: Stack Traces

- **Stack Trace:**
  List of functions called along the path to the current function
- **Goal:**
  Find incomming path of Wi-Fi frames
- **e.g.:**
  Function?()
    Function?()
      Function3()
- **Problem:**
  No frame pointer :-(

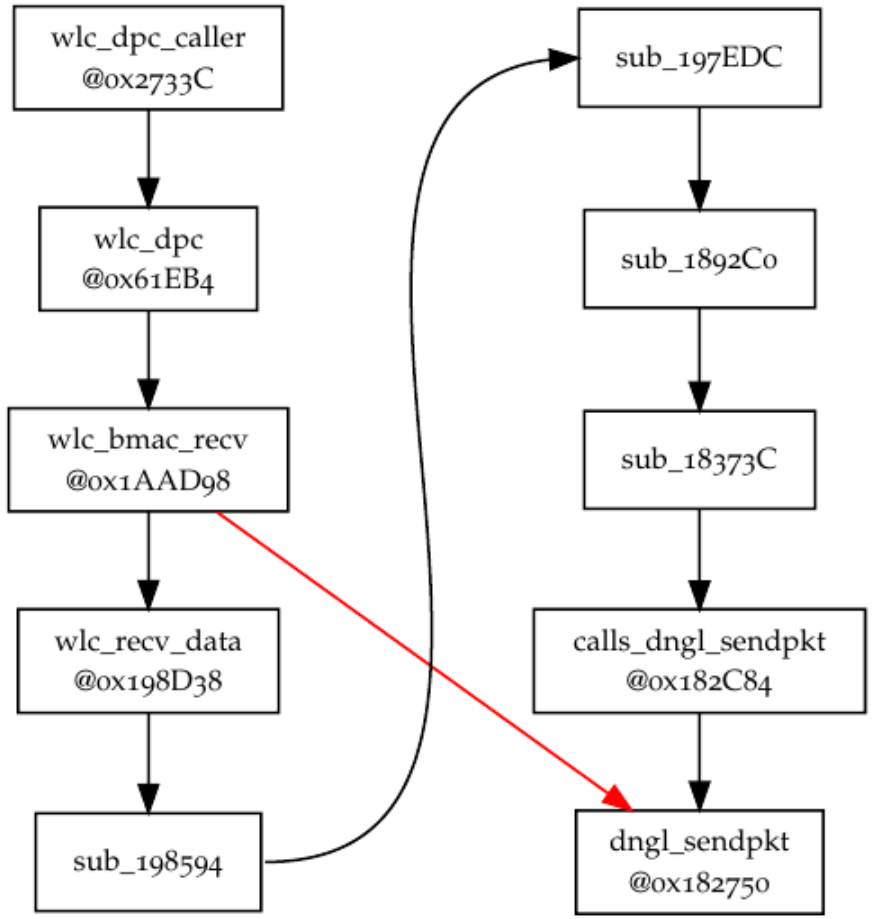


Code Examination: Stack Traces



Code Examination: Stack Traces

1. Find function which handles **received raw frames**
2. Find function which **sends** out frames **to te driver**
3. **Directly** call the outgoing function



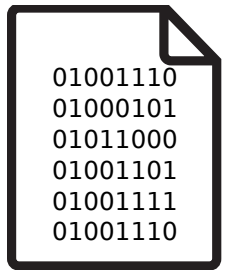
Patching Framework: Overview

- Writing ARM assembly is **tedious** and **error prone** => Write Firmware pachtes in **C** instead!

patch.c

patch.ld

wrapper.h



fw_patched.bin

Patching Framework: Details

patch.c:
Your code goes here, e.g. function hooks

patch.ld:
Where should you code be located in the firmware

wrapper.h:
Function declaration for existing FW functions

patcher.py:
Copy everything together, modify jump commands

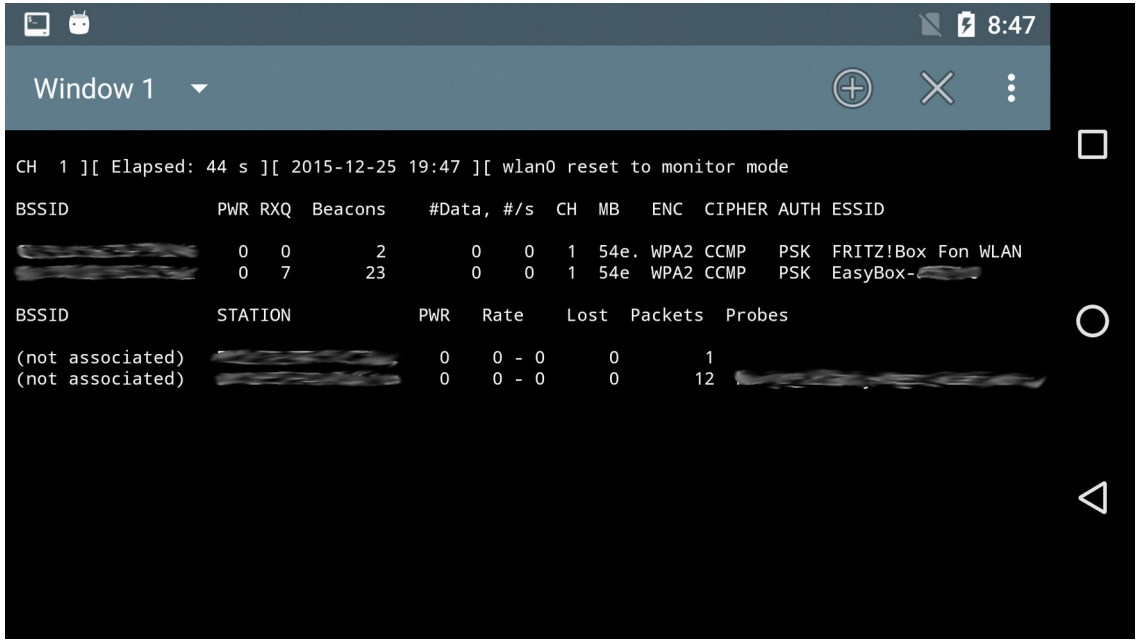
Patching Framework: Benefits

Modifiying **any code** in the RAM

- **Calling** existing firmware functions
- **Easily modifying** existing firmware functions
  e.g.: write function hooks and mess with the parameters
- **Template Projects** help you to create your own firmware patch!

We did it team!

- Working Monitor Mode
- Aircrack-ng tools work, tested:
  - Airodump-ng
  - Aireplay-ng (deautentication attack)



Demo

Enough talk!
Show me a Demo!

Native Monitor Mode: IOCTLs

- Monitor and Promisc IOCTLs
  - WLC_SET_MONITOR
  - WLC_SET_PROMISC
- In the firmware:

```
if ( *(_DWORD *)&wlc_ptr->monitor )
{
  if ( rxh->RxStatus2 & 1 )
    sub_2C2CC((int)wlc_ptr, (int)rxh, (int)p);
  else
    sub_18D648(wlc_ptr, (int)rxh, p, 0);
}
```

wlc_monitor_amsdu(): Function for aggregated frames

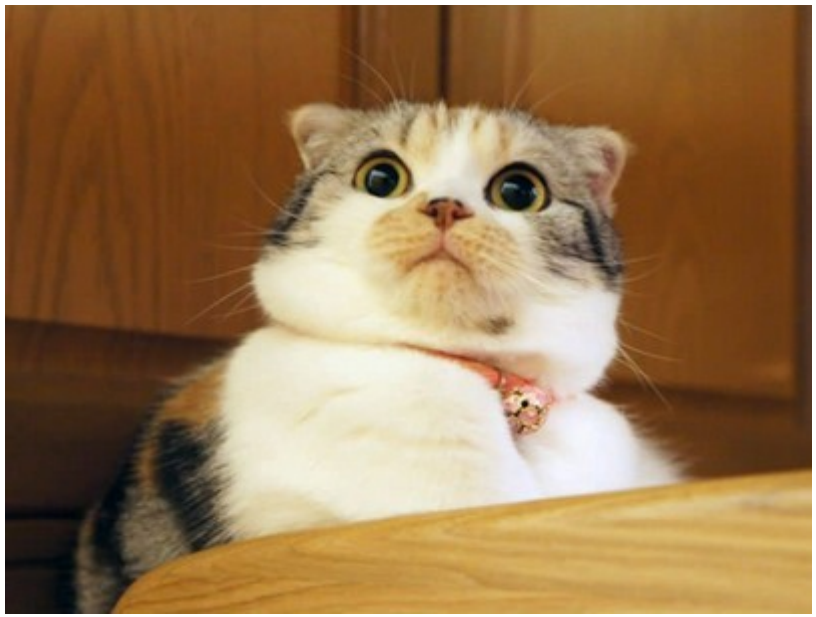
wlc_monitor(): Function for all other frames

- Most Broadcom chips got a built-in Monitor Mode!

Native Monitor Mode: Drawbacks

- No **Radiotap** header
  => airodump-ng will not work
- No **Injection** support
  => aireplay-ng will not work

We **fixed** this in our current Monitor Mode patch!



Related Projects: Raspberry Pi 3

- Raspberry Pi 3 (BCM43438)
- Simple Monitor Mode works!
- ToDo:
  - Switching channels
  - Injection
  - Radiotap header infos: RSSI, Channel, Timestamps
- Checkout: **rpi3.nexmon.org**

nexmon

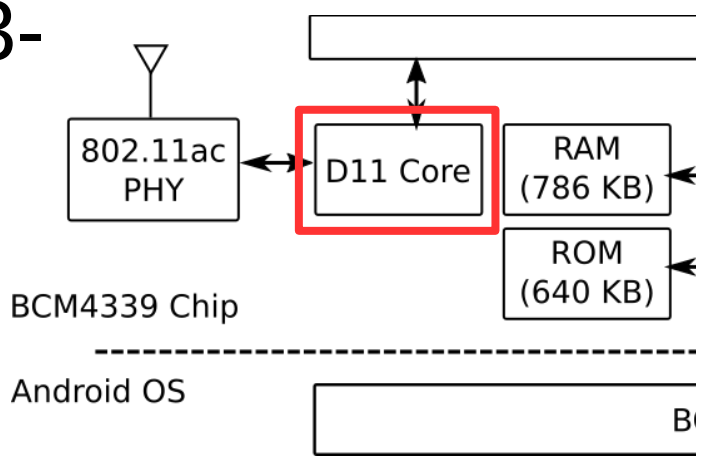
RaspberryPi

Source: raspberry.piaustralia.com.au

Related Projects

D11 core modifications

- Separate firmware
- Responsible for **time critical** operations
- **Disassembler** available: https://github.com/mbuesch/b43-tools



Future Work

- **Fixing bugs** in the Raspberry Pi 3 Monitor Mode
  - ROM Modifications using FPB (Flash Patch and Breakpoint)
- Enalbe Monitor Mode on **more devices**
  - Nexus 6P
  - Other Broadcom Chips

Source: ifixit.com

Thank you for listening

Contact:

- Write us an E-Mail:
  - Daniel Wegemer: **dwegemer@seemoo.de**
  - Matthias Schulz: **mschulz@seemoo.de**
- Follow us on twitter: **@nexmon_dev**
- Visit **nexmon.org** and **rpi3.nexmon.org** for complete Source Code!