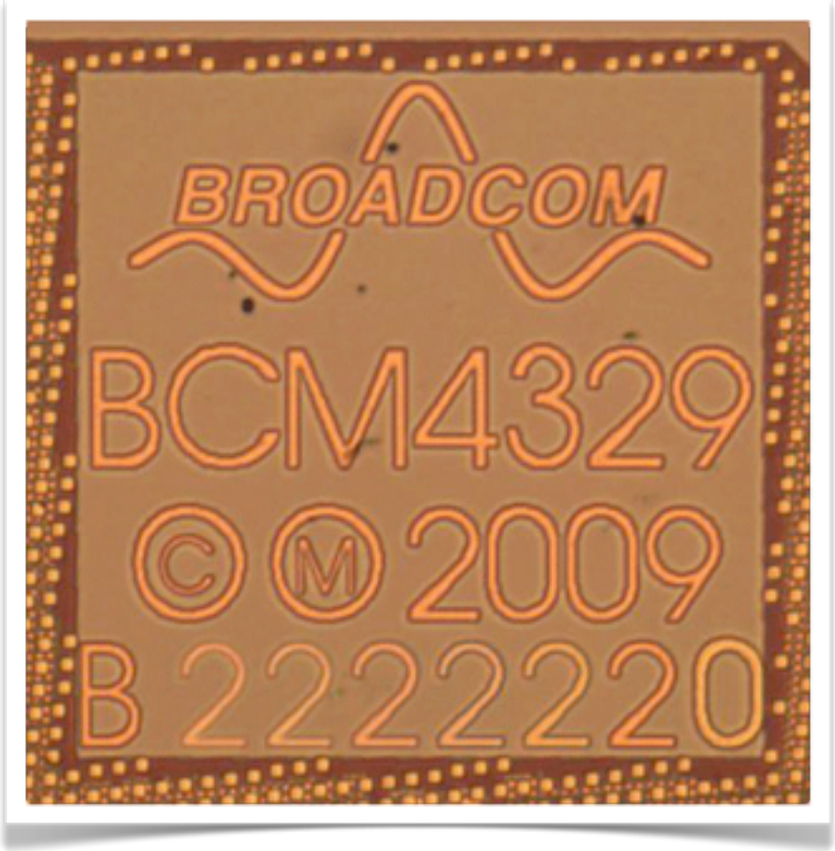
One firmware to monitor 'em all



Andrés Blanco - Matías Eissler

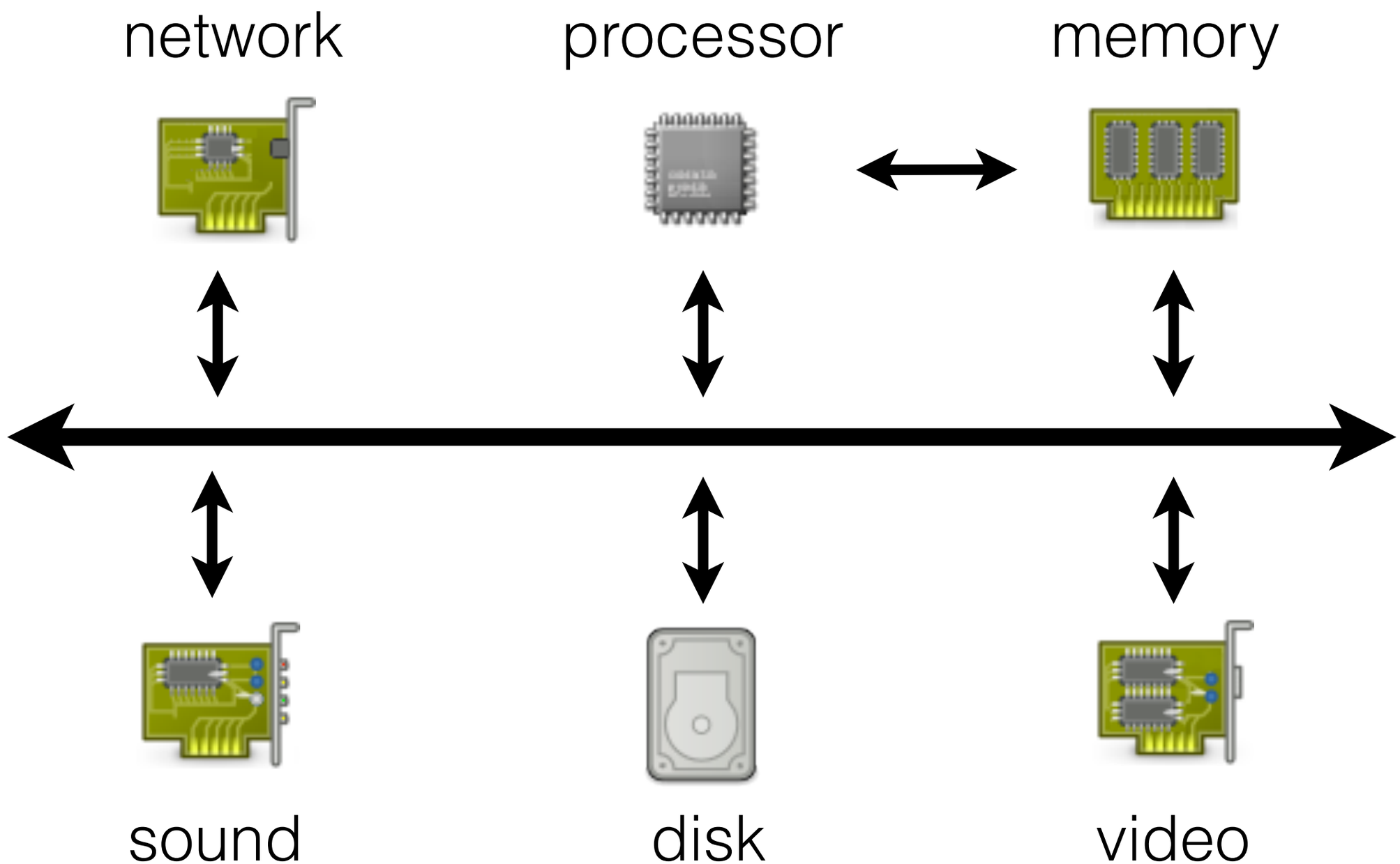
CORE SECURITY™

HACK.LU

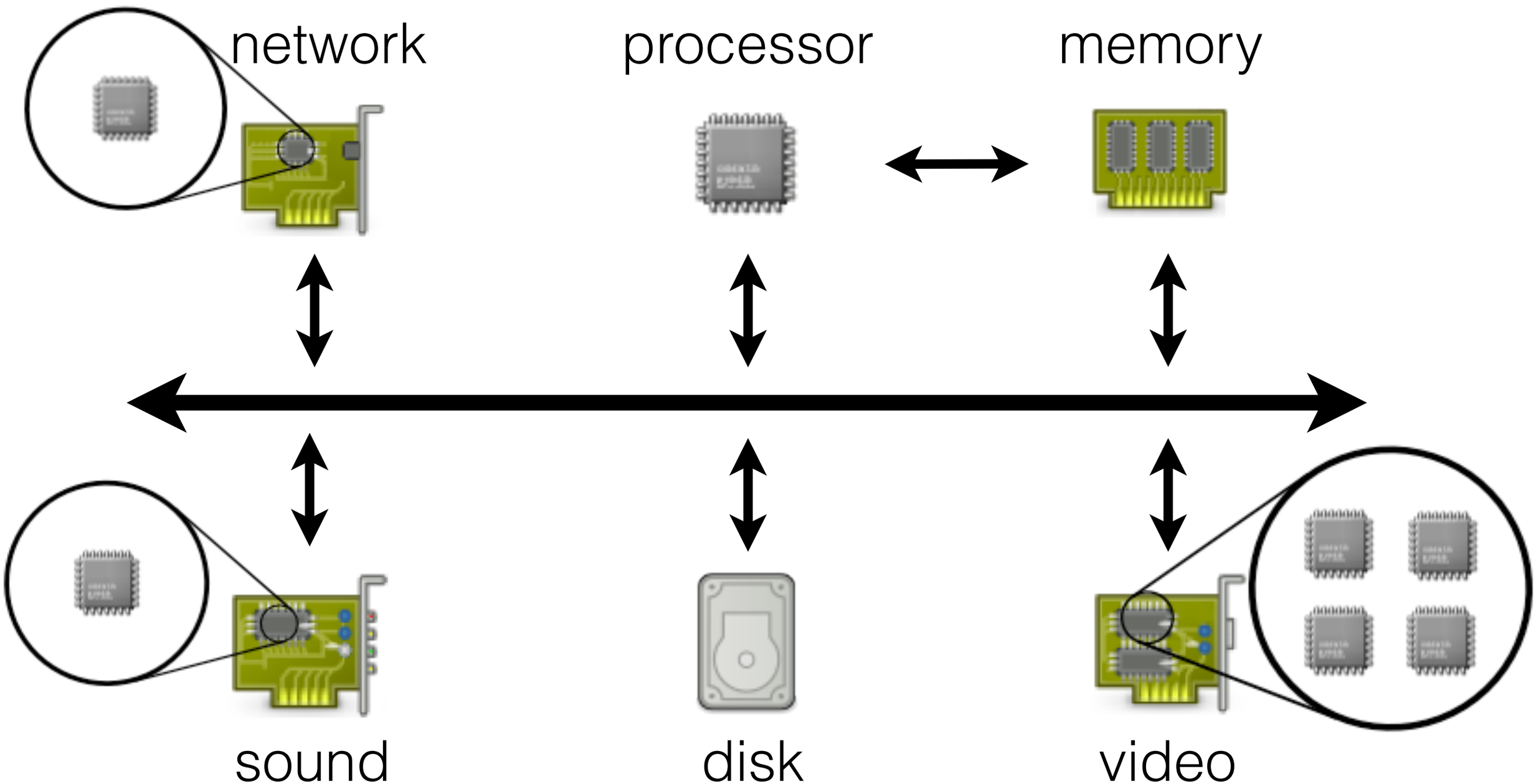
Agenda

- Intro
- Motivation
- Reverse engineering process
- Patching
- Monitor mode
- Injection

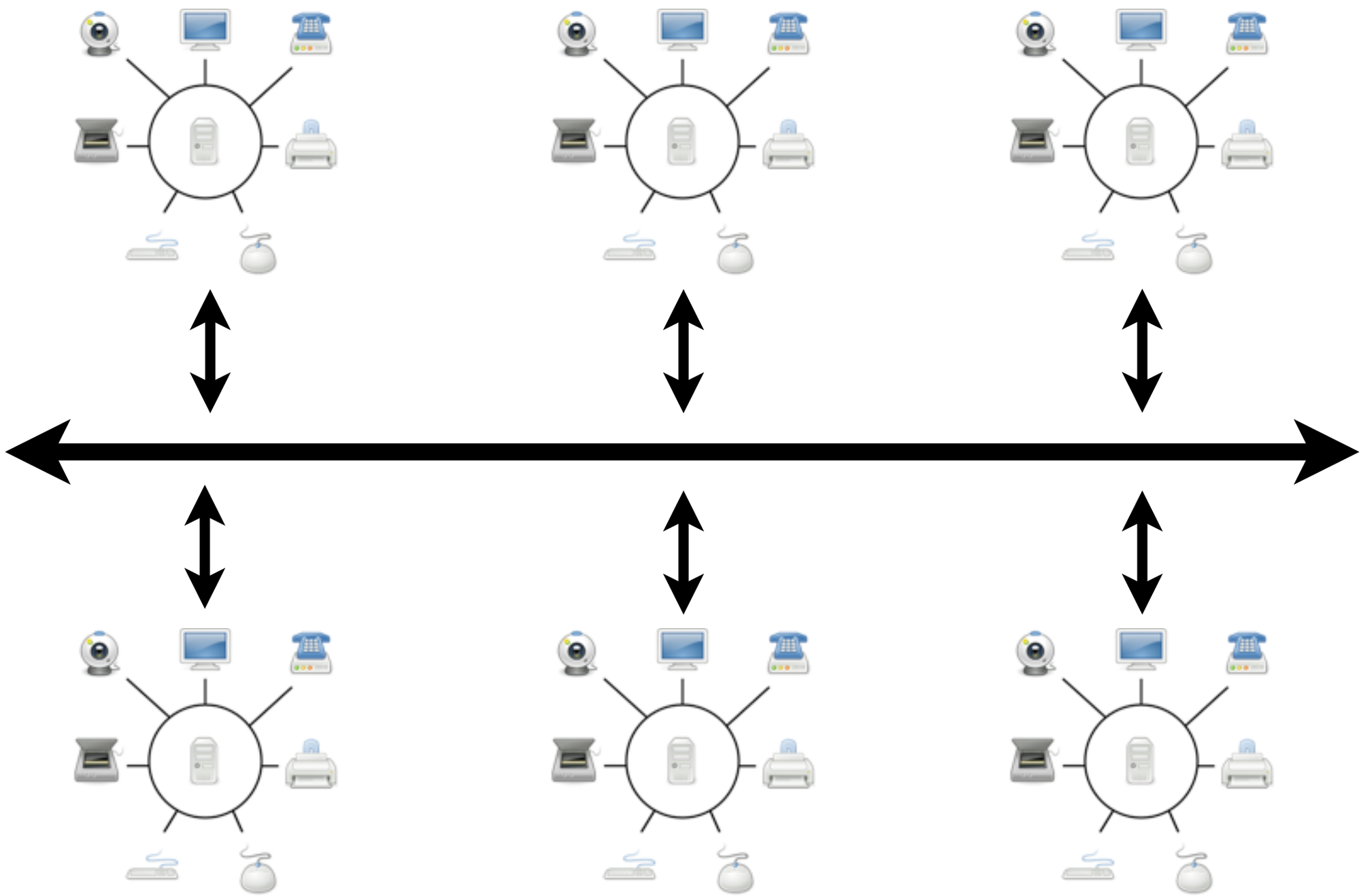
Everything but the processor is a peripheral



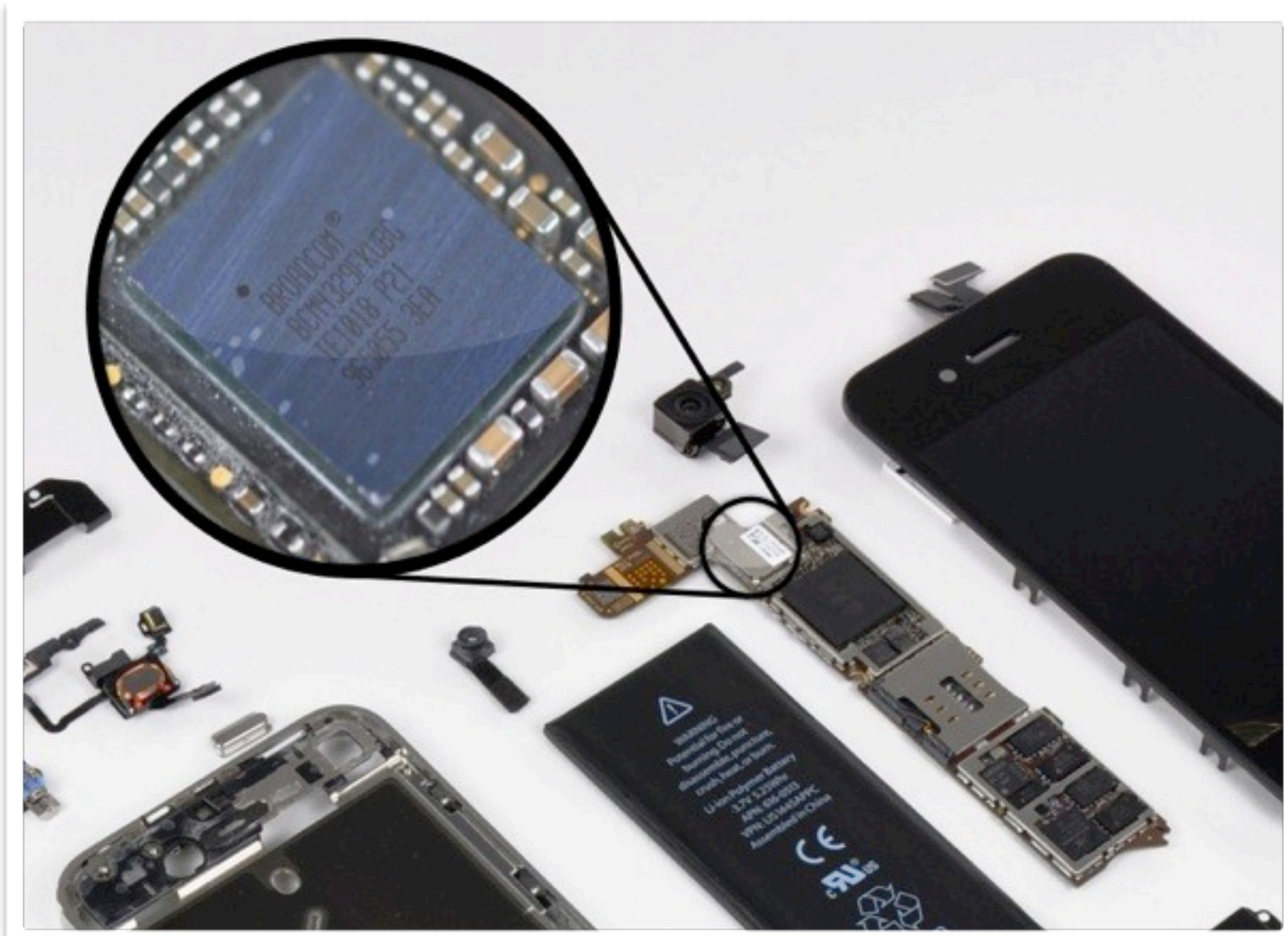
But which processor?



Peripherals as computers



Size does matter



Motivation

	PC	Mobile
Persistent code Closer to metal: Reverse engineering the Broadcom NetExtreme's firmware G Delugre - hack.lu 2010 [1]	✓	?
NiC to OS through DMA Can you still trust your network card? L Duflot, et al. - Cansec 2010 [2]	✓	?
Exploiting IO/MMU Exploiting an i/ommu vulnerability F. L. Sang, et. al. MALWARE - 2010 [3]	✓	?
Hardware direct P2P Project Maux Mk. II, I Own the NIC, now I want a shell A Triulzi - PacSec 2008 [4]	✓	?
Attacks drivers "from below" The jedi packet trick takes over the deathstar A. Triulzi - Cansec 2010 [5]	✓	?

Motivation (cont)

	PC	Mobile
Man-in-the-middle	✓	✓
Firewall bypass / bridge	✓	?
802.11 Monitor Mode	✓	✓
802.11 Raw frame injection	✓	✓

Some vendors

Some devices

- iPod Touch 2 generation
- iPod Touch 3 generation
- iPad 1 generation
- iPad 2 generation
- iPad 3 generation
- iPhone 3GS
- iPhone 4
- iPhone 4S
- Apple TV 2 generation
- Apple TV 3 generation

- Spica
- Galaxy Tab
- Galaxy S 4G
- Nexus S
- Stratosphere
- Fascinate
- Galaxy S2

- Devour
- Xoom
- Droid x2
- Atrix

The Firmware

- Common file in the OS file system:
  - /usr/share/firmware/wifi/43xx/
  - /system/etc/wifi/
- Not signed!
- Closed source.
- Loaded at boot time by the NiC Driver.

Binary chunk?

```
00000000  00 00 00 00 7d 0b 00 00  89 00 00 00 89 00 00 00  |....}...........|
00000010  89 00 00 00 89 00 00 00  89 00 00 00 89 00 00 00  |................|
00000020  89 00 00 00 89 00 00 00  89 00 00 00 89 00 00 00  |................|
00000030  89 00 00 00 89 00 00 00  89 00 00 00 89 00 00 00  |................|
00000040  89 00 00 00 89 00 00 00  89 00 00 00 89 00 00 00  |................|
00000050  89 00 00 00 89 00 00 00  89 00 00 00 89 00 00 00  |................|
00000060  89 00 00 00 89 00 00 00  89 00 00 00 89 00 00 00  |................|
00000070  89 00 00 00 89 00 00 00  89 00 00 00 89 00 00 00  |................|
00000080  00 48 00 47 7d 0b 00 00  68 46 83 69 41 69 0b b5  |.H.G}...hF.iAi..|
00000090  03 69 5a 46 51 46 0e b4  4a 46 41 46 06 b4 c3 68  |.iZFQF..JFAF...h|
000000a0  82 68 41 68 fe b4 03 68  c2 69 ef f3 03 81 0e b4  |.hAh...h.i......|
000000b0  82 69 ef f3 05 81 06 b4  03 48 01 68 00 29 fe d0  |.i.......H.h.)..|
000000c0  68 46 88 47 14 b0 00 bd  98 d9 02 00 00 00 00 00  |hF.G............|
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

Architecture

- How to detect the architecture?
  - google
  - common sense (binary code should make sense)
  - bruteforcing
  - learning
- ARM Cortex M3 [6]

Instruction Set

- Can be identified by Undefined Instruction Exception, using google or just trying.
- BCM 4325 → ARMv6
- BCM 4329/30 → ARMv7

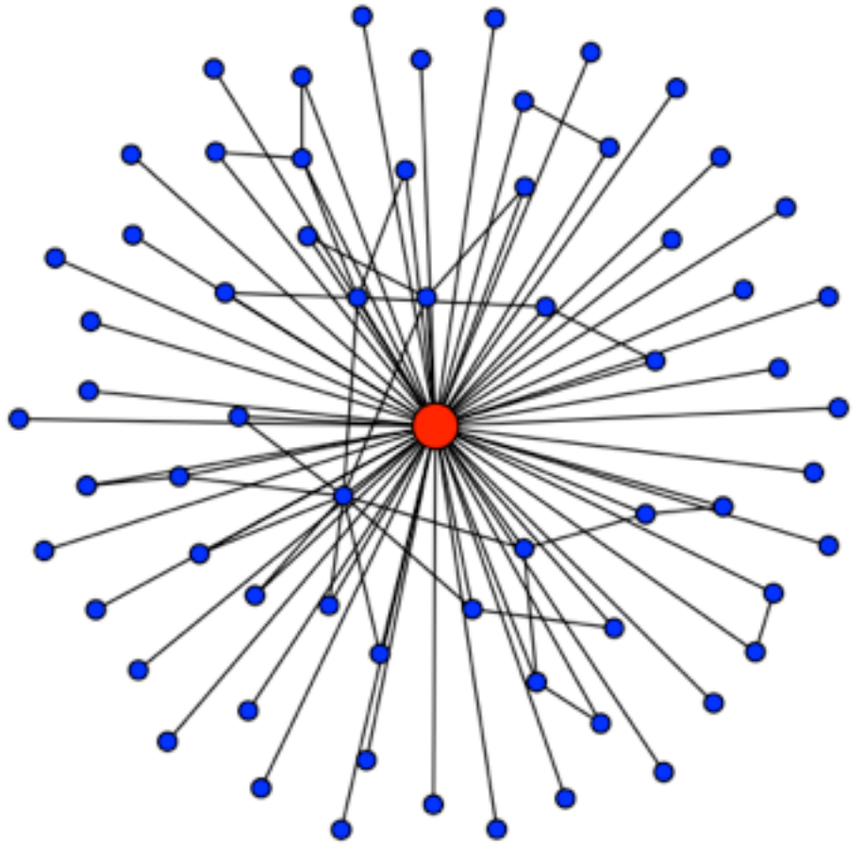
Disassembling

- ARM Functions must be aligned to 4 bytes (learned the hard way).
- Prologues are padded with 2-byte NOP.
- Not all functions start with prologue.

```
                    sub_778
000 13 4B           LDR     R3, =dword_2F310
000 2D E9 F0 41     PUSH.W  {R4-R8,LR}
018 D3 F8 00 80     LDR.W   R8, [R3]
018 12 4B           LDR     R3, =unk_2F378
018 DB 88           LDRH    R3, [R3,#(word_2F37E - 0x2F378)]
018 08 B9           CBNZ    R0, loc_78C
```

Primitive function identification

- Three tricks to identify functions:
  - Most called technique [7]
  - Memory address vecinity
    strcpy, strncpy, strcmp, strncmp
  - Puzzle Identification:
    memset(p, 0, n) -> p = malloc(n)



802.11 function identification

Introduction

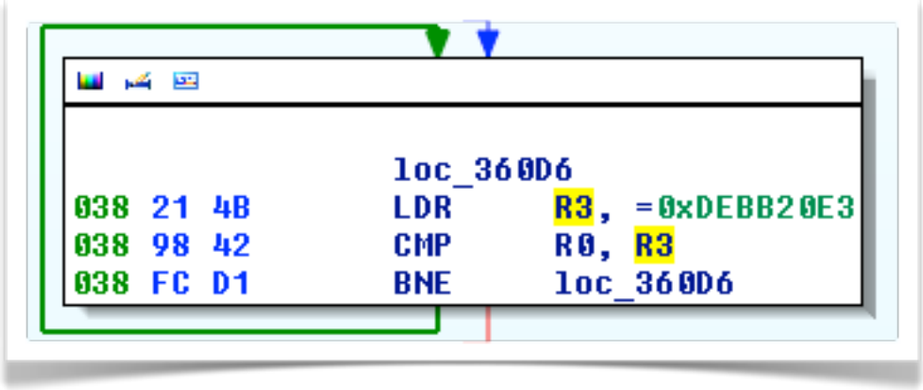
```
⊟ IEEE 802.11 Probe Request, Flags: ........C
    Type/Subtype: Probe Request (0x04)
  ⊞ Frame Control: 0x0040 (Normal)
    Duration: 0
    Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Source address: d8:a2:5e:51:56:a6 (d8:a2:5e:51:56:a6)
    BSS Id: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Fragment number: 0
    Sequence number: 1368
  ⊞ Frame check sequence: 0x0d66be50 [correct]
⊟ IEEE 802.11 wireless LAN management frame
  ⊟ Tagged parameters (89 bytes)
    ⊞ Tag: SSID parameter set: Broadcast
    ⊞ Tag: Supported Rates 1, 2, 5.5, 11, [Mbit/sec]
    ⊞ Tag: Extended Supported Rates 6, 9, 12, 18, 24, 36, 48, 54, [Mbit/sec]
    ⊞ Tag: HT Capabilities (802.11n D1.10)
    ⊞ Tag: Vendor Specific: 00:10:18
    ⊞ Tag: Vendor Specific: 00:90:4c: HT Capabilities (802.11n D1.10)
```

802.11 Function identification

- Probe request (Epigram OUI)
- 6-byte memcpy/memcmp
- 802.11 header addresses pattern
- Found many 802.11 implementation Function:
  - searchForIE, beaconHandler, createFrameHeader, searchForVendorSpecific, etc.

Patching the firmware

- Modifying strings
- Finding the checksum.
  - Firmware verifies itself.
  - 0xDEBB20E3 magic constant.
  - FindCrypt IDA Plugin
- Patching ethernet addresses.



The missing code

- References to code on memory address outside known section (ROM).

```
loc_1A1A2
MOVS     R1, #0
LDR      R4, =0x1E0019A9
MOV.W    R0, #0x200
BLX      R4
MOVS     R1, #0
STR.W    R0, [R7,#0x7F8]
MOVS     R0, #0x80 ; 'Ç'
BLX      R4
LDR.W    R1, [R7,#0x7F8]
STR.W    R0, [R7,#0x7FC]
CBZ      R1, loc_1A1D0
```

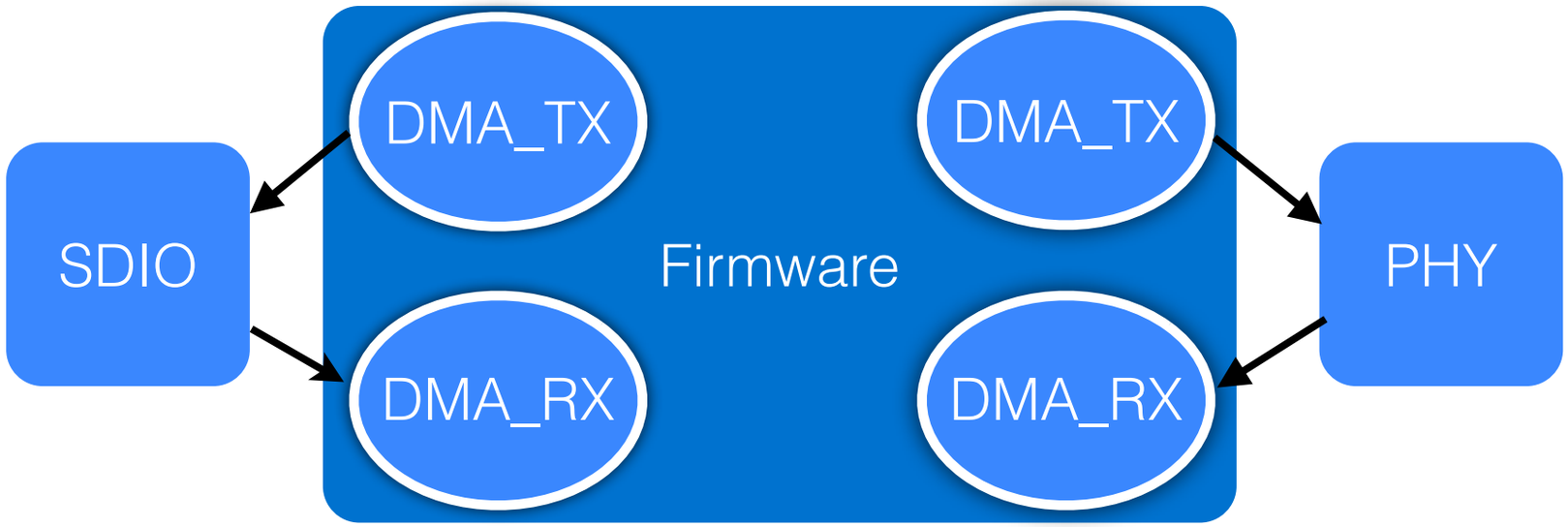
Thanks Android (leak?)

```
#define SI_FLASH2                0x1c000000      /* Flash Region 2 (regio/
#define SI_FLASH2_SZ             0x02000000      /* Size of Flash Region 2
#define SI_ARMCM3_ROM            0x1e000000      /* ARM Cortex-M3 ROM */
#define SI_FLASH1                0x1fc00000      /* MIPS Flash Region 1 */
#define SI_FLASH1_SZ             0x00400000      /* MIPS Size of Flash
#define SI_ARM7S_ROM             0x20000000      /* ARM7TDMI-S ROM */
#define SI_ARMCM3_SRAM2          0x60000000      /* ARM Cortex-M3 SRAM
#define SI_ARM7S_SRAM2           0x80000000      /* ARM7TDMI-S SRAM Region
#define SI_ARM_FLASH1            0xffff0000      /* ARM Flash Region 1 */
#define SI_ARM_FLASH1_SZ         0x00010000      /* ARM Size of Flash
--
Linux kernel driver for BCM source code [8]
```

How to dump the rom?

Dumping the ROM

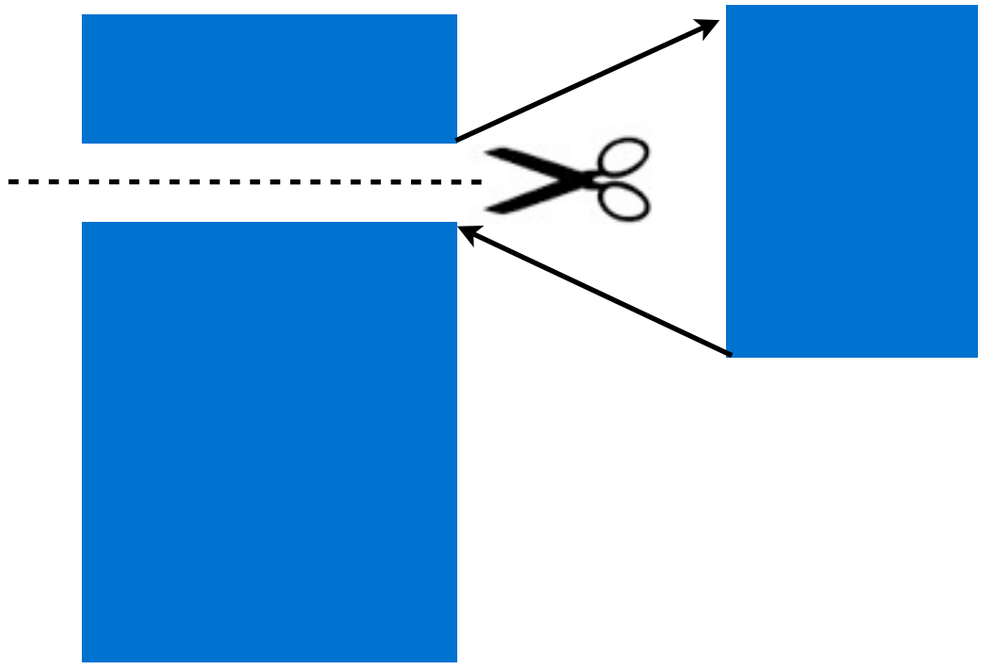
- Dump to air
- Dump to kernel
- IOCTL



Towards monitor mode

Obtaning Monitor Mode

- Getting 802.11 & PHY Headers



- Getting all the traffic (Management, Control & Data).

- wlc_bmac_mctrl() function.

Mac Control Flags

```
void wlc_bmac_mctrl(struct wlc_hw_info *wlc_hw, u32 mask, u32 val)
{
        u32 maccontrol;
        u32 new_maccontrol;

        if (val & ~mask)
                return; /* error condition */

        maccontrol = wlc_hw->maccontrol;
        new_maccontrol = (maccontrol & ~mask) | val;

        if (new_maccontrol == maccontrol)
                return;

        wlc_hw->maccontrol = new_maccontrol;
        wlc_mctrl_write(wlc_hw);
}
```

Android source code for BCM drviers [9]

- ARM BIC (Bit Clear) Instruction.

Mac control flags

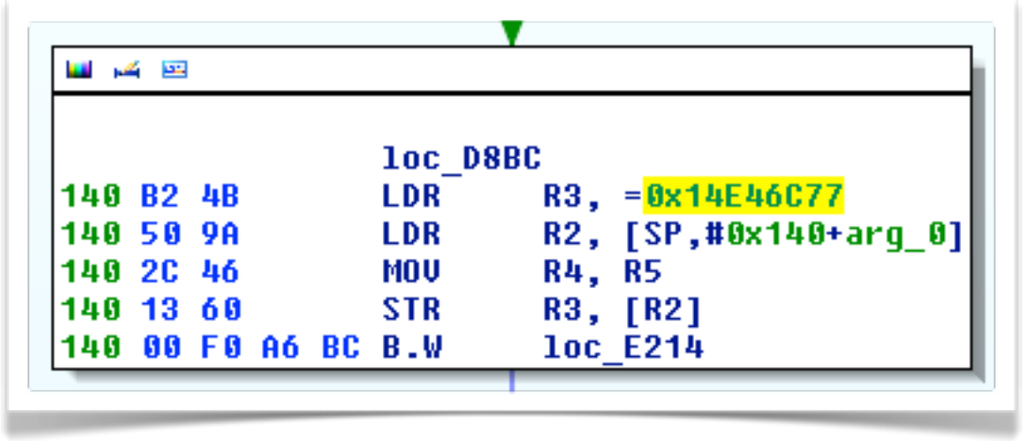
```
/* maccontrol register */
#define MCTL_GMODE                (1U << 31)
#define MCTL_DISCARD_PMQ          (1 << 30)
#define MCTL_WAKE                 (1 << 26)
#define MCTL_HPS                  (1 << 25)
#define MCTL_PROMISC              (1 << 24)
#define MCTL_KEEPBADFCS           (1 << 23)
#define MCTL_KEEPCONTROL          (1 << 22)
#define MCTL_PHYLOCK              (1 << 21)
#define MCTL_BCNS_PROMISC         (1 << 20)
#define MCTL_LOCK_RADIO           (1 << 19)
#define MCTL_AP                   (1 << 18)
#define MCTL_INFRA                (1 << 17)
#define MCTL_BIGEND               (1 << 16)
#define MCTL_GPOUT_SEL_MASK       (3 << 14)
#define MCTL_GPOUT_SEL_SHIFT      14
#define MCTL_EN_PSMDBG            (1 << 13)
#define MCTL_IHR_EN               (1 << 10)
#define MCTL_SHM_UPPER            (1 <<  9)
#define MCTL_SHM_EN               (1 <<  8)
```

Android source code for BCM drivers [10]

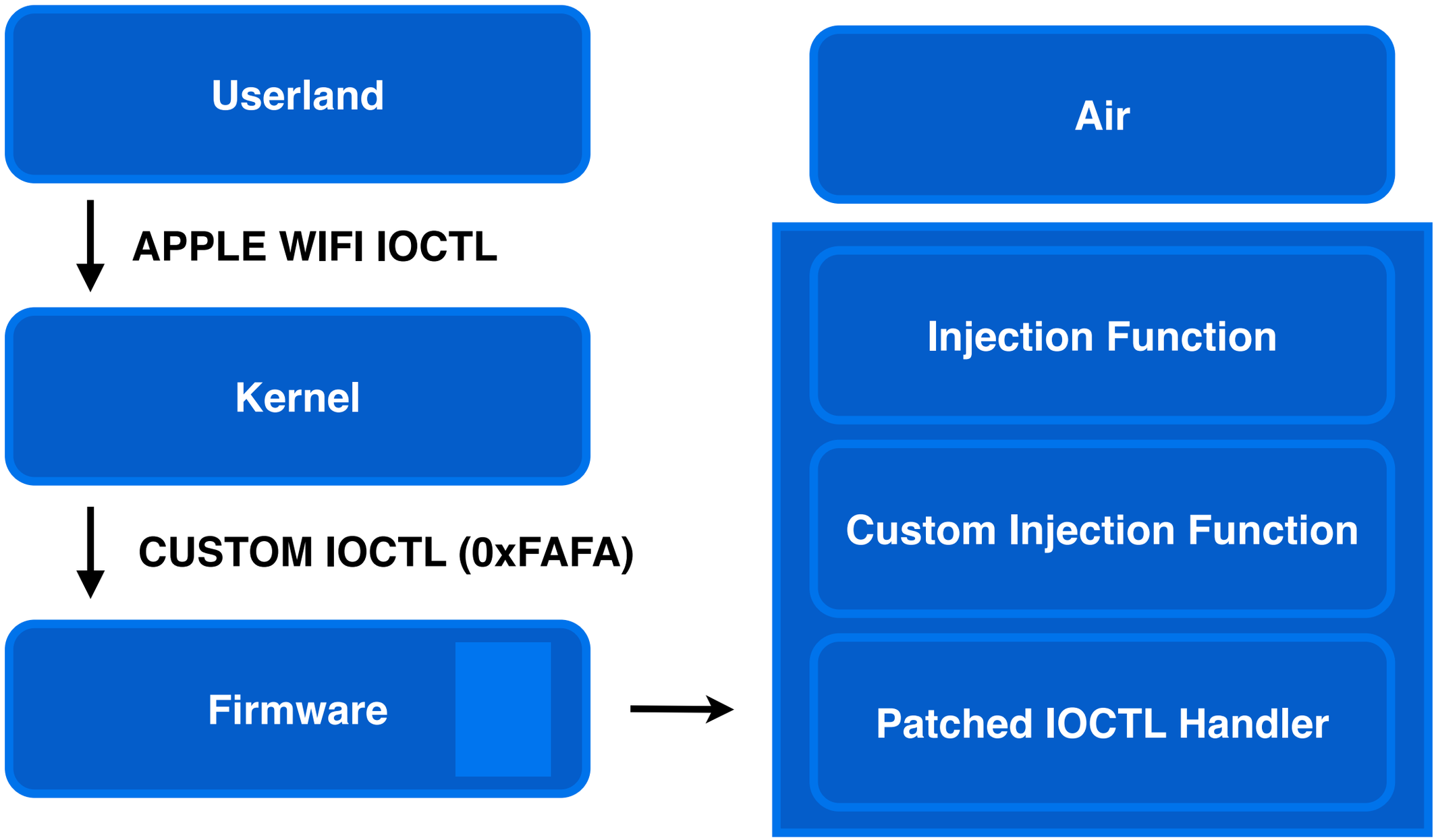
Monitor mode

I want to inject!

- **IOCTL handler function**
  - WLC_MAGIC IOCTL 0x14e46c77
  - LARGEST SWITCH
- **wlc_sendpkt_mac80211 function**
  - Follow the path from probe request



Injection scheme

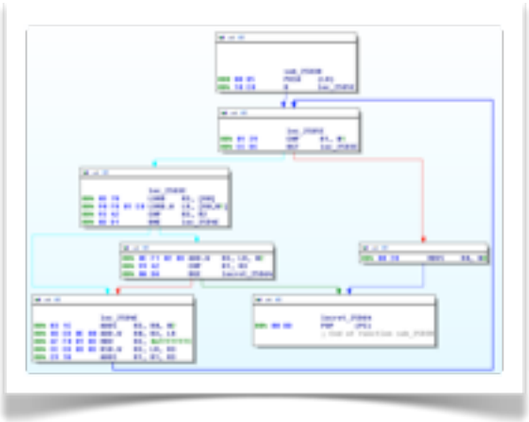


Injection time

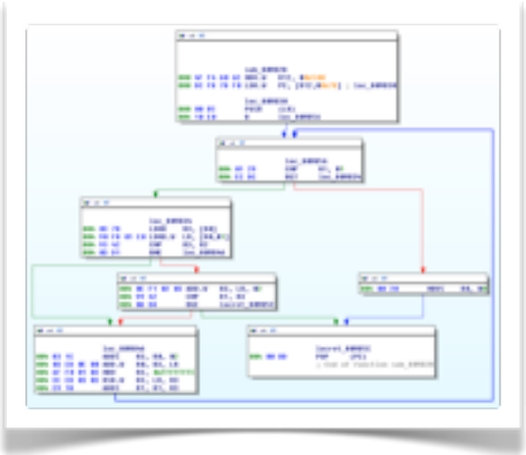
Possible attacks

- Monitor Wireless Networks remotely.
- Perform MiTM attacks (such as SSL strip).
- Control the flow of the frames (create/drop) without the OS notice.
- ARP/DNS cache poisoning.
- Create 802.11 covert channels.
- Leak Information using 802.11 frames.

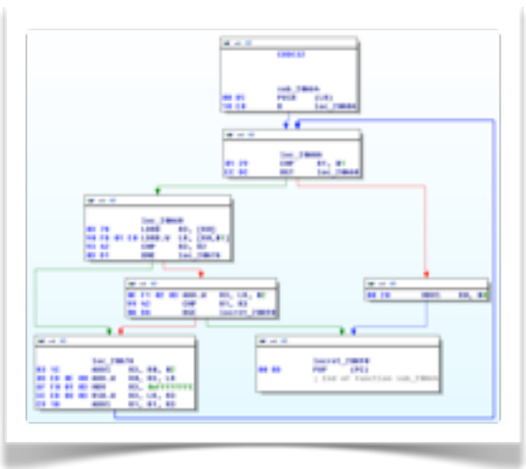
One Firmware?



BCM4329 - iPad 1 Generation



BCM4330 - iPhone 4S



BCM4329 - Galaxy Tab

Appearance

Appearance

Questions

Gracia'

- Ezequiel Gutesman & Anibal Sacco (for helping out)
- iOS & Android Jailbreakers (for making devices free)
- Ekoparty (for the good wave)
- Starbucks cafe (for the crappy internet and long hours of reversing)
- Our wives (for the sundays).

References

[1] Guillaume Delugré. Closer to metal: reverse-engineering the broad- com netextreme's firmware. Hack.lu, 2010 - http://esec-lab.sogeti.com/dotclear/public/publications/10-hack.lu-nicreverse_slides.pdf

[2] Loïc Duflot, Yves-Alexis Perez, Guillaume Valadon, and Olivier Levillain. Can you still trust your network card? CanSecWest Applied Security Conference, 2010 - http://www.ssi.gouv.fr/IMG/pdf/csw-trustnetworkcard.pdf

[3] F. L. Sang, E. Lacombe, V. Nicomette, and Y. Deswarte. Exploit- ing an i/ommu vulnerability. In Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on, pages 7–14 2010 - http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5665798

[4] Arrigo Triulzi. Project maux mk. ii, i own the nic, now i want a shell. The 8th annual PacSec conference, 2008 - http://www.alchemistowl.org/arrigo/Papers/Arrigo-Triulzi-PACSEC08-Project-Maux-II.pdf

[5] Arrigo Triulzi. The jedi packet trick takes over the deathstar. tak- ing nic backdoors to the next level. CanSecWest Applied Security Conference, 2010 - http://www.alchemistowl.org/arrigo/Papers/Arrigo-Triulzi-CANSEC10-Project-Maux-III.pdf

[6] BCM4330 brochure http://www.broadcom.com/products/Wireless-LAN/802.11-Wireless-LAN-Solutions/BCM4330

[7] IDAPython script to find memcpy http://exploiting.wordpress.com/2012/07/02/quickpost-idapython-locating-libc-in-an-unknown-firmware-without-string-references/

[8] Source http://lxr.free-electrons.com/source/drivers/staging/brcm80211/include/hndsoc.h?v=2.6.37;a=arm

[9] More source http://lxr.free-electrons.com/source/drivers/staging/brcm80211/sys/wlc_bmac.c?v=2.6.38#L1610

[10] Even more source http://lxr.free-electrons.com/source/drivers/net/wireless/brcm80211/brcmsmac/d11.h#L458