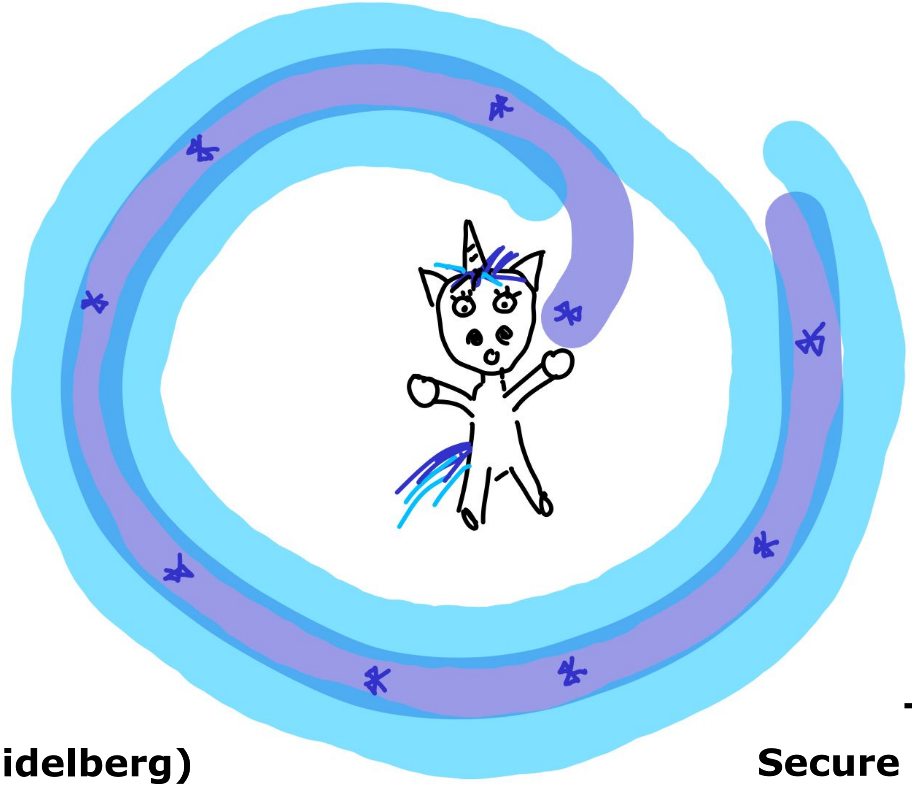
Bluetooth, does it spark joy?



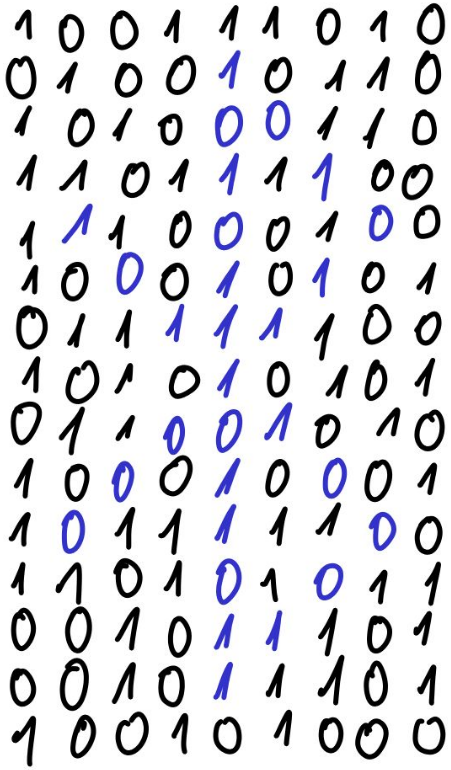
Dennis Mantz
Security Analyst @ ERNW GmbH (Heidelberg)

Jiska Classen
Technische Universität Darmstadt
Secure Mobile Networking Lab - SEEMOO

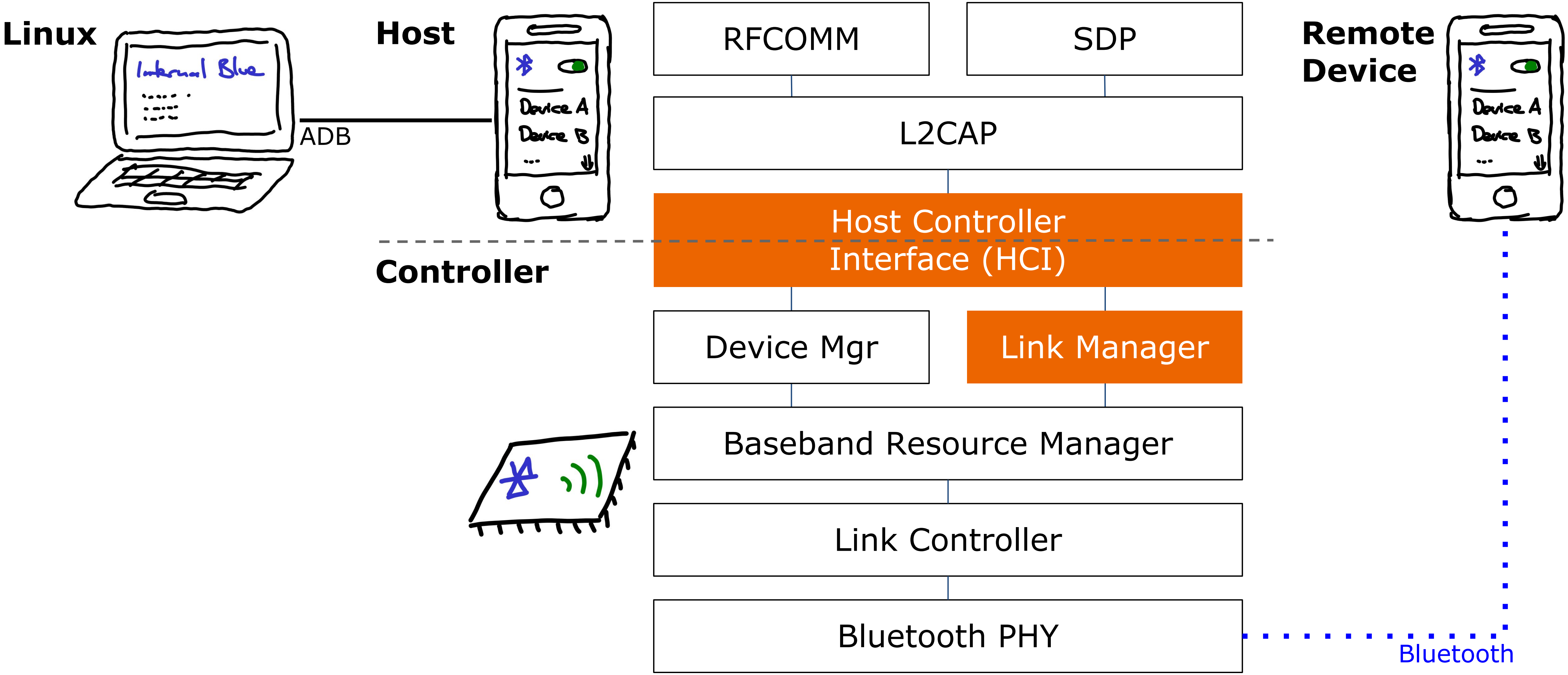
Motivation

Reverse engineering Bluetooth firmware - why?!

- Dissecting firmware gives interesting insights on a **security** perspective.
- Modifying firmware allows to have a **full-featured working Bluetooth** implementation and then **adding your features**…
- Attach open source to a **“closed” source** project.
- Requires background in security, code analysis, wireless signals… Not many people can do it, but many require the results.
- We like reverse engineering and already had great experiences with similar projects (e.g.: nexmon).



Platform Overview



Linux

ADB

Host

Controller

Remote Device

RFCOMM

SDP

L2CAP

Host Controller Interface (HCI)

Device Mgr

Link Manager

Baseband Resource Manager

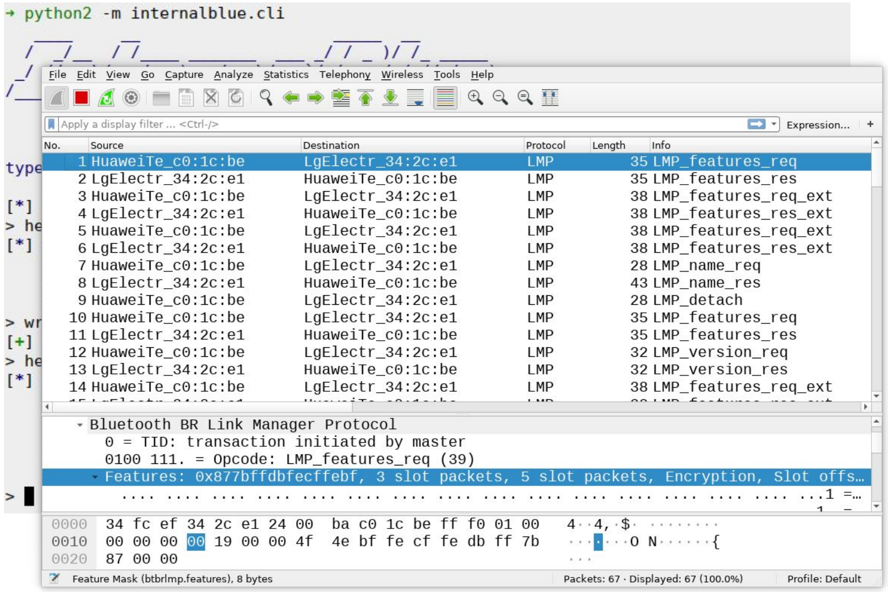
Link Controller

Bluetooth PHY

Bluetooth

Features

InternalBlue



```
→ python2 -m internalblue.cli
```

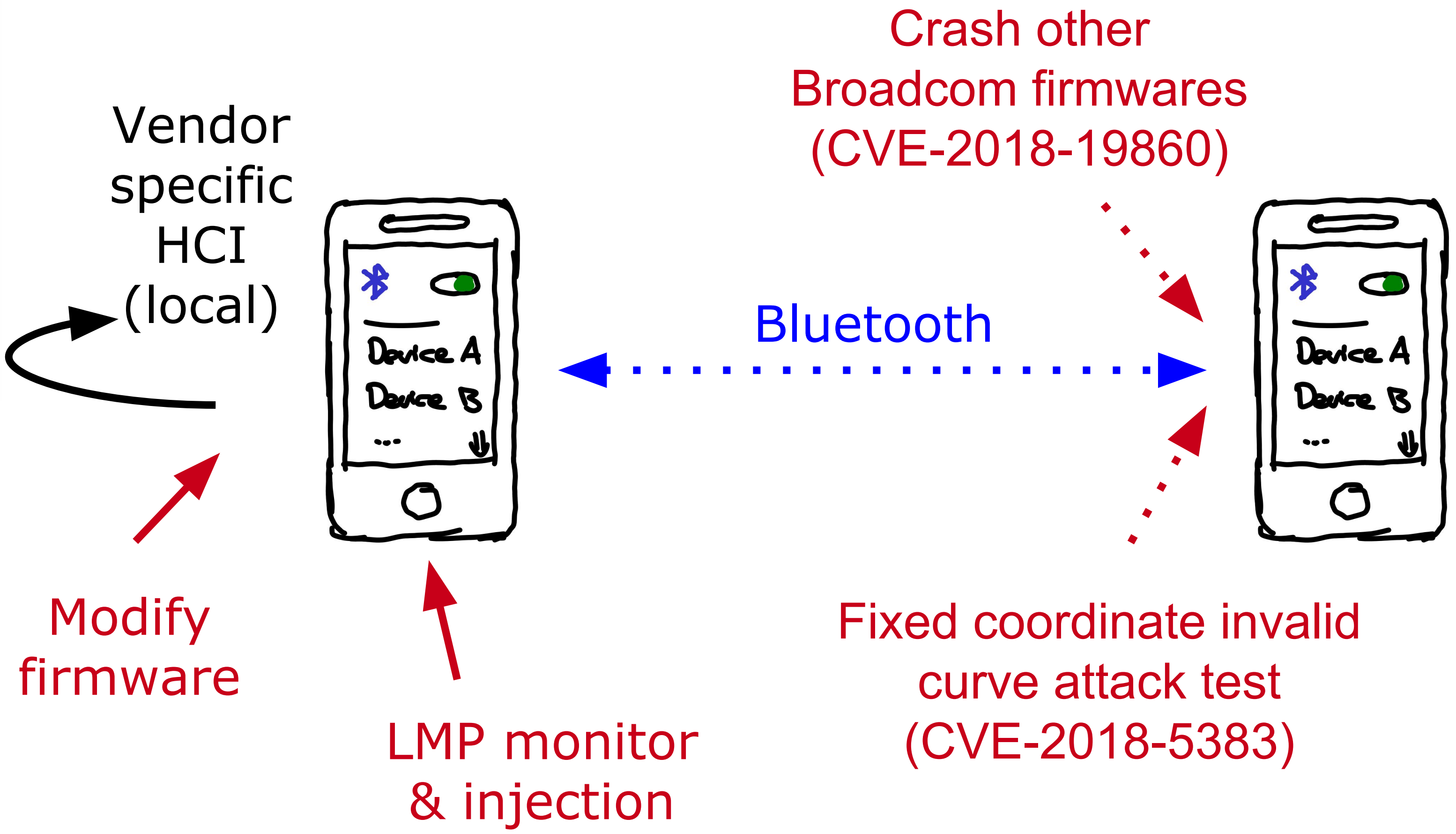
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Source               Destination          Protocol  Length Info
  1  HuaweiTe_c0:1c:be    LgElectr_34:2c:e1    LMP       35 LMP_features_req
  2  LgElectr_34:2c:e1    HuaweiTe_c0:1c:be    LMP       35 LMP_features_res
  3  HuaweiTe_c0:1c:be    LgElectr_34:2c:e1    LMP       38 LMP_features_req_ext
  4  LgElectr_34:2c:e1    HuaweiTe_c0:1c:be    LMP       38 LMP_features_res_ext
  5  HuaweiTe_c0:1c:be    LgElectr_34:2c:e1    LMP       38 LMP_features_req_ext
  6  LgElectr_34:2c:e1    HuaweiTe_c0:1c:be    LMP       38 LMP_features_res_ext
  7  HuaweiTe_c0:1c:be    LgElectr_34:2c:e1    LMP       28 LMP_name_req
  8  LgElectr_34:2c:e1    HuaweiTe_c0:1c:be    LMP       43 LMP_name_res
  9  HuaweiTe_c0:1c:be    LgElectr_34:2c:e1    LMP       28 LMP_detach
 10  HuaweiTe_c0:1c:be    LgElectr_34:2c:e1    LMP       35 LMP_features_req
 11  LgElectr_34:2c:e1    HuaweiTe_c0:1c:be    LMP       35 LMP_features_res
 12  HuaweiTe_c0:1c:be    LgElectr_34:2c:e1    LMP       32 LMP_version_req
 13  LgElectr_34:2c:e1    HuaweiTe_c0:1c:be    LMP       32 LMP_version_res
 14  HuaweiTe_c0:1c:be    LgElectr_34:2c:e1    LMP       38 LMP_features_req_ext

  Bluetooth BR Link Manager Protocol
    0 = TID: transaction initiated by master
    0100 111. = Opcode: LMP_features_req (39)
    Features: 0x877bffdbfecffebf, 3 slot packets, 5 slot packets, Encryption, Slot offs…
      .... .... .... .... .... .... .... .... .... .... .... .... .... .... .... ...1 =…

0000  34 fc ef 34 2c e1 24 00  ba c0 1c be ff f0 01 00   4··4,·$· ········
0010  00 00 00 00 19 00 00 4f  4e bf fe cf fe db ff 7b   ········O N·····{
0020  87 00 00                                           ···

Feature Mask (btbrlmp.features), 8 bytes     Packets: 67 · Displayed: 67 (100.0%)     Profile: Default
```



Crash other Broadcom firmwares (CVE-2018-19860)

Vendor specific HCI (local)

Bluetooth

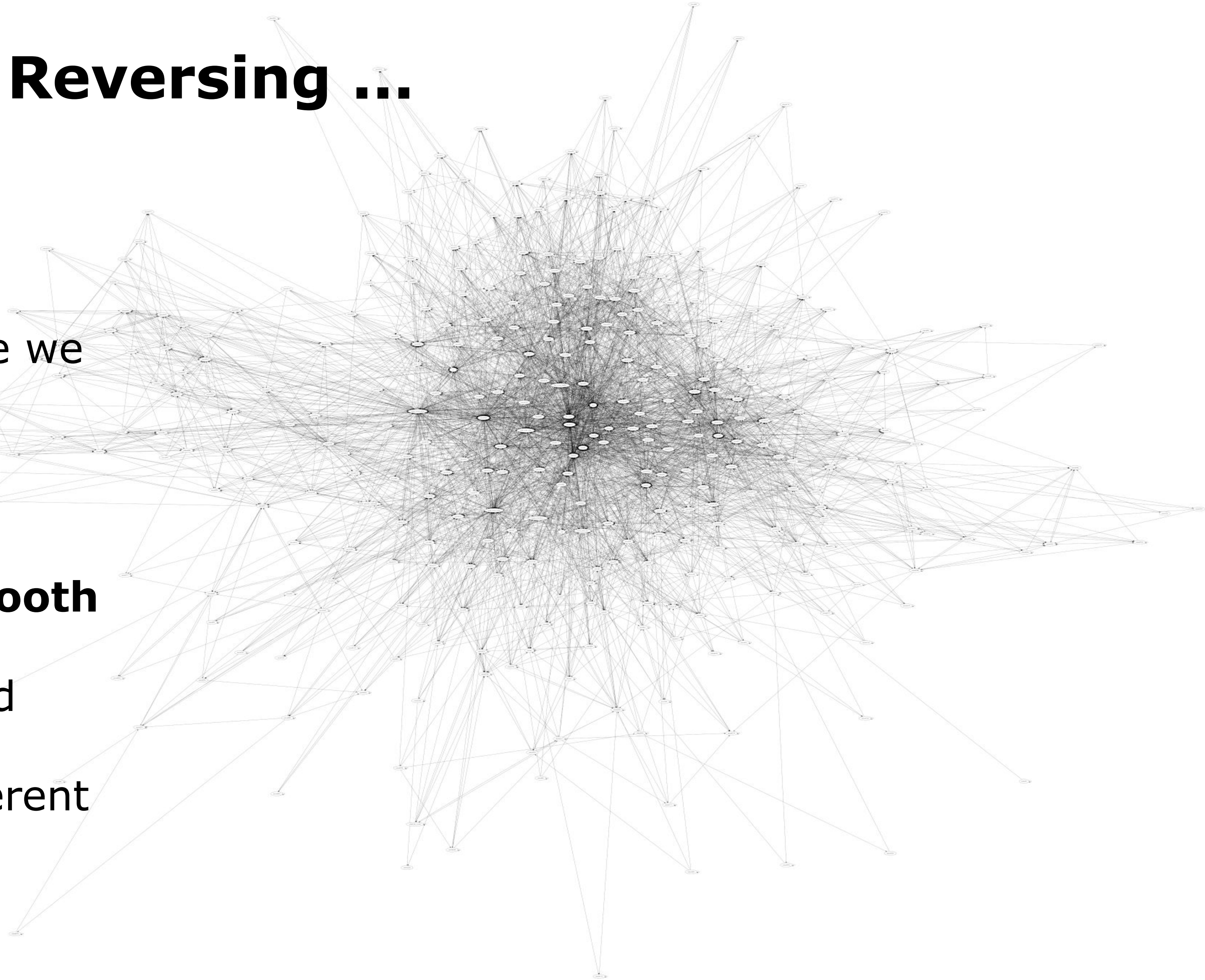
Modify firmware

LMP monitor & injection

Fixed coordinate invalid curve attack test (CVE-2018-5383)

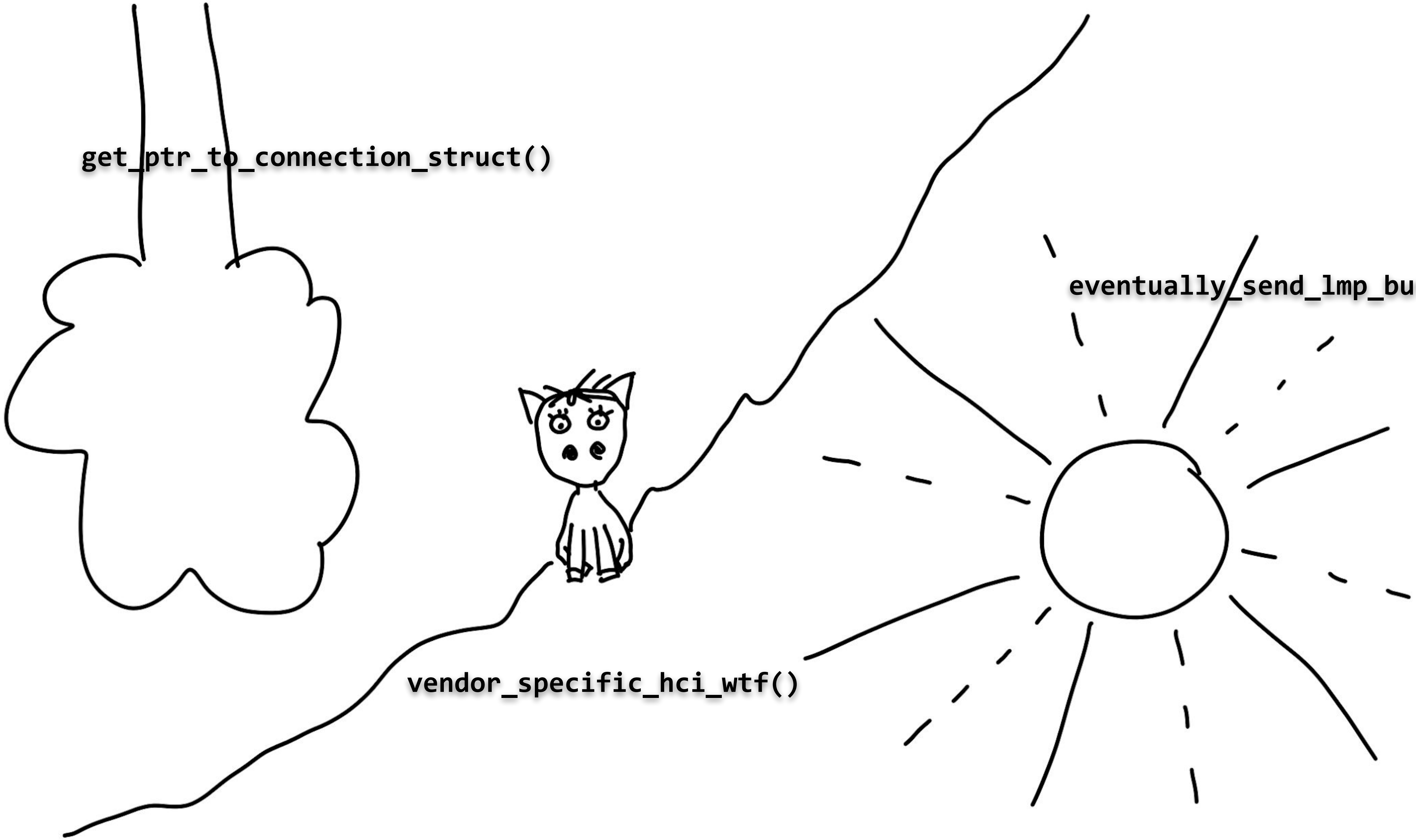
https://github.com/seemoo-lab/internalblue

Reversing ...



- Okay… maybe not that simple. Where can we patch? What are we patching? Which functions are interesting?
- Almost no strings, no function names, **no documentation except 2822 pages of Bluetooth 5.0 standard**.
- Byte sequences in the standard help locating some functions.
- Many similarities between different firmware versions :)

Reverse engineering without symbols



get_ptr_to_connection_struct()

eventually_send_lmp_buffer()

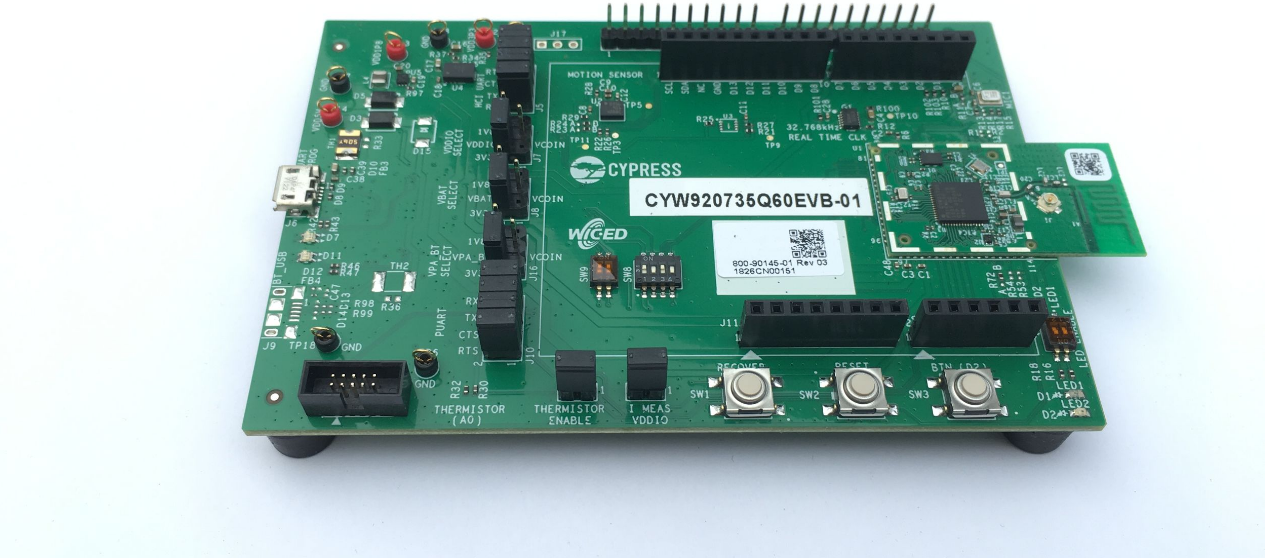
vendor_specific_hci_wtf()

Does it work on the newest device?

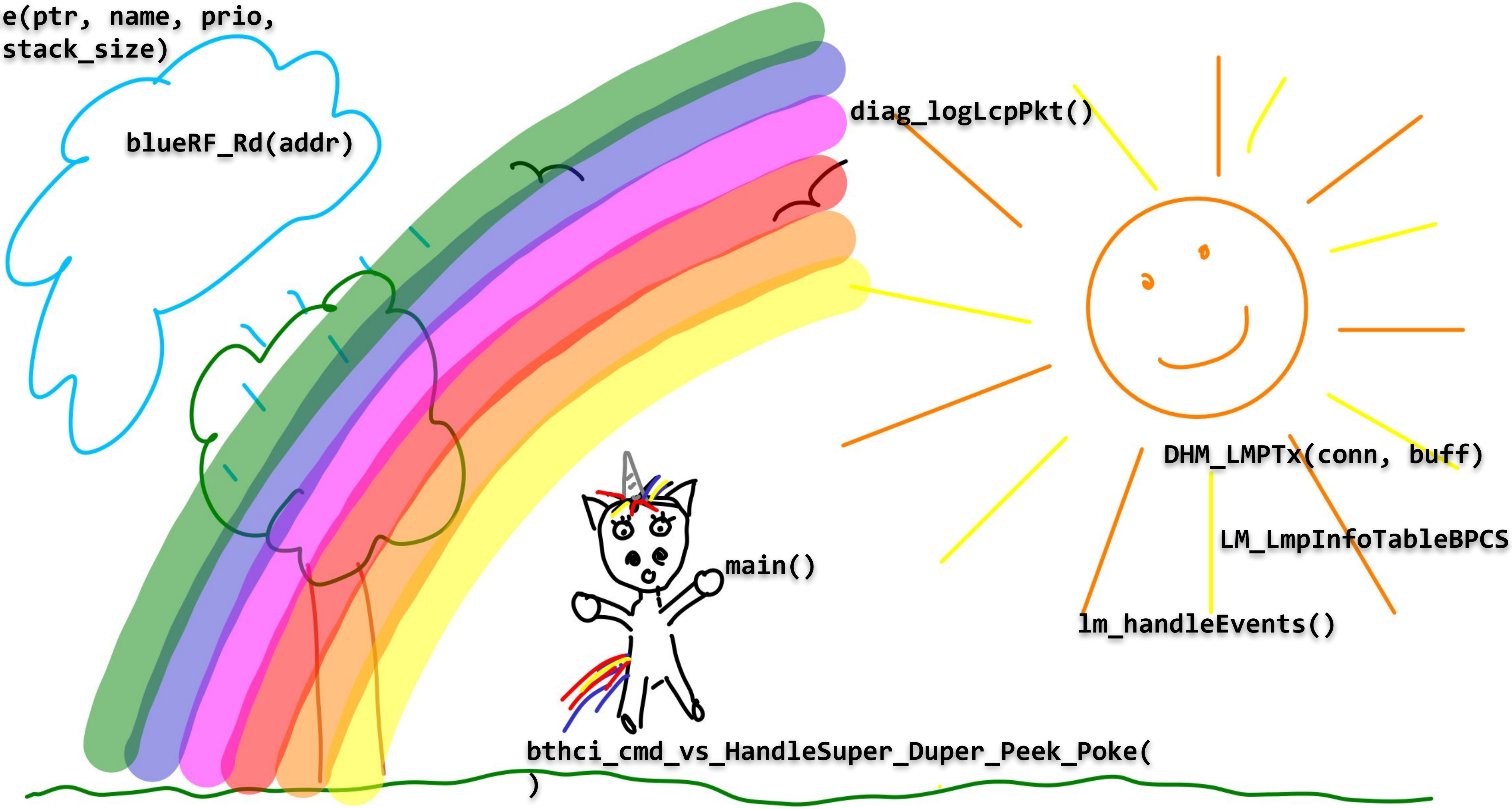
- We ported InternalBlue from **Nexus 5** to **Raspberry Pi 3/3+** and **Nexus 6P**.

- Tested on CYW20735 Bluetooth 5.0-compliant BT/BLE wireless MCU, it still has `READ_RAM`, `WRITE_RAM`, `LAUNCH_RAM` HCI commands.
  - Firmware version **January 18 2018**
- Reading out the whole firmware and applying temporarily patches without any checks in 2018, thank you ~~Broadcom~~Cypress!

- Reversing could have been faster: `patch.elf` shipped with development software contains **symbol table** for almost every firmware function…



Reverse engineering with symbols



thread_Create(ptr, name, prio, func, 0, 0, stack_size)

blueRF_Rd(addr)

diag_logLcpPkt()

DHM_LMPTx(conn, buff)

LM_LmpInfoTableBPCS

main()

lm_handleEvents()

bthci_cmd_vs_HandleSuper_Duper_Peek_Poke( )

Patching firmware

- Broadcom offers vendor specific HCI commands `READ_RAM`, `WRITE_RAM`, `LAUNCH_RAM`.
- `.hcd`-files shipped with the driver also use these commands to apply patches to RAM and ROM.
- ROM-patching is limited to a few slots, but that’s sufficient for branches into RAM.
- Neither `.hcd`-files nor vendor specific HCI commands require signatures, authentication, etc. **Just insert your code :)**

- Currently only assembly code, but we’re **working on C support** with NexMon (work in progress on branch `bluetooth-wip`).

NexMon. https://github.com/seemoo-lab/nexmon

Adding C support with Nexmon

Hidden Broadcom Features

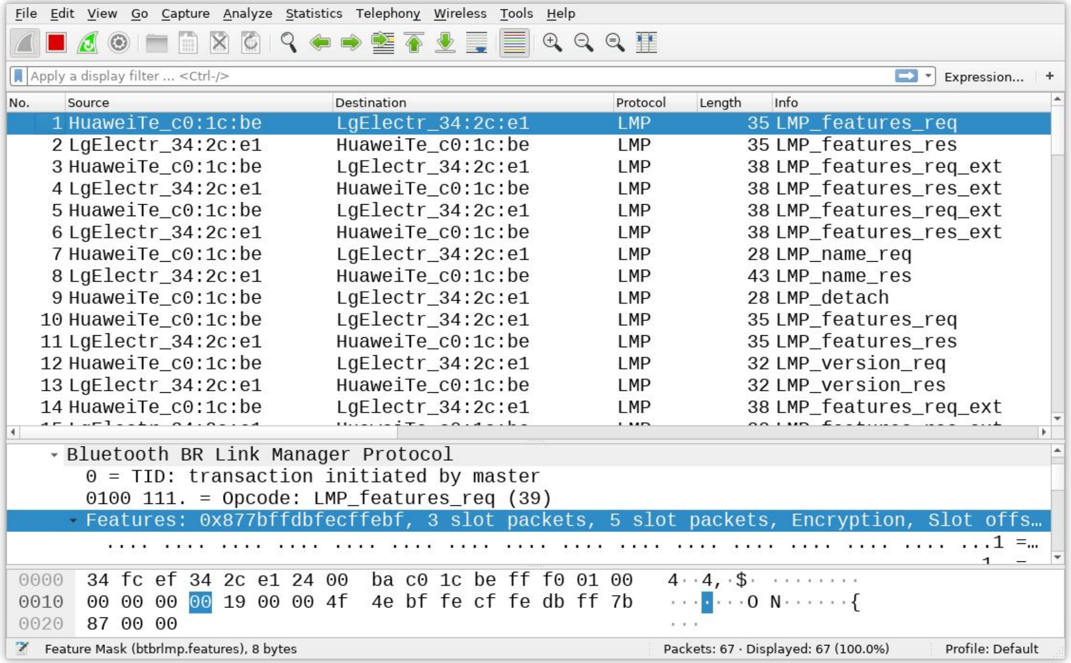
Broadcom Diagnostics Protocol

- LMP: **Link Manager Protocol**
- Located below HCI, cannot easily be sniffed as handling happens within firmware.

- Legacy version: **binary patches** for Nexus 5 and Nexus 6P to enable LMP monitoring and injection.

- HCI reversing:
  - **HCI command to send LMP packets** already included, but packets are checked for validity.

- Diagnostics protocol:
  - **Patch Android driver** to forward H4 type 0x07.
  - **LMP and LCP logging on all Brodcom chips** (at least 2008-2018).


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Expression…
+

No.	Source	Destination	Protocol	Length	Info
1	HuaweiTe_c0:1c:be	LgElectr_34:2c:e1	LMP	35	LMP_features_req
2	LgElectr_34:2c:e1	HuaweiTe_c0:1c:be	LMP	35	LMP_features_res
3	HuaweiTe_c0:1c:be	LgElectr_34:2c:e1	LMP	38	LMP_features_req_ext
4	LgElectr_34:2c:e1	HuaweiTe_c0:1c:be	LMP	38	LMP_features_res_ext
5	HuaweiTe_c0:1c:be	LgElectr_34:2c:e1	LMP	38	LMP_features_req_ext
6	LgElectr_34:2c:e1	HuaweiTe_c0:1c:be	LMP	38	LMP_features_res_ext
7	HuaweiTe_c0:1c:be	LgElectr_34:2c:e1	LMP	28	LMP_name_req
8	LgElectr_34:2c:e1	HuaweiTe_c0:1c:be	LMP	43	LMP_name_res
9	HuaweiTe_c0:1c:be	LgElectr_34:2c:e1	LMP	28	LMP_detach
10	HuaweiTe_c0:1c:be	LgElectr_34:2c:e1	LMP	35	LMP_features_req
11	LgElectr_34:2c:e1	HuaweiTe_c0:1c:be	LMP	35	LMP_features_res
12	HuaweiTe_c0:1c:be	LgElectr_34:2c:e1	LMP	32	LMP_version_req
13	LgElectr_34:2c:e1	HuaweiTe_c0:1c:be	LMP	32	LMP_version_res
14	HuaweiTe_c0:1c:be	LgElectr_34:2c:e1	LMP	38	LMP_features_req_ext

```
▾ Bluetooth BR Link Manager Protocol
    0 = TID: transaction initiated by master
    0100 111. = Opcode: LMP_features_req (39)
  ▾ Features: 0x877bffdbfecffebf, 3 slot packets, 5 slot packets, Encryption, Slot offs…
      .... .... .... .... .... .... .... .... .... .... .... .... .... .... .... ...1 =…
```

```
0000  34 fc ef 34 2c e1 24 00  ba c0 1c be ff f0 01 00   4··4,·$· ········
0010  00 00 00 00 19 00 00 4f  4e bf fe cf fe db ff 7b   ·······O N······{
0020  87 00 00                                           ···
```

Feature Mask (btbrlmp.features), 8 bytes   Packets: 67 · Displayed: 67 (100.0%)   Profile: Default

We ❤ Bluetooth

Broken by design...

Discoverability

- If Bluetooth is on, **anyone can connect to a device** - no matter if it is discoverable.
- MAC addresses can be derived by sniffing with a software-defined radio.

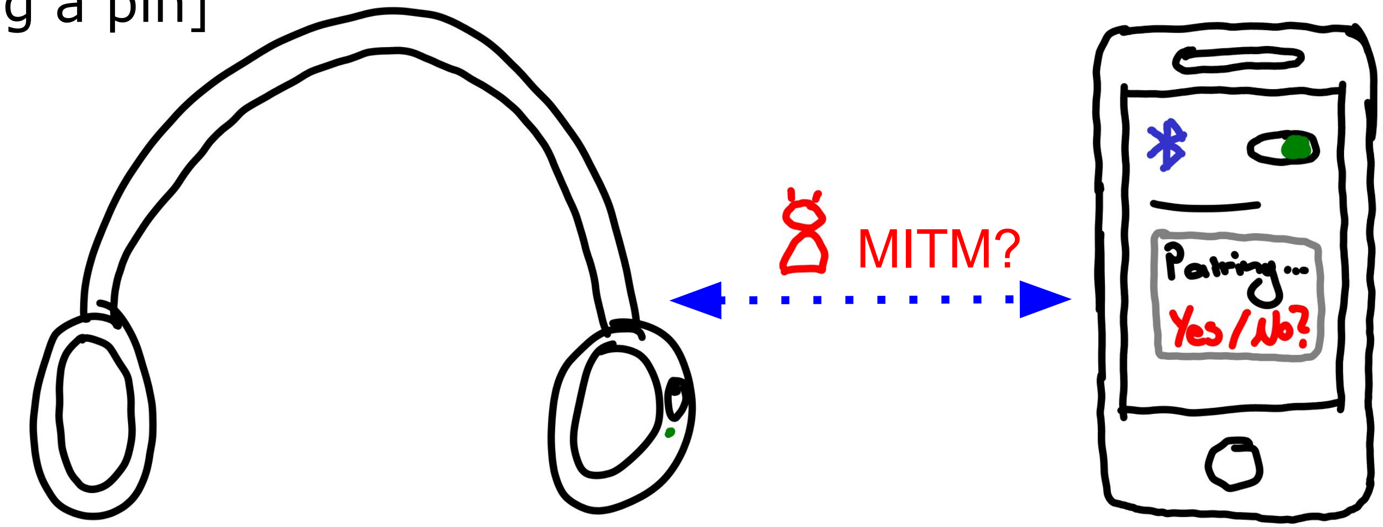
- [Demo opening connections via kown Bluetooth addresses]

Bluetooth smells like chicken. Dominic Spill, Michael Ossmann, Mark Steward. https://www.youtube.com/watch?v=qMQv1OqS-_8. 2009.

Niño

- Bluetooth 5.0 still offers **“Just Works”** pairing if a device claims to have **no input and no output**. IO capabilities are not authenticated.
- “Just Works” pairing is not secure against MITM.
- MITM can simply fake Niño and then attack “Just Works”.
- Smartphones only show a **yes/no-question** instead of warning the user: *This might be insecure pairing!*

- [Demo of other devices not showing a pin]

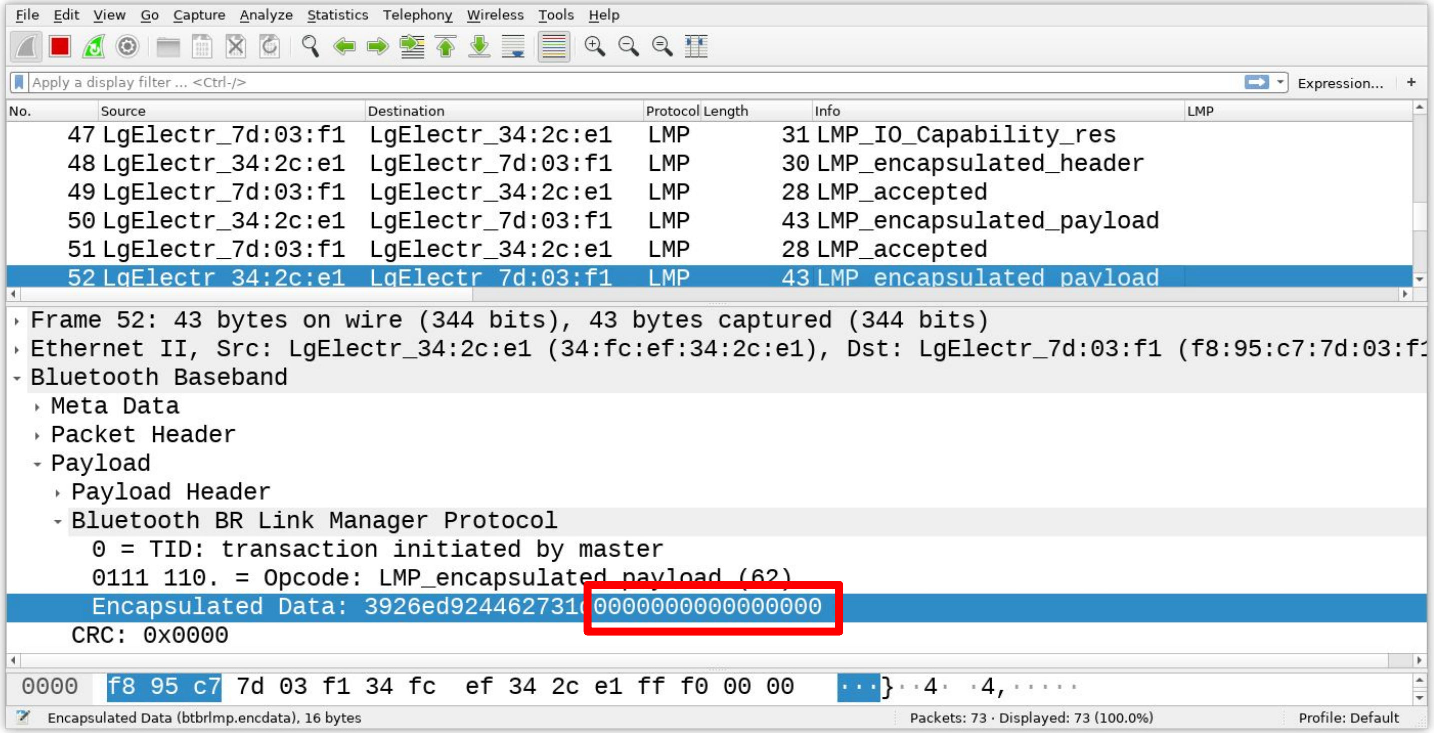


“Niño” Man-In-The-Middle Attack on Bluetooth Secure Simple Pairing. Konstantin Hypponen, Keijo M.J. Haataja. 2007.

Testing other devices for known bugs

- CVE-2018-5383 aka “Fixed-coordinate Invalid Curve Attack” (23.07.2018)
- [PoC zeroed y-coordinate in elliptic curve crypto]

  https://media.ccc.de/v/2018-154-internalblue-a-deep-dive-into-bluetooth-controller-firmware#t=1690



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
+

```
No.  Source              Destination         Protocol Length Info                          LMP
  47 LgElectr_7d:03:f1   LgElectr_34:2c:e1   LMP          31 LMP_IO_Capability_res
  48 LgElectr_34:2c:e1   LgElectr_7d:03:f1   LMP          30 LMP_encapsulated_header
  49 LgElectr_7d:03:f1   LgElectr_34:2c:e1   LMP          28 LMP_accepted
  50 LgElectr_34:2c:e1   LgElectr_7d:03:f1   LMP          43 LMP_encapsulated_payload
  51 LgElectr_7d:03:f1   LgElectr_34:2c:e1   LMP          28 LMP_accepted
  52 LgElectr_34:2c:e1   LgElectr_7d:03:f1   LMP          43 LMP_encapsulated_payload
```

```
› Frame 52: 43 bytes on wire (344 bits), 43 bytes captured (344 bits)
› Ethernet II, Src: LgElectr_34:2c:e1 (34:fc:ef:34:2c:e1), Dst: LgElectr_7d:03:f1 (f8:95:c7:7d:03:f1
⌄ Bluetooth Baseband
  › Meta Data
  › Packet Header
  ⌄ Payload
    › Payload Header
    ⌄ Bluetooth BR Link Manager Protocol
        0 = TID: transaction initiated by master
        0111 110. = Opcode: LMP_encapsulated_payload (62)
        Encapsulated Data: 3926ed924462731 0000000000000000
      CRC: 0x0000
```

```
0000  f8 95 c7 7d 03 f1 34 fc  ef 34 2c e1 ff f0 00 00   ···}··4· ·4,·····
```

Encapsulated Data (btbrlmp.encdata), 16 bytes   Packets: 73 · Displayed: 73 (100.0%)   Profile: Default

Details on this attack: http://www.cs.technion.ac.il/~biham/BT/

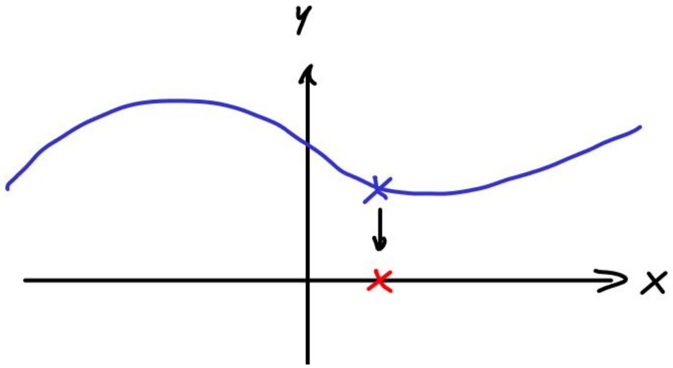
Try this at home! https://github.com/seemoo-lab/internalblue/blob/master/examples/CVE_2018_5383_Invalid_Curve_Attack_PoC.py

Fixed-coordinate Invalid Curve Attack

- Pairing uses DH Key Exchange with Elliptic Curves (ECDH)
- Public Key is a point on the curve
- The Y-coordinate of the point is not authenticated by the PIN
- MITM attacker can set the Y-coordinate to 0

  (point not on the curve anymore, ‘invalid curve’)
- Result: Both participants calculate a null-key
- Only works if both private keys (random; uniform) are even

  (25% success probability)



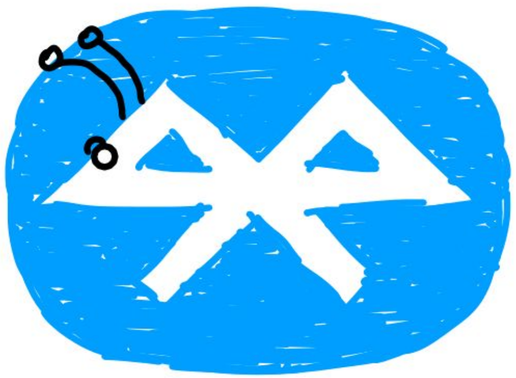
Fixed-coordinate Invalid Curve Attack

- Idea: Use InternalBlue to test other BT devices for the vulnerability
- A Patch can zero the Y-coordinates just like an attacker
- Additionally enforce the private key to be even

  (increase success rate to 50%)
- Nexus 5 itself is vulnerable: no need to bypass any checks ^^
- All devices which pair successfully with the patched Nexus 5 are vulnerable

Finding Bugs

here it is →



Our own little bug...

- Just a missing "if" somewhere. They **silently patched** it in firmware version **~summer 2014** but never shipped `.hcd`-patches for older firmwares. Long development cycles mean those devices are still around.

- Incomplete list of vulnerable devices:
  - Nexus 5
  - iPhone 5, 5s, 6
  - MacBook Pro 13" mid 2012, early 2015, 2016
  - Xperia Z3, Z5
  - Raspberry Pi 3
  - Samsung Galaxy Note 3

"does not exist"

"not standard compliant"

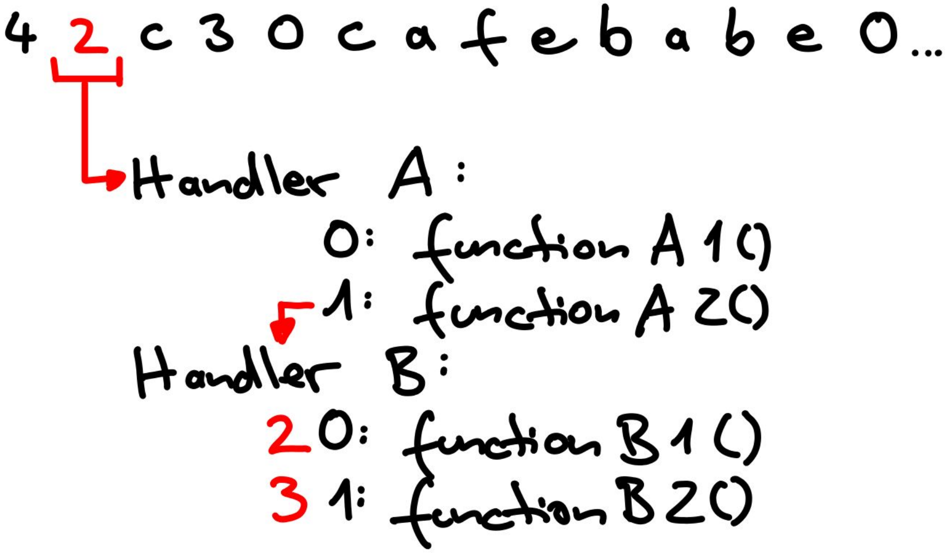
"does not affect WiFi performance"

- CVE-2018-19860 / **BT-B-g0ne**
  [Demo of remote crash]

...little bugs grow up so fast!

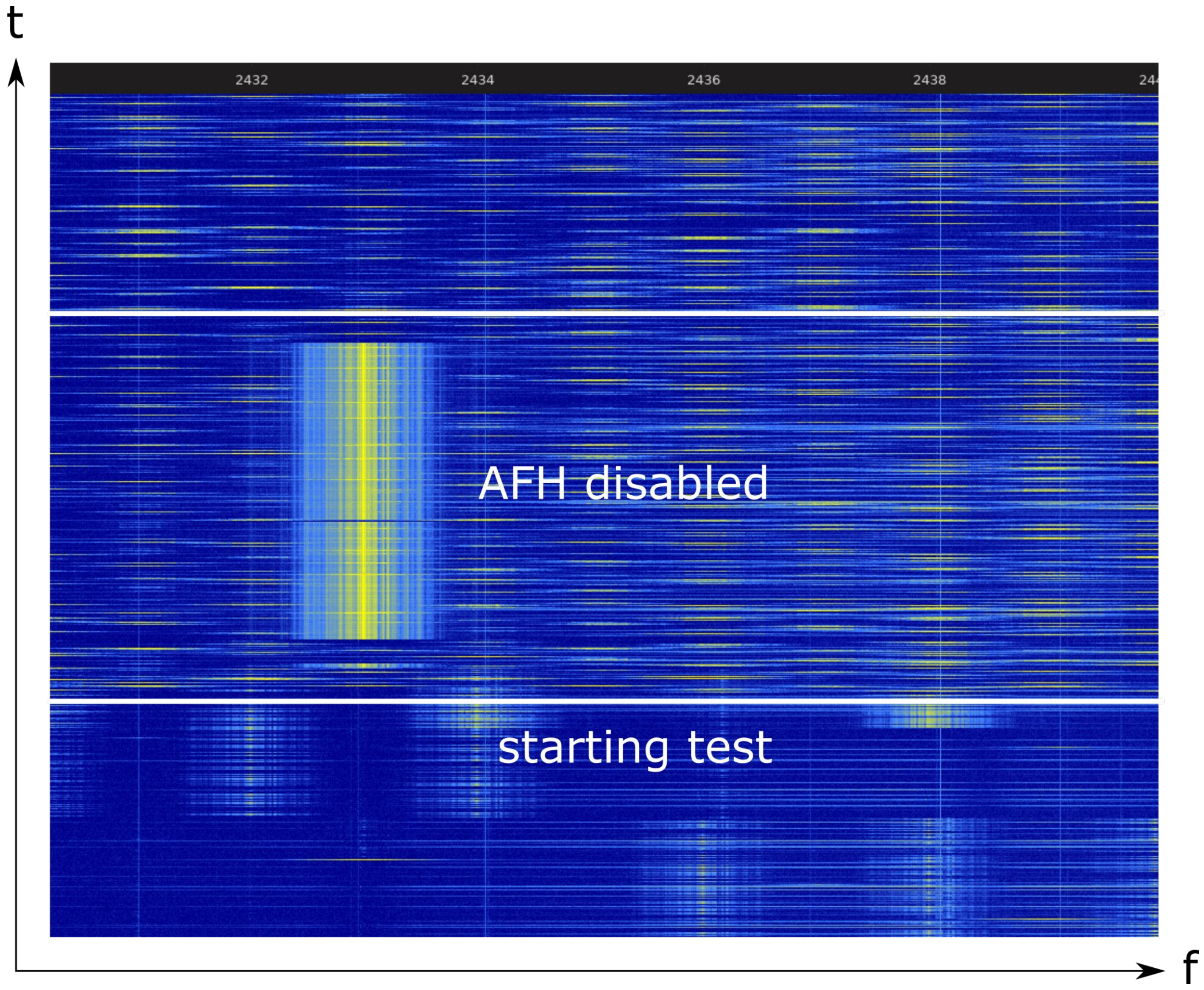
- Missing parameter check...
- **Crashes are the best case!**
- More reversing allows to **execute meaningful code**, but for each firmware version memory contents are different.
  (So far we did not find arbitrary code execution on Nexus 5.)
- On Nexus 5 we are able to execute test mode, which normally needs to be enabled locally on the host.

- CVE-2018-19860 / **BT-B-g0ne**
  [Demo of remote device under test / jamming]



Test mode execution

- Master (attacker) and remote device exchange test packets.
- Master can **disable adaptive frequency hopping** (AFH) on target device but not change its own…
- No matter if AFH is disabled or not, one can see both devices hopping on all channels during test mode.
- Works on **Nexus 5 and Xperia Z3** (BCM4339).



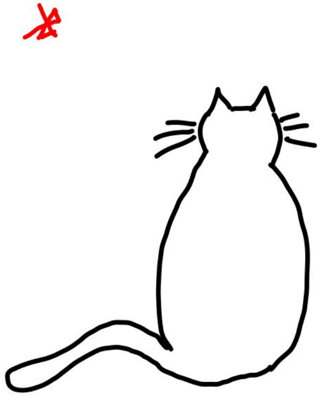
Bug finding toolchain

- Adding **tracepoints** with InternalBlue - only execute once, dump registers, stack and heap, example here is for LMP dispatcher in Nexus 5:

  ```
  tp add 0x3f3f4
  ```

- **Emulation** with Unicorn/radare2 which generates **function call sequences** and **memory diffs**. Currently only running for one function call.
- Emulation with qemu/gdb for sequences of incoming frames (work in progress).

- Whatever, it generates tons of hexadecimal stuff on that you can stare for hours.



Unicorn/radare2 emulation is a modified setup from Hugo (got it after Fitbit talk at 34C3) and Matthias Hanreich (who extended the emulator to a Fitbit fuzzer).

Fixing Bugs

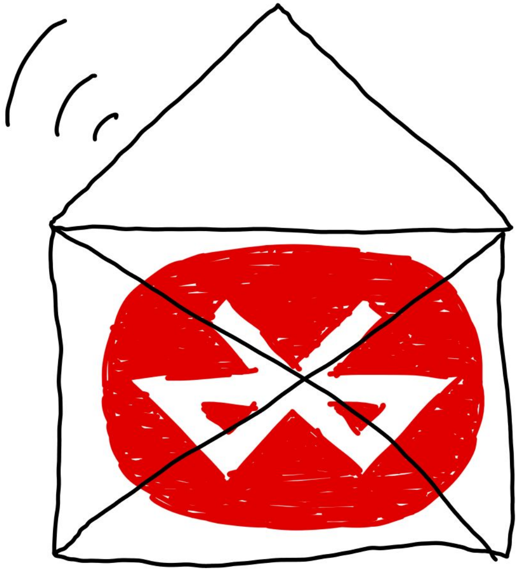
It's dead, Jim!

Bluetooth firewall

- **Actual fix**: Fix vulnerable handler. We have a `.hcd`-patch ready for Nexus 5.
  Releasing that fix would tell you which handler is vulnerable.
  Patch size is **14 bytes**…

- **Generic fix**: Apply generic **filters**, because invisible devices will reply to pings, connection establishments, etc.

  No standard compliant behavior, crashes
  Apple's bluetoothd - oops ;)



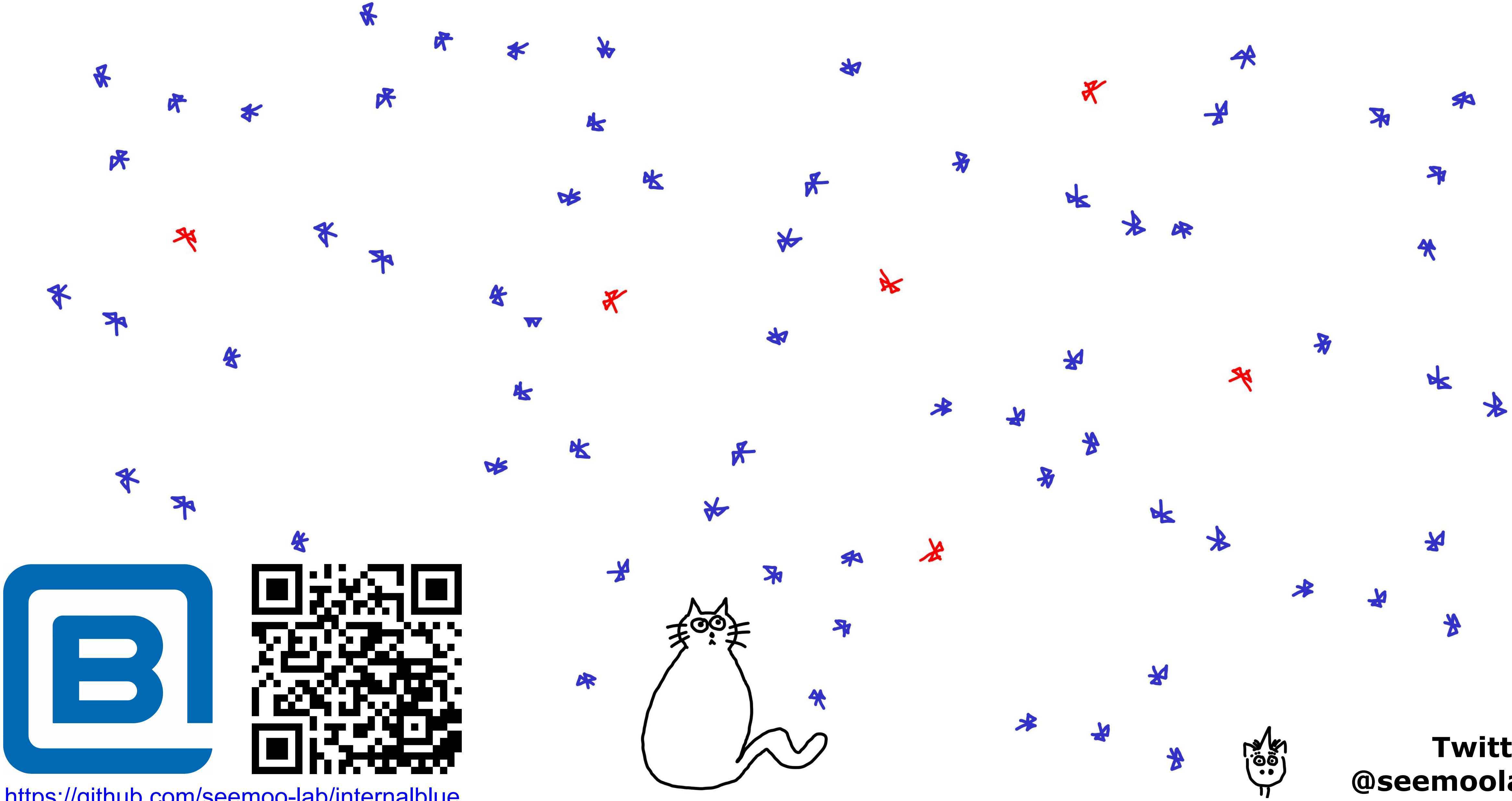
Try this at home! https://github.com/seemoo-lab/internalblue/blob/master/examples/LMP_MAC_Address_Filter.py

How long will the old bug be around?

- **Vendor fix**: vendors need to provide updated `.hcd`-files with their operating system updates.
- Some devices are **too old** to get vendor updates…
- Vendor updates will **leak the vulnerability**.

Turn off Bluetooth if your device has a Broadcom chipset and was introduced to the market before 2017.

- Long development cycles make firmware from 2014 existing in Bluetooth devices produced in 2016.
- If you have a very old chip you are not vulnerable: iPhone 4, 4s, Thinkpad T420, iMac 2009…



https://github.com/seemoo-lab/internalblue

Twitter
@seemoolab