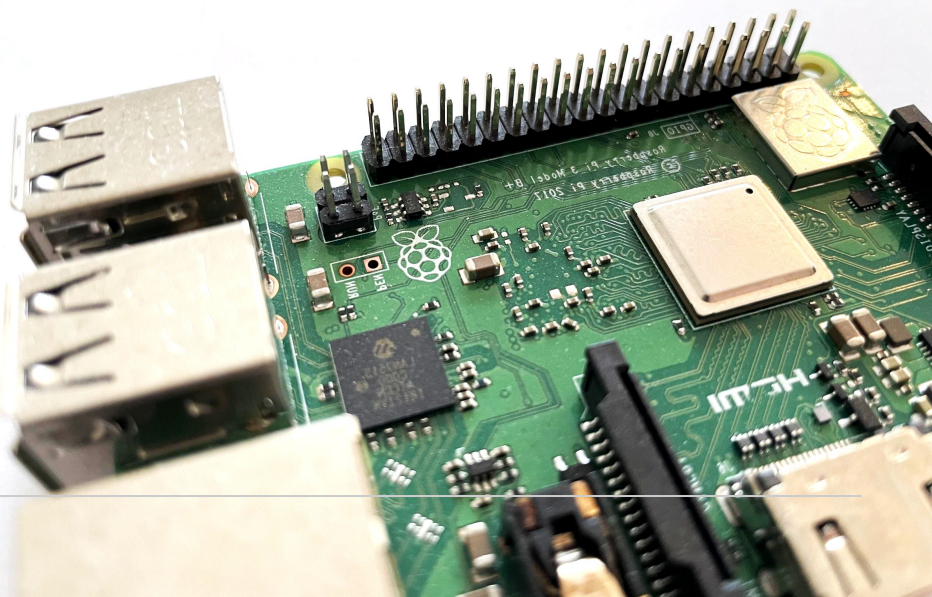
Repurposing Wireless Stacks for In-Depth Security Analysis

Jiska Classen
Secure Mobile Networking Lab - SEEMOO
TU Darmstadt, Germany

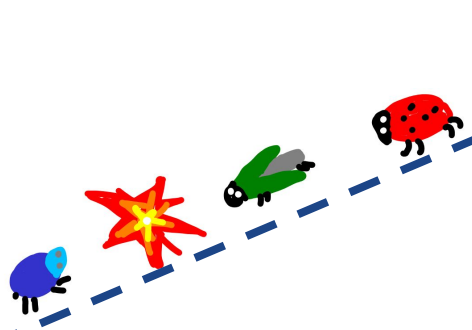
COSADE 2022 @ Leuven
April 12



SEEMOO · TECHNISCHE UNIVERSITÄT DARMSTADT · ATHENE National Research Center for Applied Cybersecurity · emergenCITY

Motivation

Wireless Security Research



Security Analysis

- Implementation Bugs
- Specification Issues

Bugs everywhere!

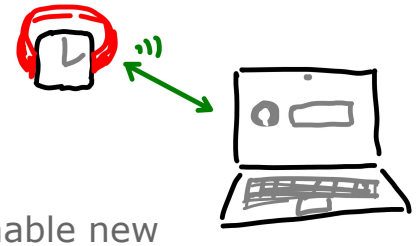
Performance Improvements

- 5G, 6G, UWB…

Battery almost empty, video is buffering...

Physical-Layer Measurements

- Distance Bounding
- Device Fingerprinting

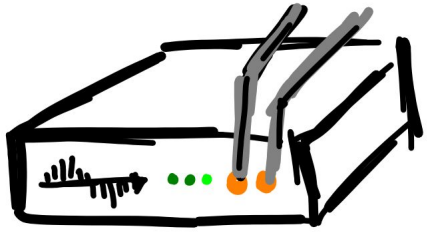


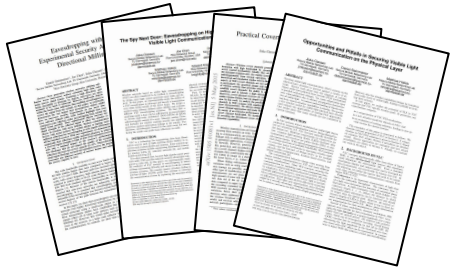
Enable new security mechanisms.

Software-Defined Radios

The ultimate wireless research tool?

- Control over every bit, even the raw signal sent over the air.
- Experiment with visible light communication and mmWaves before any consumer device is available.





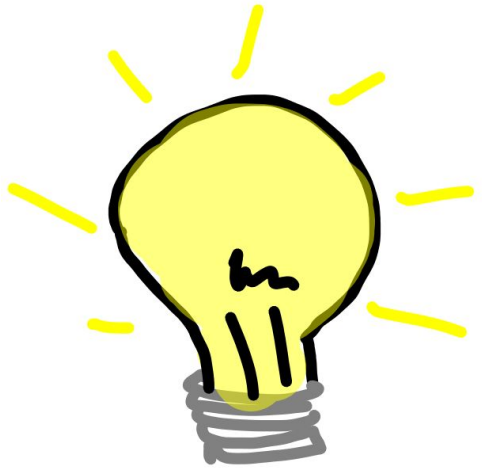
2x IEEE **CNS '15**
ACM **VLCS '15**
ACM **VLCS '16**

- Maintained, open-source protocol implementations are rare, a lot needs to be built from scratch.
  - Physical-layer only,
  - too slow for full-stack integration,
  - …
- Industry will always develop faster than a few open-source & security enthusiasts.

Staying on Track with Technology

- Collaborate with industry?
  - Non-disclosure agreements!
    → Not everything is publishable,
    other researchers cannot build upon previous results (non-public tools)
  - Research topics likely restricted…

- Build independent, open tools—but how?



Mobile Devices for Wireless Research

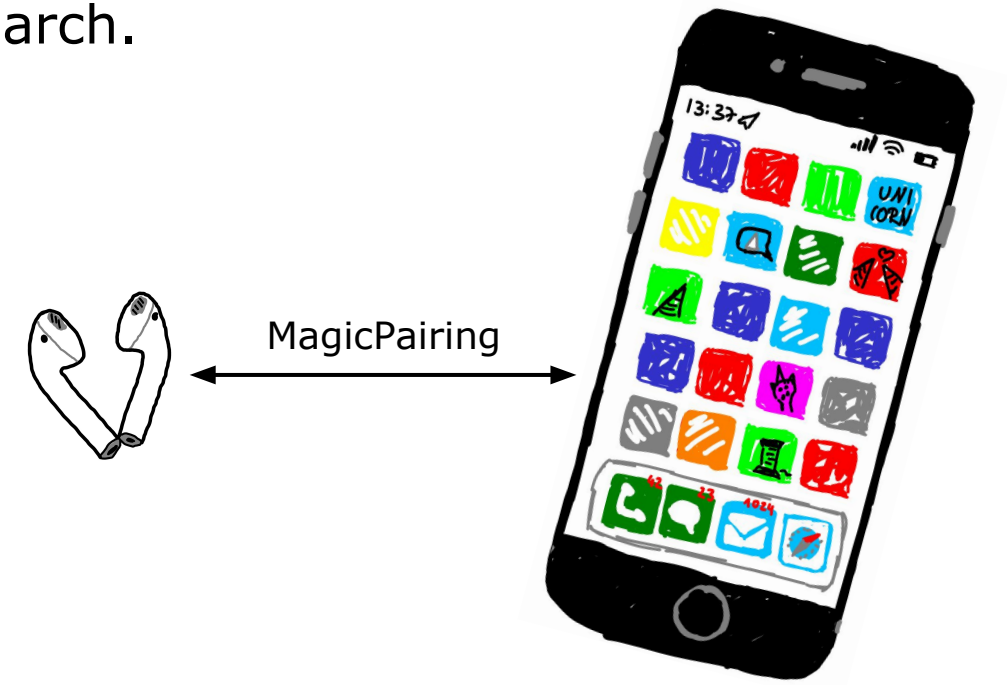
Research Proprietary Mobile Stacks

Wireless research without software-defined radios?

- Mobile devices have the most interesting stacks.
- Various vendor-specific protocols and additions.

→ Research security of mobile stacks!

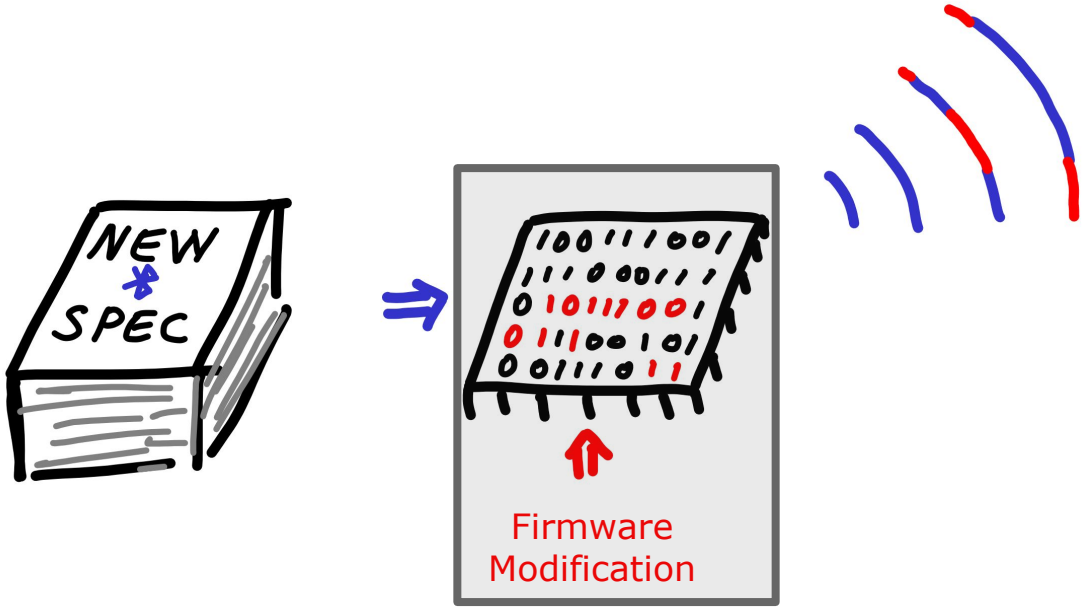
→ Repurpose these stacks for wireless research.



ACM **WiSec '20**

Repurposing Mobile Devices

Modify wireless stacks and chips of early adopters.

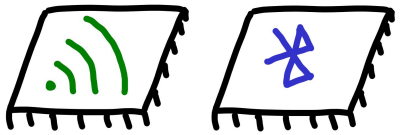


Is this possible at all?

- Smartphones are the most commonly available devices with new wireless technologies.
- Openly available evaluation kits often lag behind or miss the full-stack device integration.

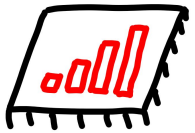
Research Framework Status

Wi-Fi & Bluetooth



- NexMon (Wi-Fi)
- InternalBlue (BT)
- Broadcom & Cypress chips
- Firmware patching support even on the latest Samsung, iPhone, MacBook, Raspberry Pi, etc.

LTE

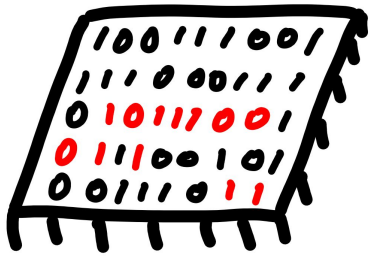


- Work in progress
- Intel chips
- Reverse-engineering of firmware and proprietary management protocols
- Recent European iPhones

Ultra-wideband (UWB)

- Work in progress
- Apple chips
- Reverse-engineering of firmware
- Modern Apple wireless ecosystem: iPhone, HomePod, Watch, …

Adding New Technologies

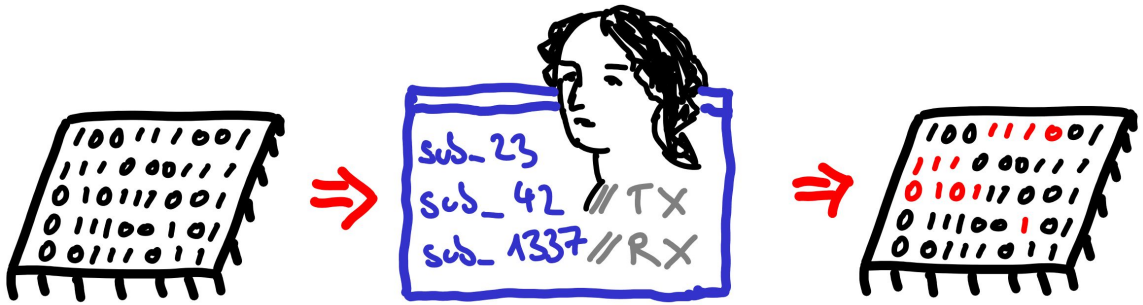


Framework Creation Process

Why do you still not support this device/chip/technology?

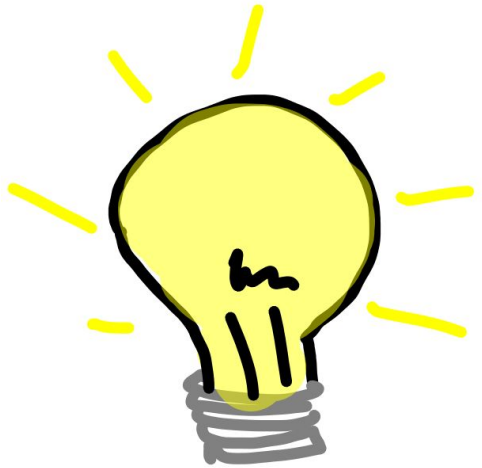
- Root/jailbreak smartphone.
- Extract wireless firmware.
- Reverse-engineer firmware.
- Analyze communication between wireless chip & iOS/Android.
- Get code execution on the chip.
- Add basic patching capabilities for C/Assembler.
- Program hooks that enable overhearing & modifying wireless traffic.

Project fails if this is impossible.

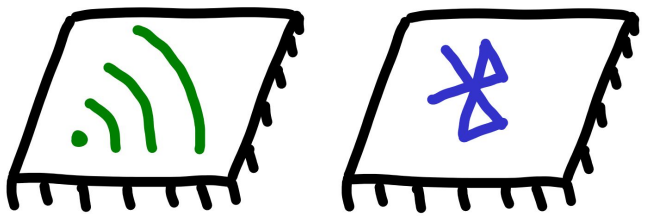


Technology & Chip Selection

- Which technologies matter?
- What exactly cannot be researched with existing tooling?
- Pick early adopters.

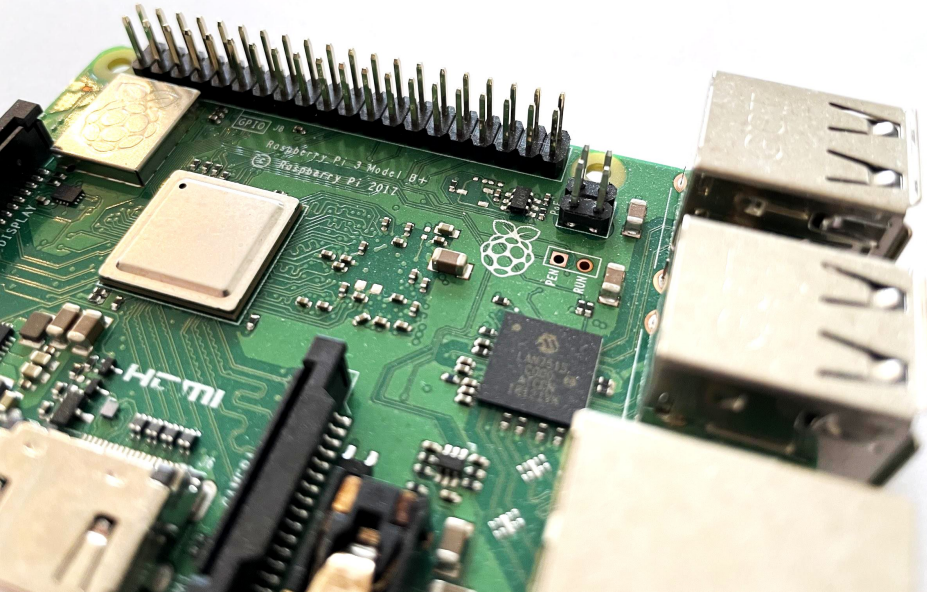


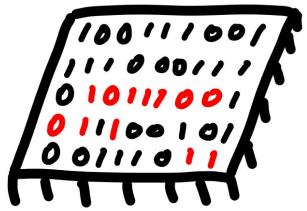
Broadcom Chips



Broadcom and Cypress Chips

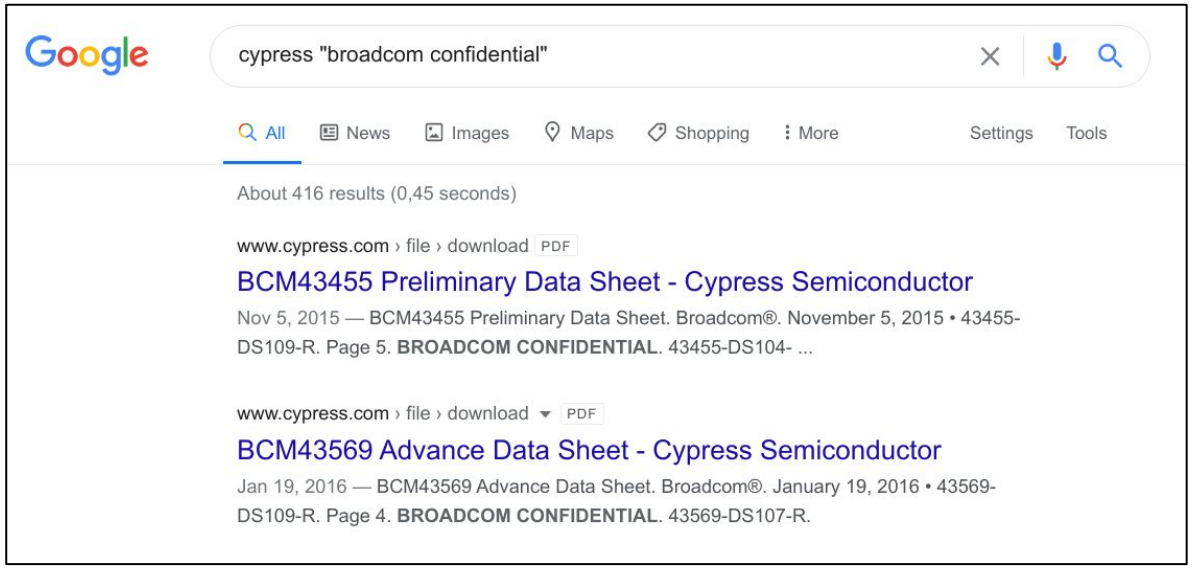
- Present in >1 billion of devices.
- Devices are popular, cheap, and easy to buy: various smartphones, evaluation kits … and the Raspberry Pi.
- Bluetooth and Wi-Fi firmware run on separate ARM cores, slightly different technologies and firmware update mechanism.
- Firmware patches are temporarily applied into RAM.
- The ROM does not verify firmware patches!





Cypress?

- Broadcom sold their wireless IoT division to Cypress in 2016.
  - IoT = small customers, too expensive to maintain their firmware, build customized chips, etc.
- Cypress published various datasheets that were Broadcom confidential.
- … and also some development tools.



nexmon

- Initially created by Matthias Schulz and Daniel Wegemer.
- Repurpose smartphones as mobile Wi-Fi sniffers in monitoring mode.
- Specifically: **Mon**itoring mode on the Google **Nex**us 5.

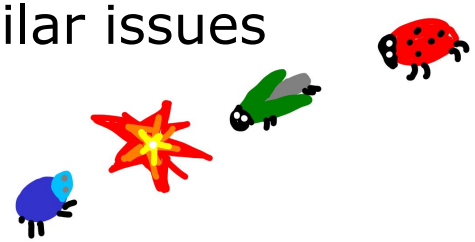
No security research focus!

- Wi-Fi sniffing and frame injection.
- Measuring channel state information on the wireless physical layer.
- Use an IQ buffer to repurpose the smartphone as 2.4GHz software-defined radio.
- Enable ARM debugging.

External Security Researchers

Published exploits, all **building upon NexMon** reverse engineering results:

- 2017: Nitay Artenstein, Broadpwn exploit
- 2017: Gal Beniamini working with Google Project Zero, full wireless exploit chain
- 2019: Hugues Anguelkov working with Quarkslab, re-exploiting similar issues



Why not do our own security research?

Bugs everywhere!

Bluetooth Framework

Ideal Platform for Security Research

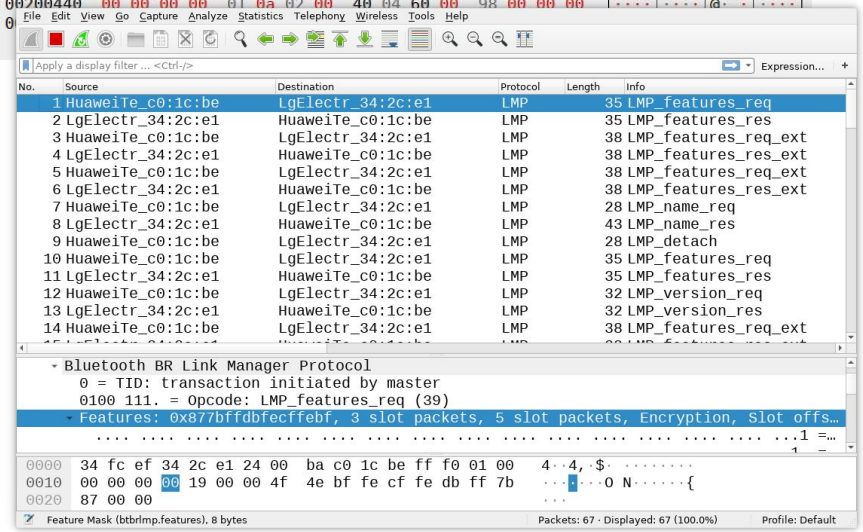
- Less focus on support of one specific chip on one specific platform.
- Support the newest chips!
  → Research requires recent security patches.
- Support all operating systems with a focus on mobile devices.
  → iOS and macOS support.

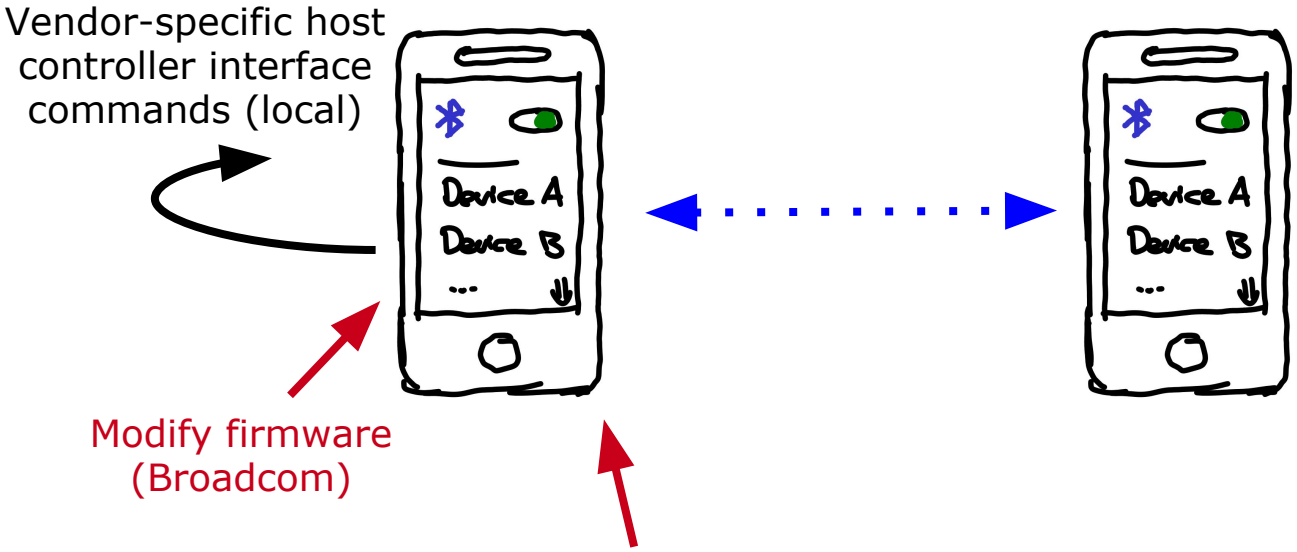
Initial InternalBlue Release

```
→ python2 -m internalblue.cli
  ____    __                  _____  __
 /  _/__  / /____ _______  ___/ / _ )/ /_ _____
_/ // _ \/ __/ -_) __/ _ \/ _  / _  / / // / -_)
/___/_//_/\__/\__/_/ /_//_/\_,_/____/_/\_,_/\__/


type <help> for usage information!

[*] Using adb device: 0cfe78fa14081c75 (Nexus 5)
> hexdump --length 0x30 0x200400
[*] 00200400  ff 1b 9d 07  00 00 00 00  09 61 44 65  63 20 31 31  |····|····|·aDe|c 11|
    00200410  20 32 30 31  32 00 18 92  fc 00 3f 1f  00 00 00 00  | 201|2···|··?·|····|
    00200420  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  |····|····|····|····|
    00200430
> writemem 0x200422 Runtime Debugging!!!
[+] Writing Memory: Written 20 bytes to 0x00200422.
> hexdump --length 0x50 0x200400
[*] 00200400  ff 1b 9d 07  00 00 00 00  09 61 44 65  63 20 31 31  |····|····|·aDe|c 11|
    00200410  20 32 30 31  32 00 18 92  fc 00 3f 1f  00 00 00 00  | 201|2···|··?·|····|
    00200420  00 00 52 75  6e 74 69 6d  65 20 44 65  62 75 67 67  |··Ru|ntim| e De|bugg|
    00200430  69 6e 67 21  21 21 00 00  00 00 00 00  00 00 00 00  |ing!|!!··|····|····|
    00200440  00 00 00 00  01 0a 02 00  40 04 60 00  98 00 00 00  |····|····|@·`·|····|
> ▉
```





ACM **WiSec '19**
ACM **MobiSys '19**

InternalBlue Now

- Monitoring and modification of management traffic.
- Firmware manipulation during runtime.
- Additional projects for firmware diffing, patching in C, emulation, etc.
- … and it runs on many devices!

Android 6–11
Samsung Galaxy S series +
Google Nexus series

iOS 12–14
iPhone 6–13

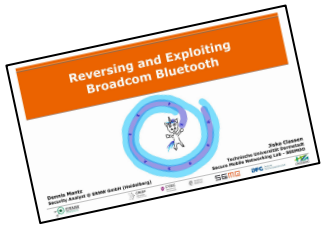
Linux (BlueZ)
Eval kits + Raspberry Pis

macOS High Sierra–Big Sur
MacBooks + iMacs

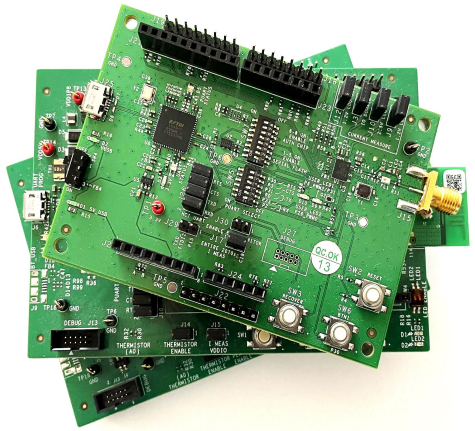
https://github.com/seemoo-lab/internalblue

Leaked Symbols

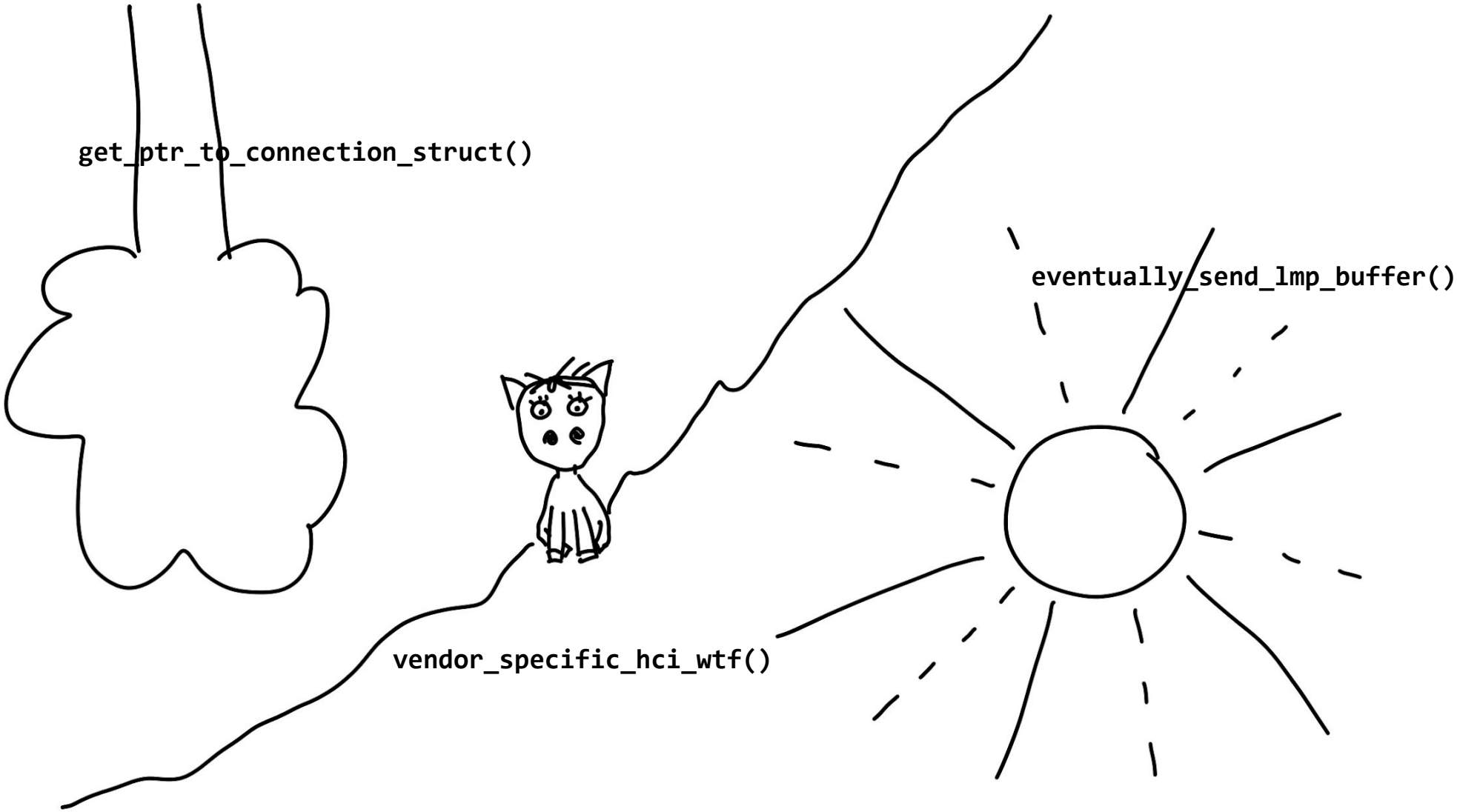
- Cypress WICED Studio 6.2–6.4 accidentally included function and global variable names for a few development kits.
- One of these chips is also contained in the MacBook Pro 2016.
- Found this after manually reverse-engineering the Google Nexus 5 firmware for a couple of months…



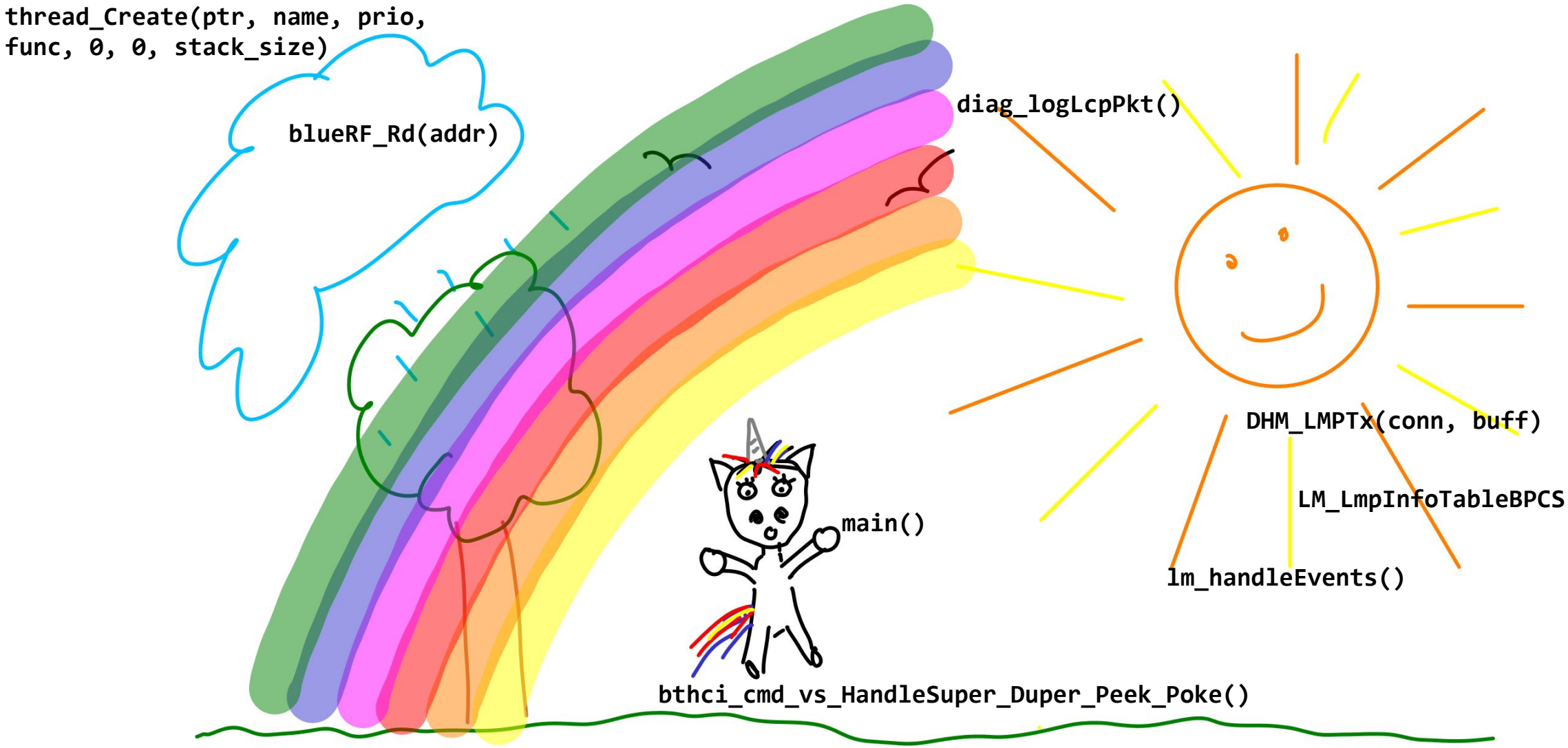
REcon ’19



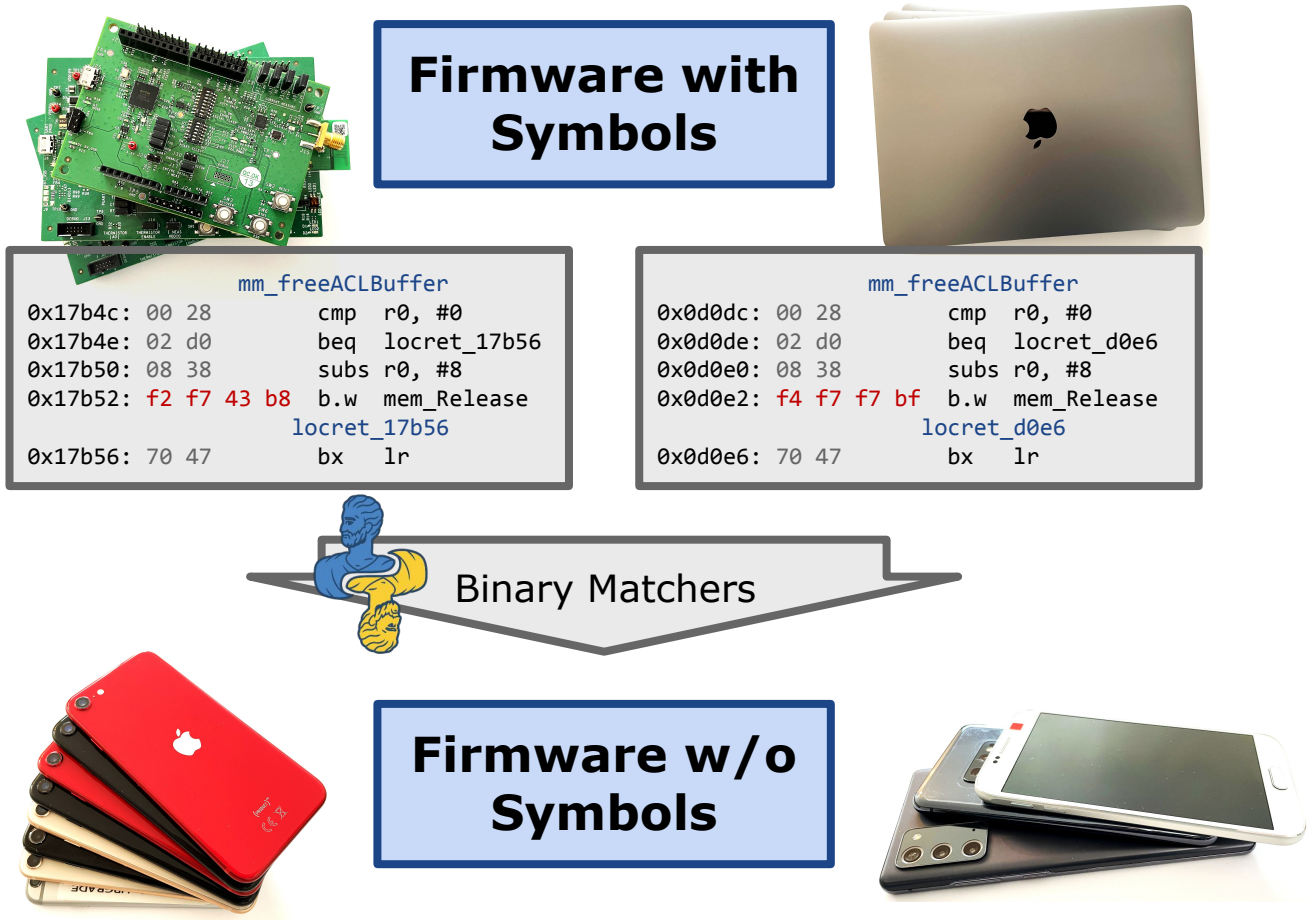
Reverse Engineering without Symbols



Reverse Engineering with Symbols



Polypyus Bindiffer

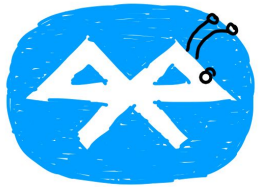


Very fast binary differ that learns from a history of binaries and applies them to other binaries within seconds.

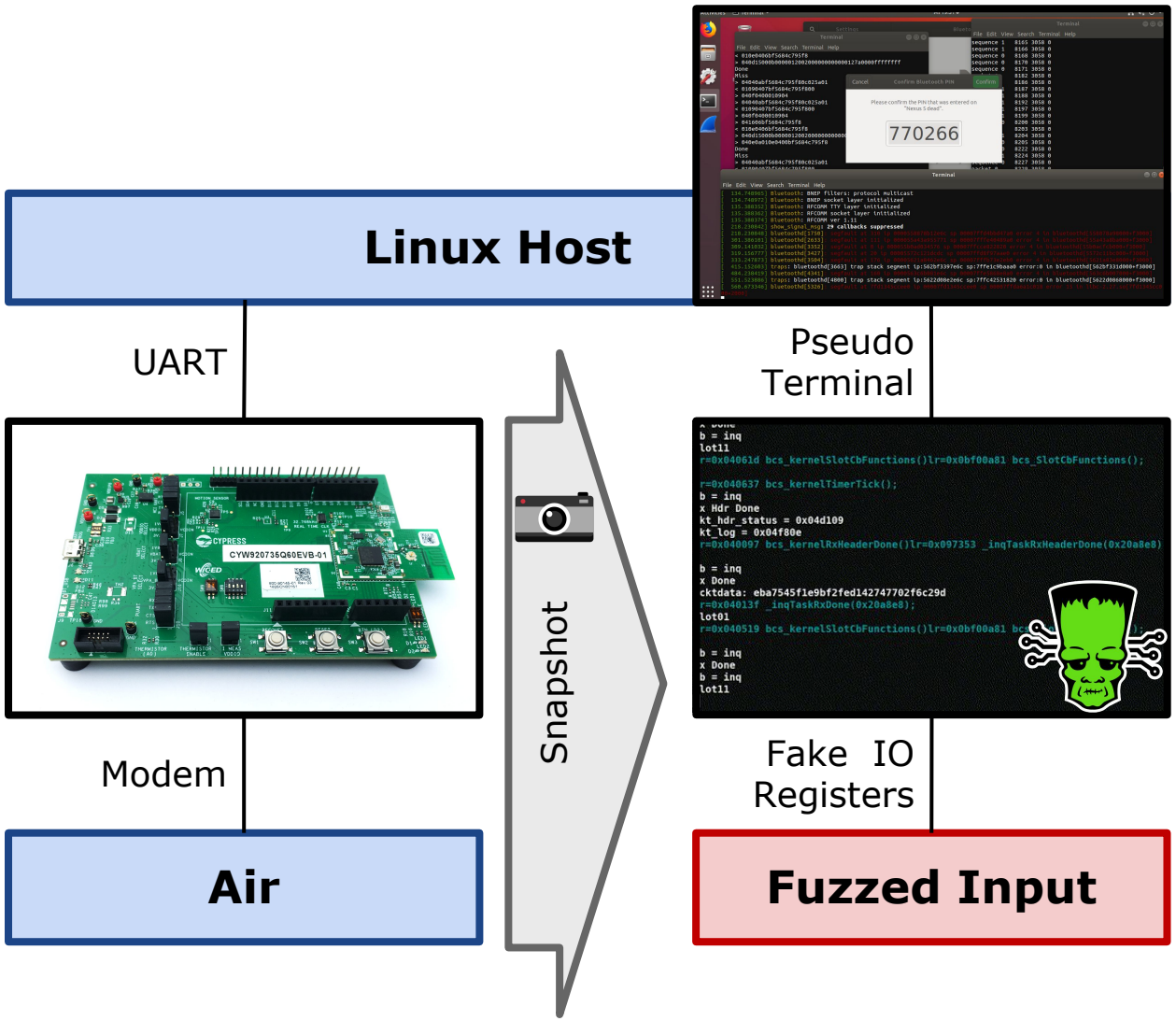
- Disassemblers (IDA Pro, Ghidra, etc.) miss ~25% of all functions in raw binaries.
- Learn history from previously reverse-engineered firmware or leaked symbols.
- Works on raw binary format.

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

Bluetooth Security



Frankenstein Emulator

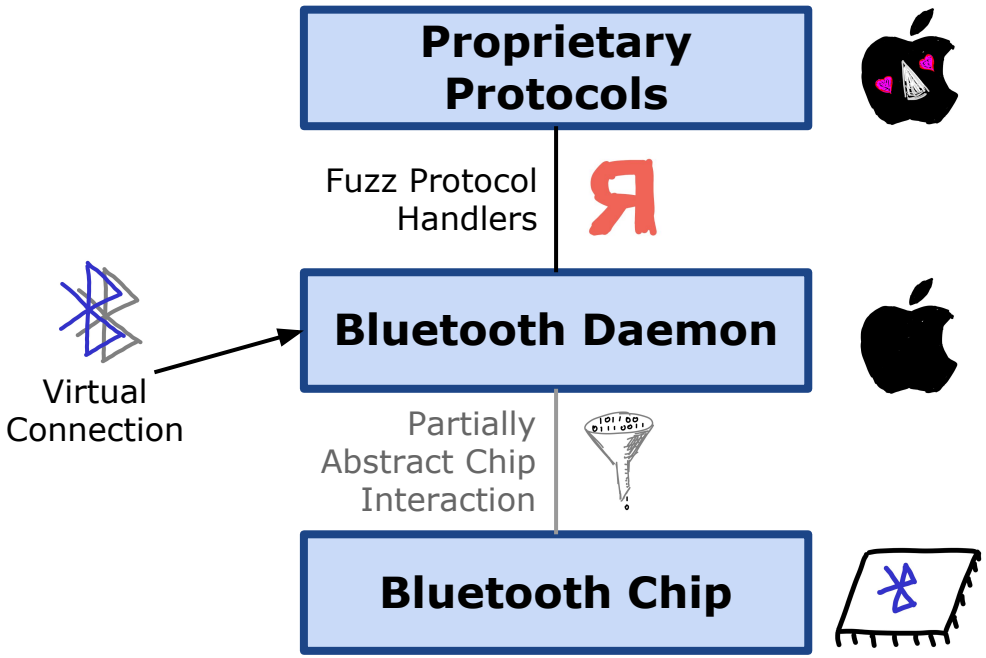


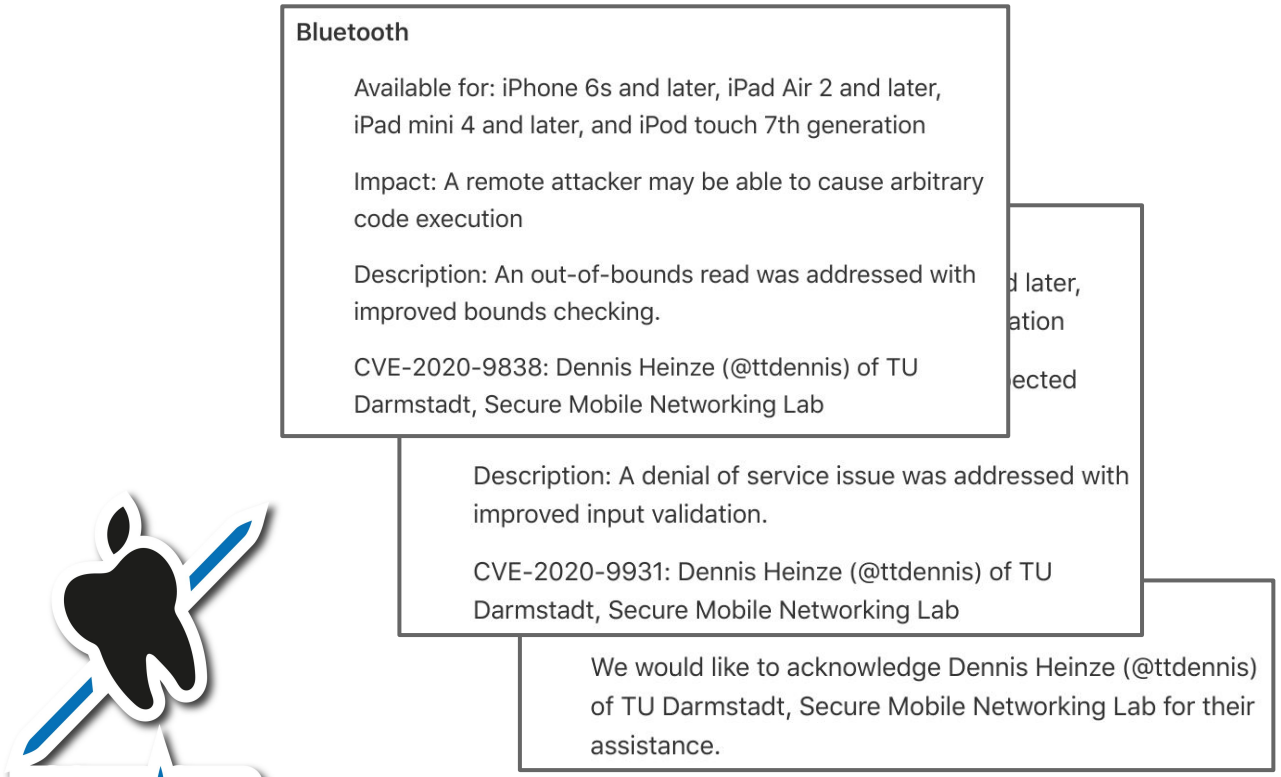
Frankenstein: Emulate Bluetooth firmware with the same speed as in hardware for realistic **full-stack fuzzing**.

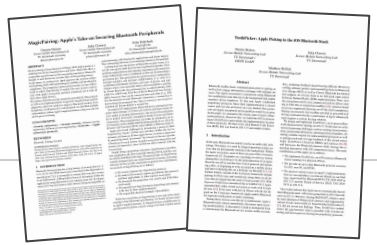
- The Linux host can run a full Bluetooth stack on a desktop setup.
- Add an xmit_state hook to the
- Bluetooth firmware function of interest, e.g., device scanning, active connection, …
- Reattach emulated snapshot with btattach, enter a similar state on the desktop, and start fuzzing.
- Identified multiple issues that can lead to remote code execution.

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

Interfaces and Protocols







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

ACM **WiSec '20**
USENIX **WOOT '20**

Hardware Vulnerability Research

Random Number Generator

- Bluetooth security (authentication, encryption) relies on secure random numbers.
- Some devices, such as the Samsung Galaxy S8, only use an insecure pseudo random number generator.

  → Security updates for Samsung devices, iPhones and MacBooks.

Bluetooth

Available for: iPhone 6s and later, iPad Air 2 and later, iPad mini 4 and later, and iPod touch 7th generation

Impact: An attacker in a privileged network position may be able to intercept Bluetooth traffic

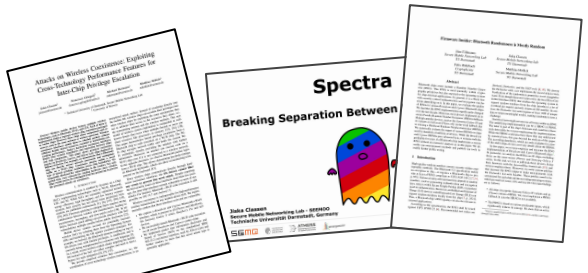
Description: An issue existed with the use of a PRNG with low entropy. This issue was addressed with improved state management.

CVE-2020-6616: Jörn Tillmanns (@matedealer) and Jiska Classen (@naehrdine) of Secure Mobile Networking Lab

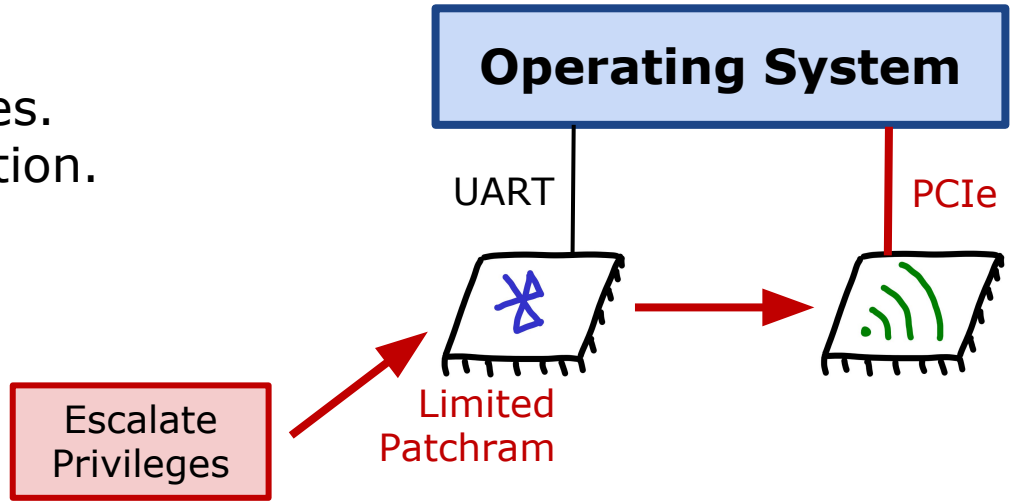
Wireless Coexistence

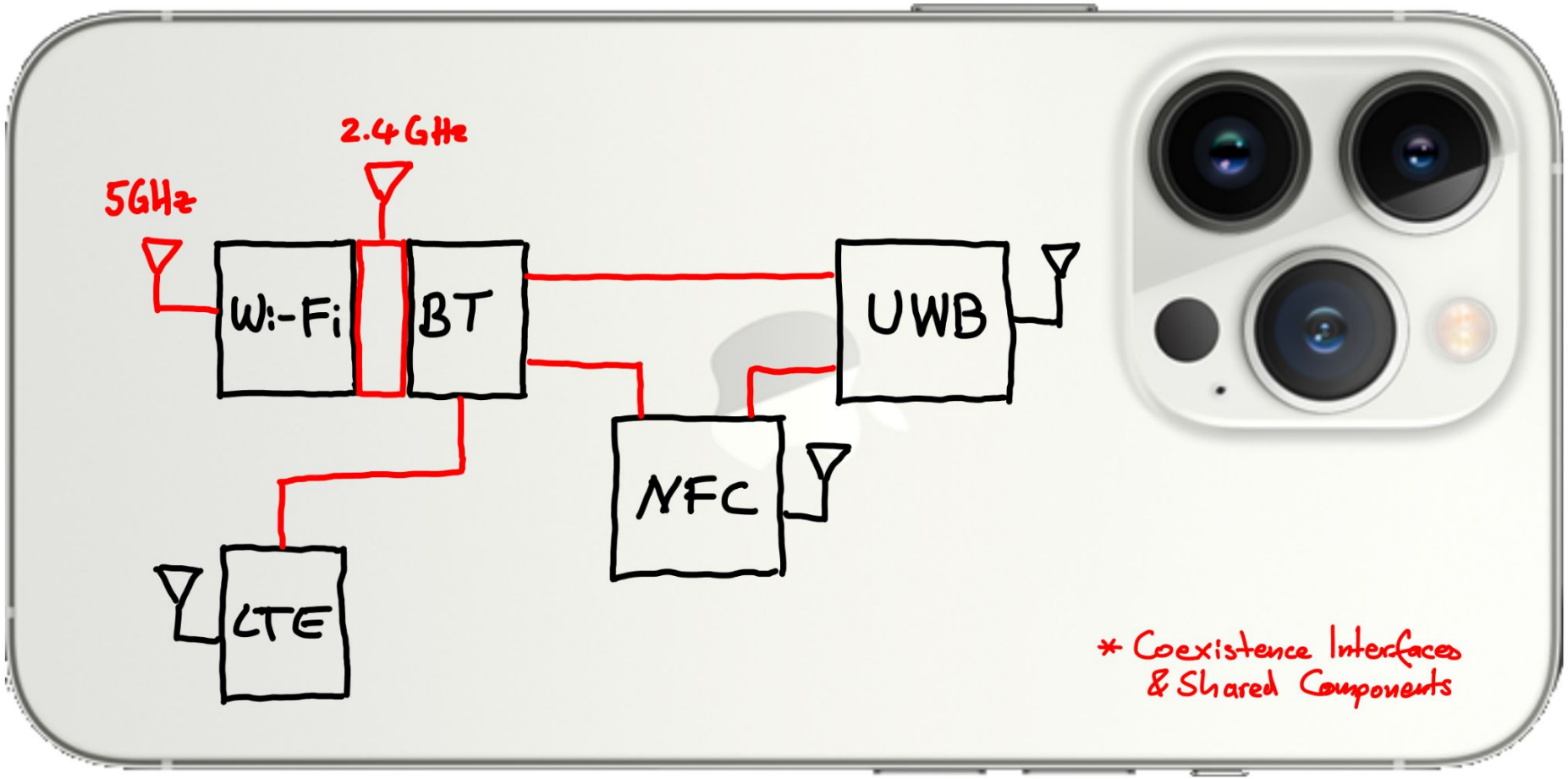
- Bluetooth and Wi-Fi run on separate ARM cores.
- Improper chip separation enables code execution.

  → Unpatchable vulnerability, no mentions.



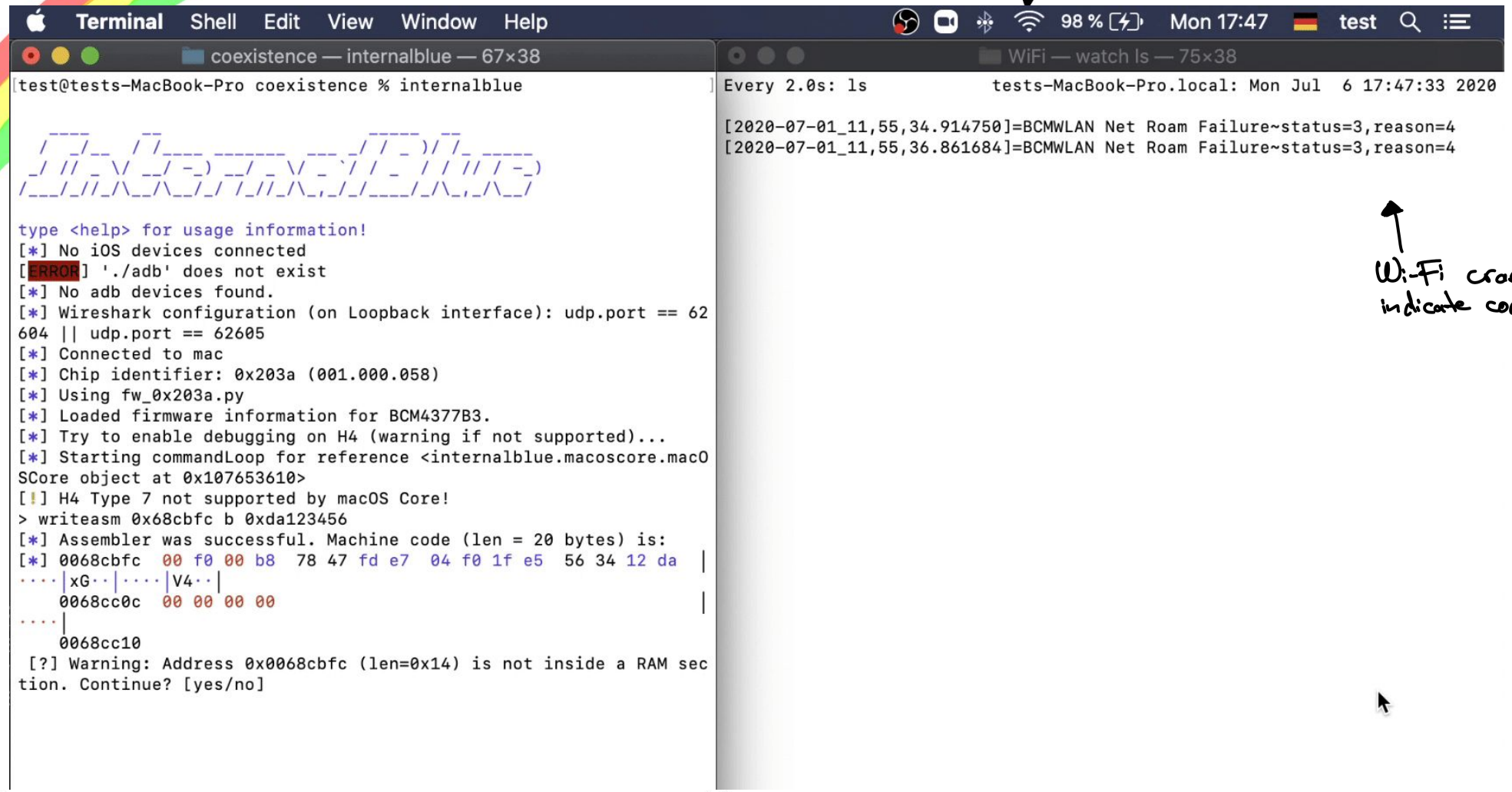
USENIX **WOOT '20**
BlackHat USA '20
S&P '22




2.4GHz
5GHz
Wi-Fi
BT
UWB
NFC
LTE
* Coexistence Interfaces
& Shared Components

Coexistence on macOS, MBP 2019/2020 (BCM4377)

OMG Wi-Fi is restarting!!!11



Execute this! →

Wi-Fi crash logs indicate code execution .^.

Memory access to Bluetooth chip...

... causing code execution in Wi-Fi!

Supporting External Researchers

- Collaboration with University of Brescia on coexistence attacks.
- Collaboration with TU Graz on Bluetooth Low Energy performance measurements.

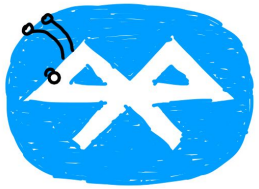
ACM **EWSN '20**

- External work building upon InternalBlue:
  - KNOB, BIAS, Method Confusion

Home | Learn About Bluetooth | Key Attributes | Bluetooth Security | Security

Bluetooth Security Notices

Vulnerability	Publication Date	Details	Specifications Affected	CVE [NVD]
Exploiting Cross-Transport Key Derivation	09/09/2020	SIG Security Notice	Core Spec, v4.2 to 5.0	CVE-2020-15802
Pairing Method Confusion	05/18/2020	SIG Security Notice	Core Spec, v2.1 to v5.2	CVE-2020-10134
Bluetooth Impersonation Attacks	05/18/2020	SIG Security Notice	Core Spec, v2.1 to v5.2	CVE-2020-10135
Key Negotiation of Bluetooth	08/13/2019	SIG Security Notice	Core Spec, v4.2, v5.0 and v5.1	CVE-2019-9506
Validation of Elliptic Curve Parameters	07/23/2018	SIG Security Notice	Core Spec, v2.1 to v5.0	CVE-2018-5383



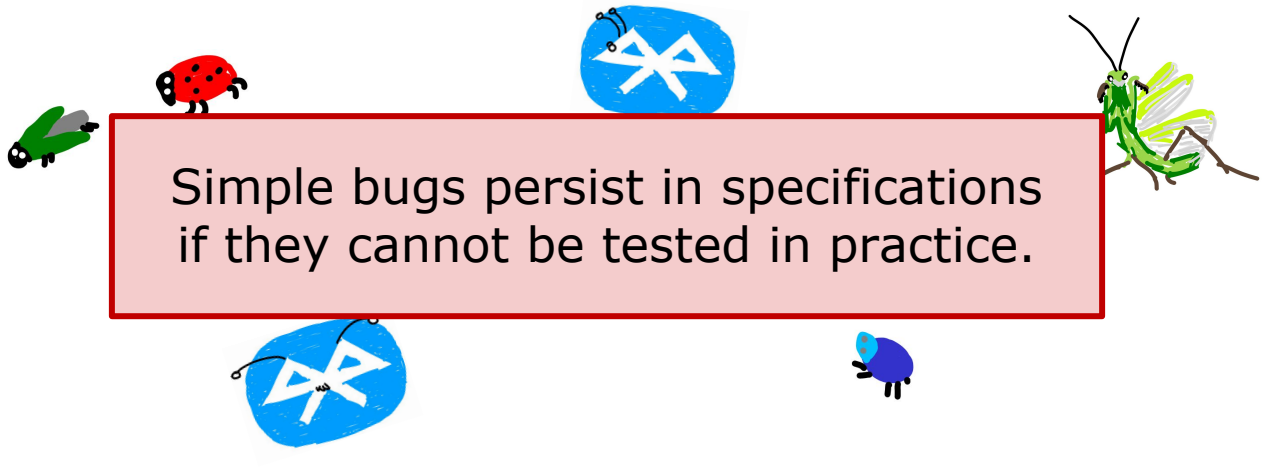
Specification-Compliant Attacks?!

- Some issues in the Bluetooth specification were so trivial that it might have been bugdoored on purpose.

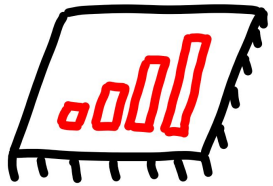
  "Only mandatory to authenticate the x coordinate but not the y coordinate of an ECDH curve point during key exchange."

  "Both parties can request to reduce the entropy of the session key from 16 bytes to 1 byte."

- Nobody could test implementations without spending 10–75k€ on an Ellisys Bluetooth analyzer.



Cellular Basebands

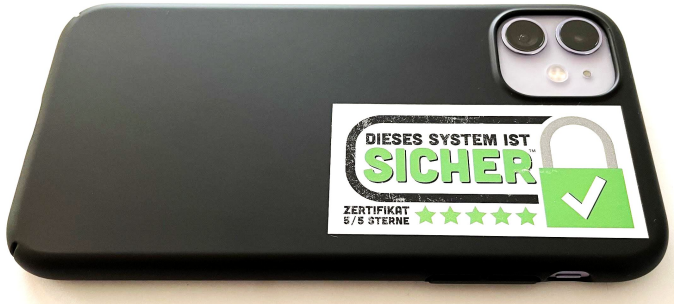


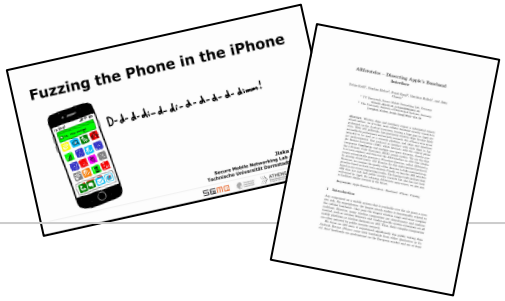
LTE & 5G

- Basic over-the-air functionality can already be tested with Osmocom and OpenAirInterface.
- Not that many security features added in 5G.

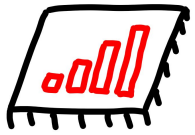
  → Different security research focus:
  - Integration into iOS/Android telephony frameworks
  - Interfaces between iOS/Android and wireless chips

- Fuzzing of the Apple-specific protocol for LTE (ARI) and the generic Qualcomm protocol for LTE+5G (QMI).
- Wireshark dissector & ARI injector open-sourced.





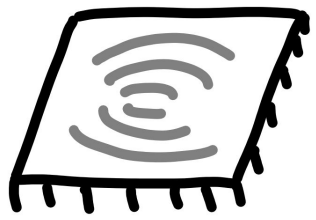
CCC **rC3** ’20
ESORICS ‘21



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


MESSAGES
You have 13 new Messages notifications
now
FaceTime
Calendar
Photos
Camera
Mail
Clock
Maps
Weather
Reminders
Notes
Stocks
Books
App Store
Podcasts
TV
1
Health
Home
Wallet
Settings
3
130
61.981

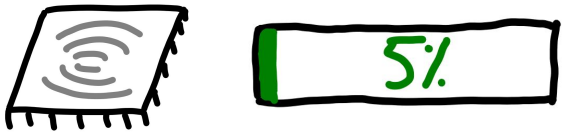
Ultra-wideband



Ultra-wideband (UWB)

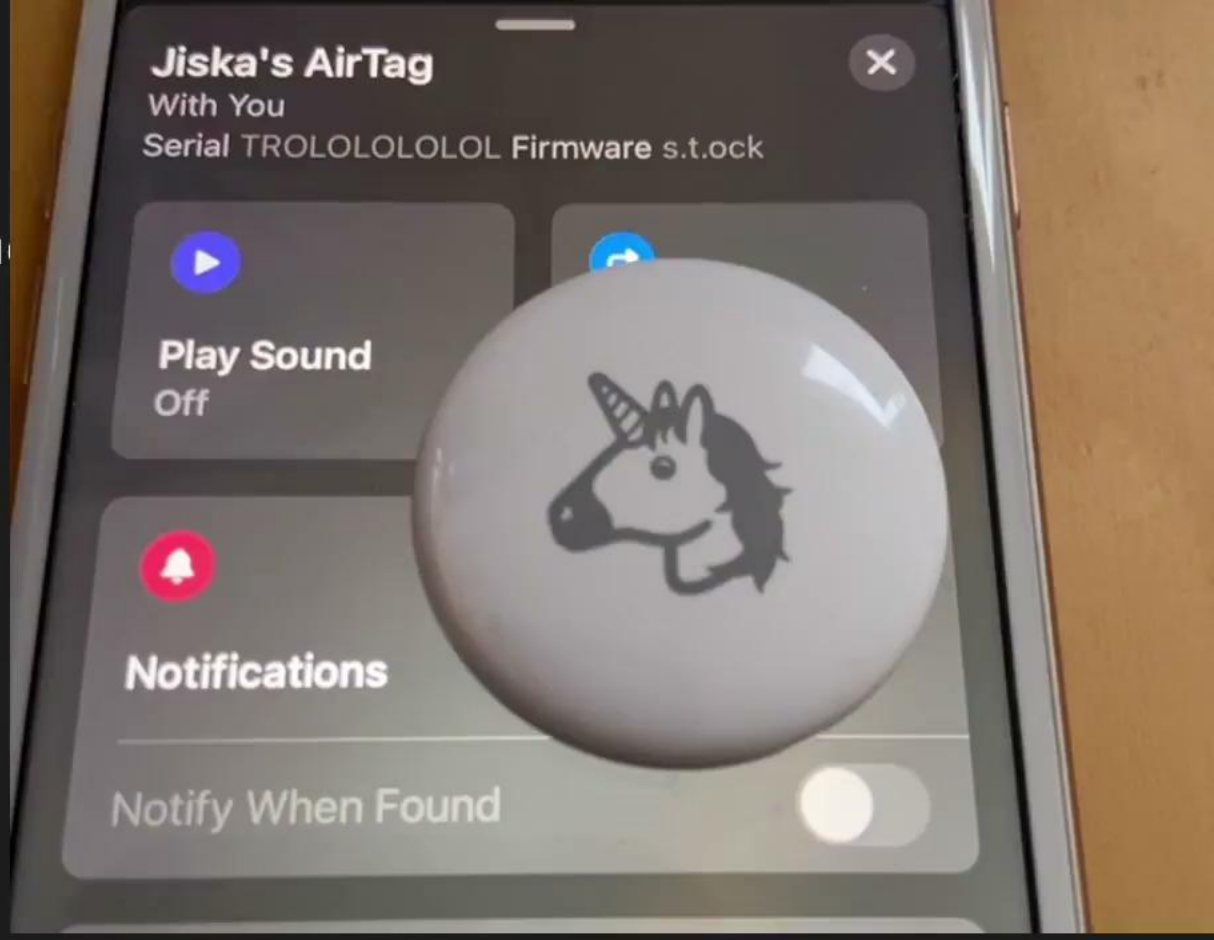
- Different frequency band than Wi-Fi and Bluetooth.
  → Highspeed, non-interfering data transmission performance!
- Supports secure ranging.
  → Built-in physical-layer security.
- A few early adopters:
  - iPhone 11+12, HomePod mini, Apple Watch 6, AirTags
  - Samsung Galaxy Note 20 5G, S21+
  - Google Pixel 6
  - Automotive for unlocking cars

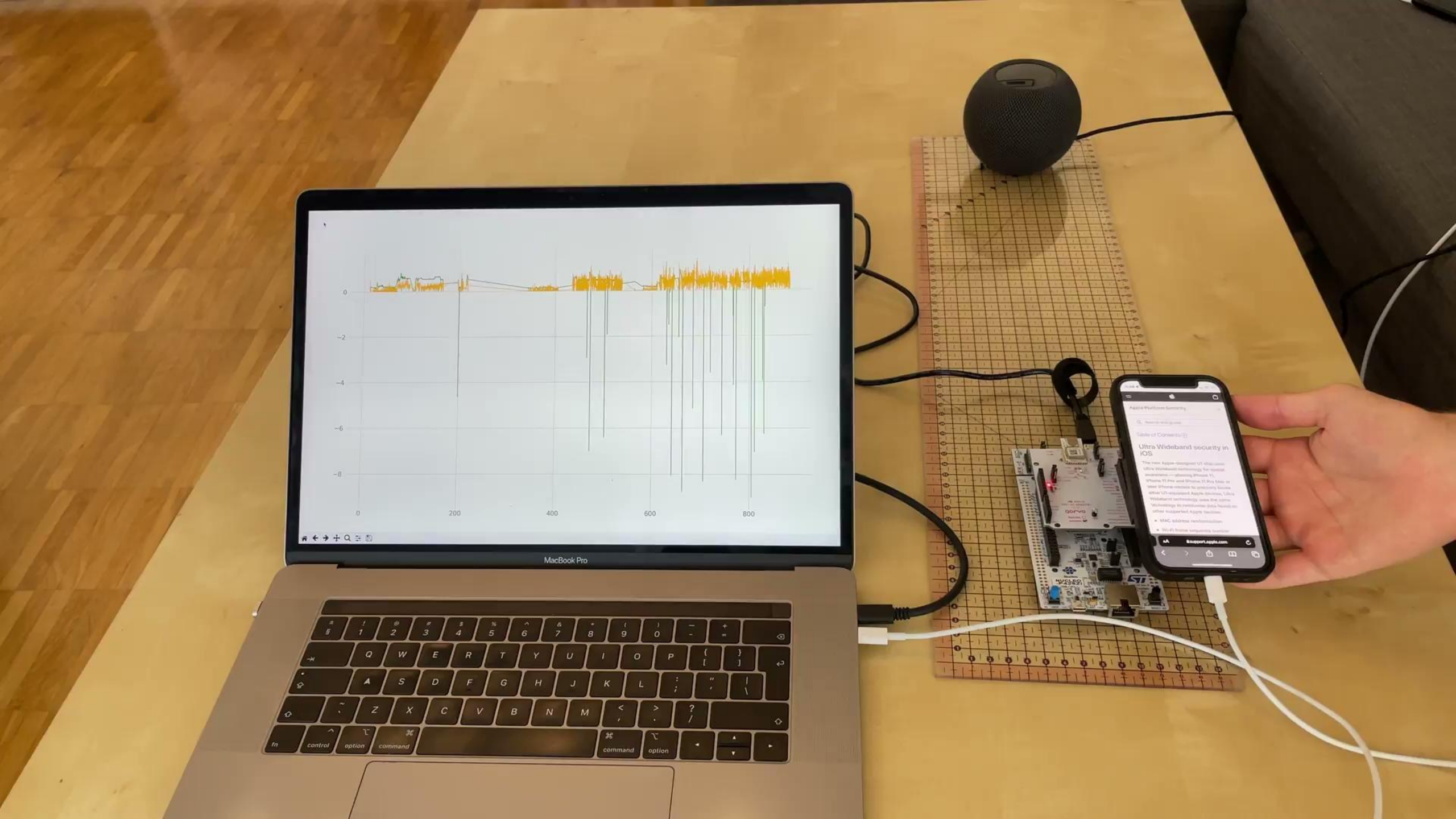
- Implemented first practical UWB distance-shortening attack.
- Firmware analysis done, but firmware is signed…



GhostPeak Paper, USENIX Security ‘22
AirTag Paper, WOOT ‘22

```
  * Received 4 bytes! Opcode 9 (Playing Sound)
  * Received 4 bytes! Opcode 9 (Playing Sound)
  * Received 4 bytes! Opcode 9 (Playing Sound)
  * Received 4 bytes! Opcode 9 (Playing Sound)
  * Received 1 bytes! Opcode 40 (Play Sound Sequ
  * CLDurianTask init
     > opcode:                14 (Set Mutex)
     > desiredLatency:        0x1
     > expectsResponse:       0x1
     > completeOnPreemption: 0x0
     > requiresMutex:         0x0
  * CLDurianDevice executeTask
  * Sent bytes!
  * Received 2 bytes! Opcode 0 (Acknowledge)
               > ACKed Opcode 14 (Set Mutex)
  * Received 7 bytes! Opcode 14 (Set Mutex)
[iPhone::locationd]->
[iPhone::locationd]->
```

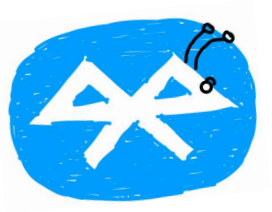



0
−2
−4
−6
−8
0
200
400
600
800
MacBook Pro
Ultra Widebband security in iOS

Conclusion

Let’s build more wireless research frameworks!

- Be the first to look into implementations of new technologies.
- Find high-impact vulnerabilities.
- Open frameworks to enable impactful research.



Q&A

 https://github.com/seemoo-lab

 Twitter: @naehrdine

 jiska@bluetooth.lol