All Your Wireless Are Belong To Us

or

The State Of Wireless Security: Screwing-up Sustainably Instead Of Sustainable Security

Joint Work With Various Members Of My Team And Collaborators:

J. Classen, E. Deligeorgopoulos,
F. Gringoli, A. Heinrich,
D. Kreitschmann, R. Klose,
M. Koch, J. Link,
M. Maas, D. Mantz, A. Mariotti,
S. Narain, G. Noubir,
M. Schulz, D. Steinmetzer, M. Stute,
F. Ullrich, D. Wegemer, and others

Projects Supporting This Work:
MAKI, NICER, CROSSING, CRISP

‘Standard’ Outline

WiFi CERTIFIED

Bluetooth SMART

AirDrop

These Standards Are The Lifeline Of Billions Of Devices

How Many?

Internet says:

Wi-Fi: >10 B

Bluetooth: >10 B

Airdrop: >1.4 B

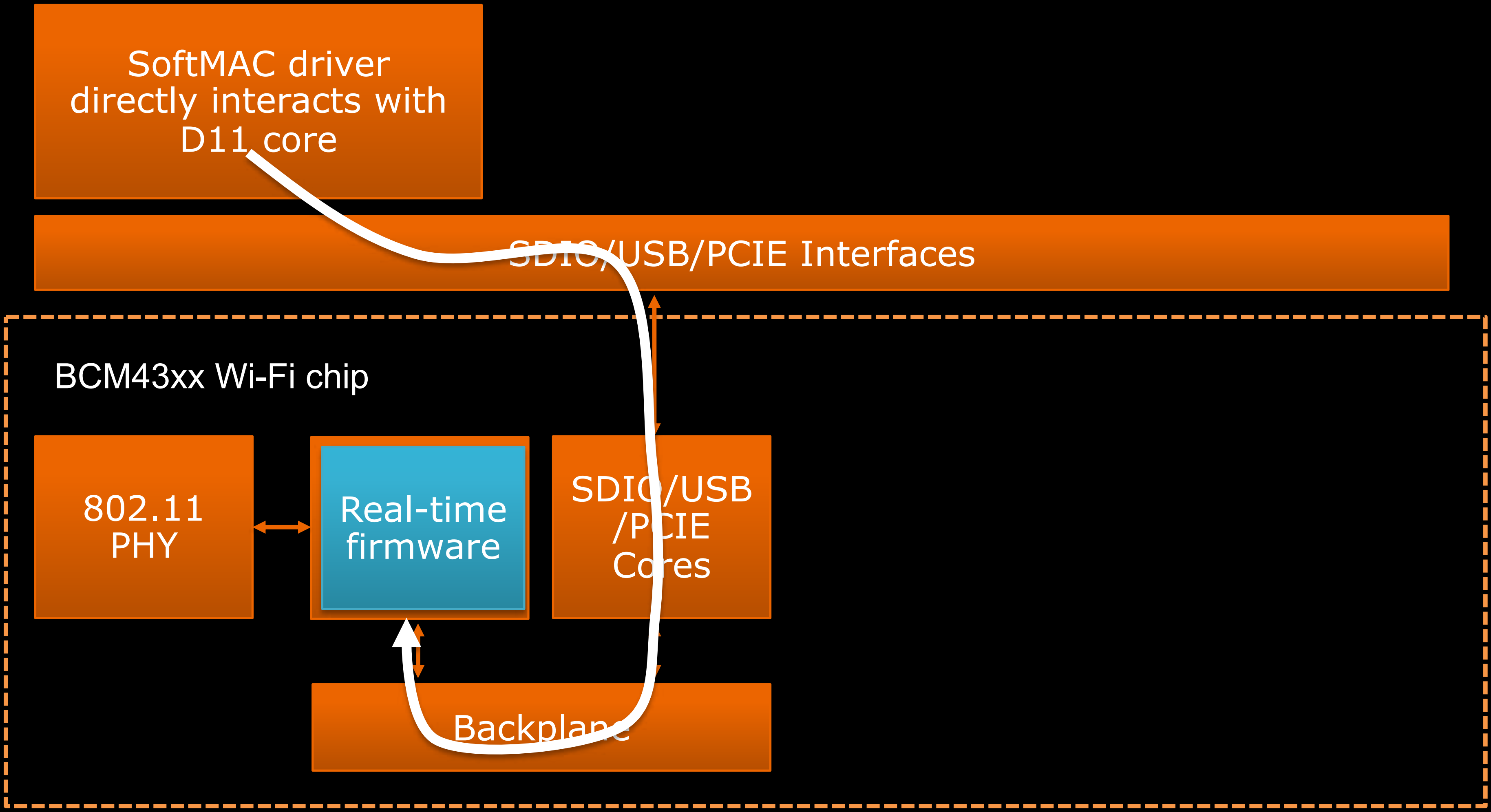
Part 1: Wi-Fi

How To Unleash The Power Of FullMAC Wi-Fi Firmware?

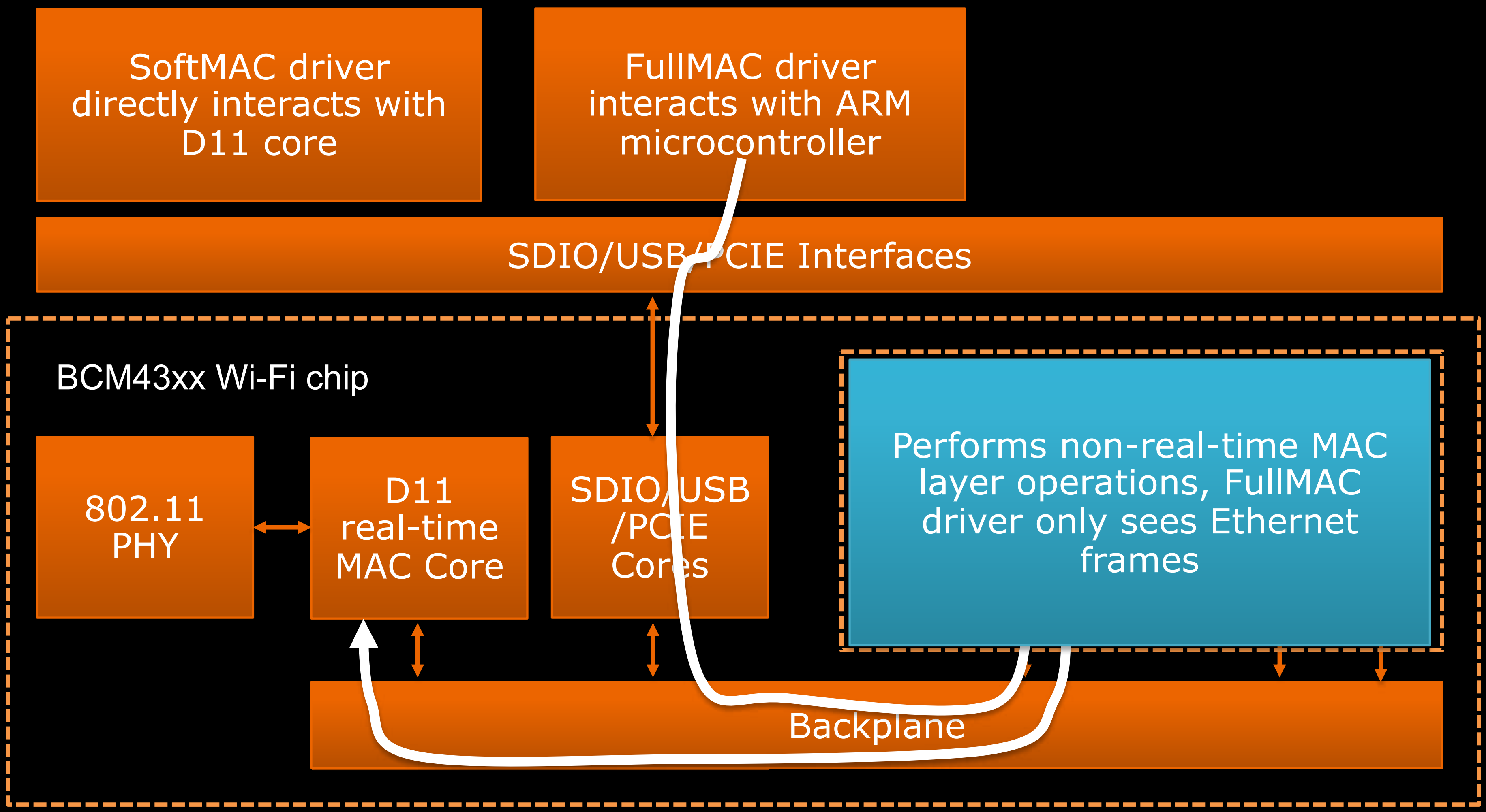
nexmon

www.nexmon.org

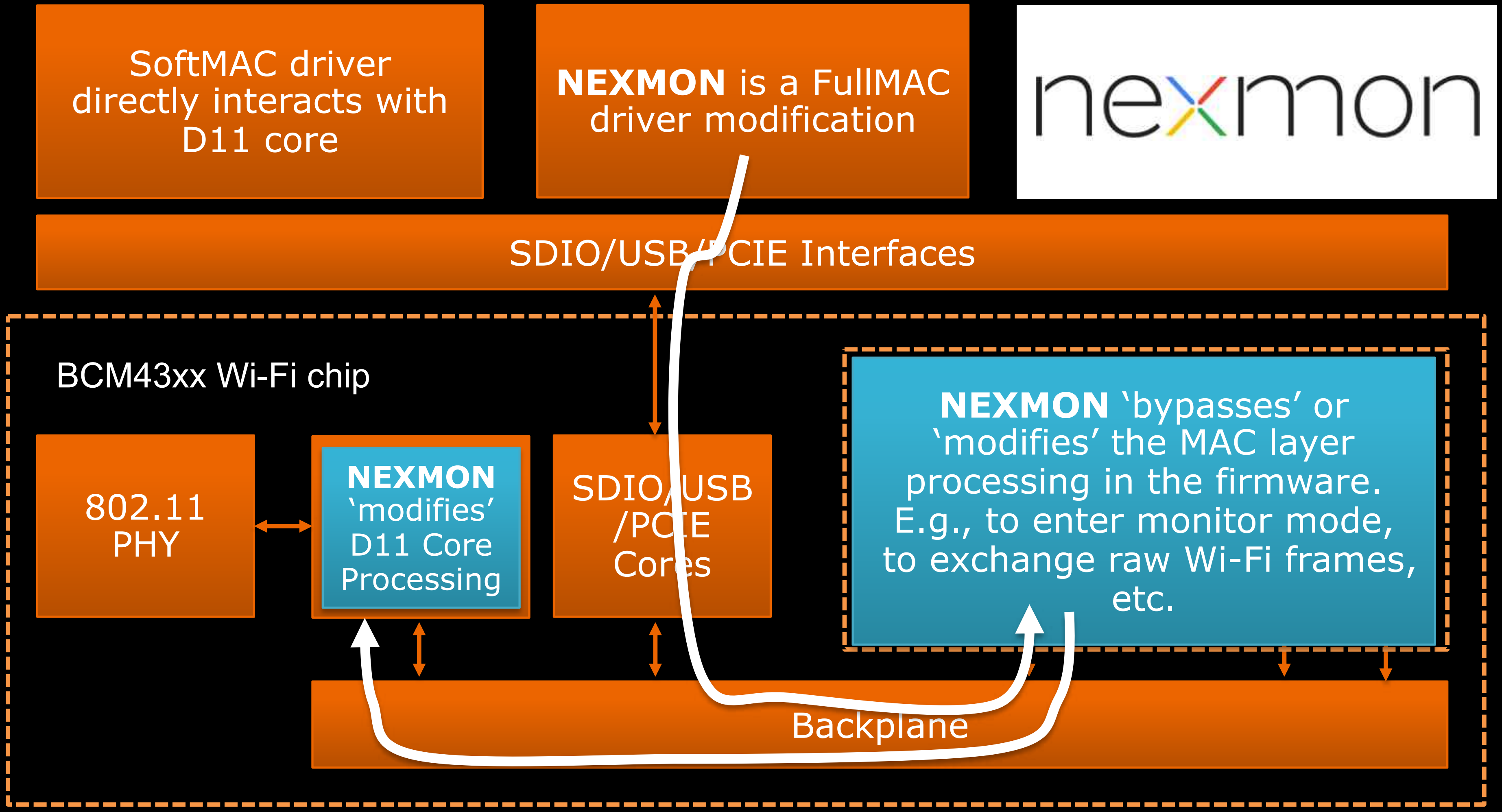
Structure of Broadcom Wi-Fi Chips



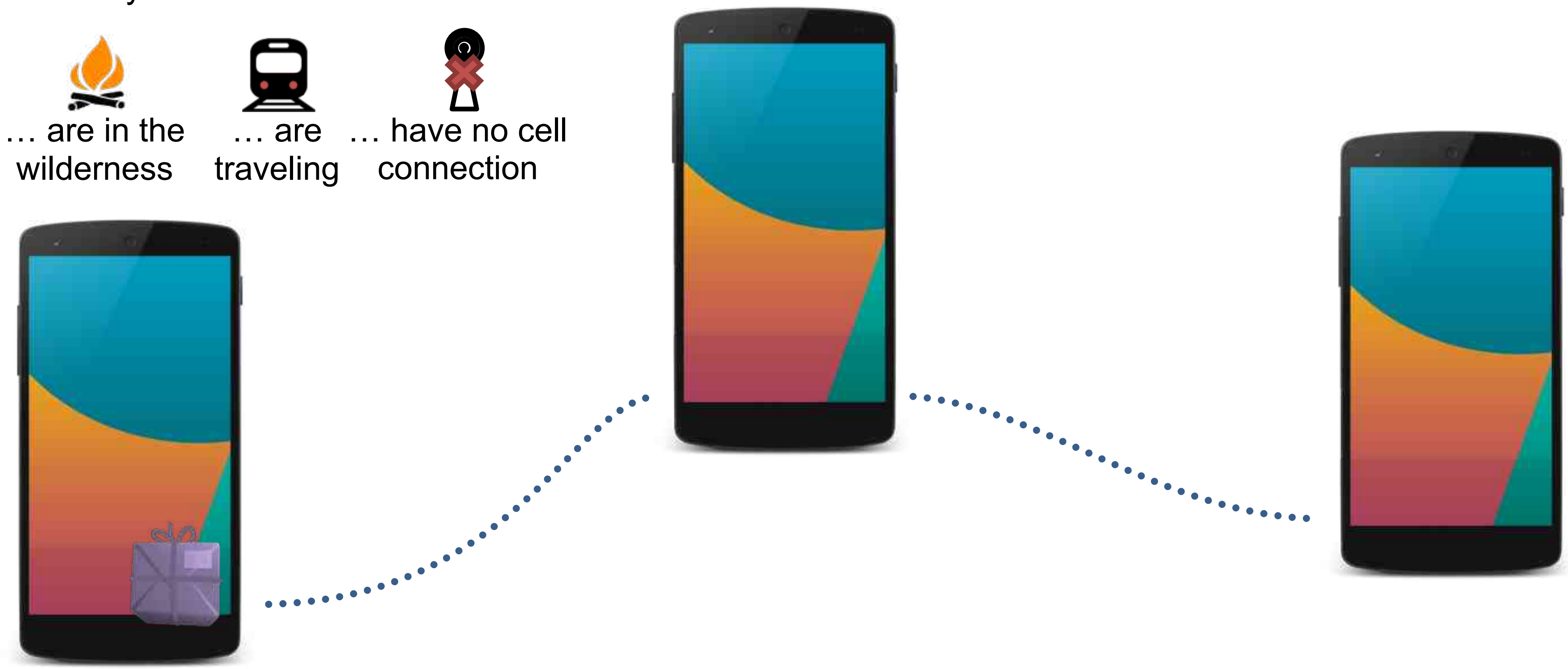
Structure of Broadcom Wi-Fi Chips



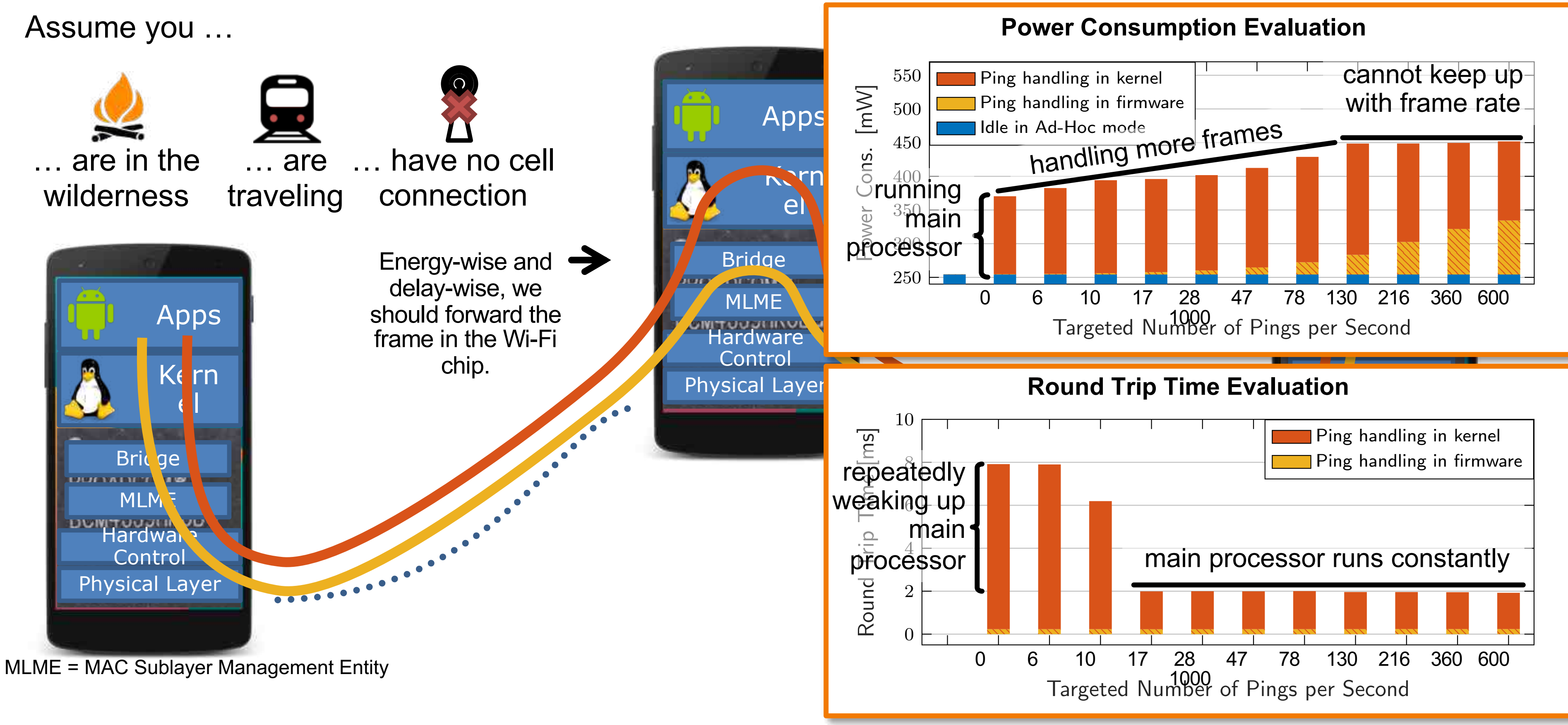
Structure of Broadcom Wi-Fi Chips



Assume you …

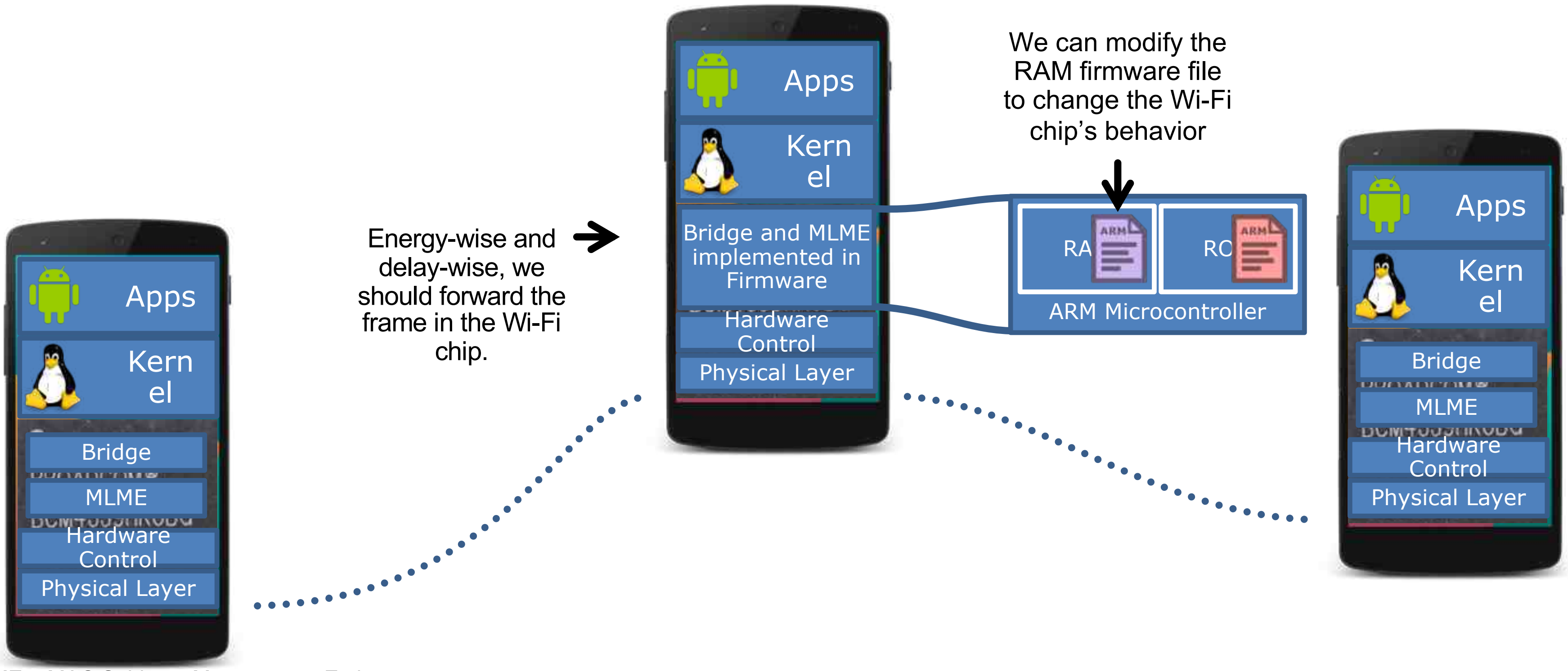


Ad Hoc Communication Scenario



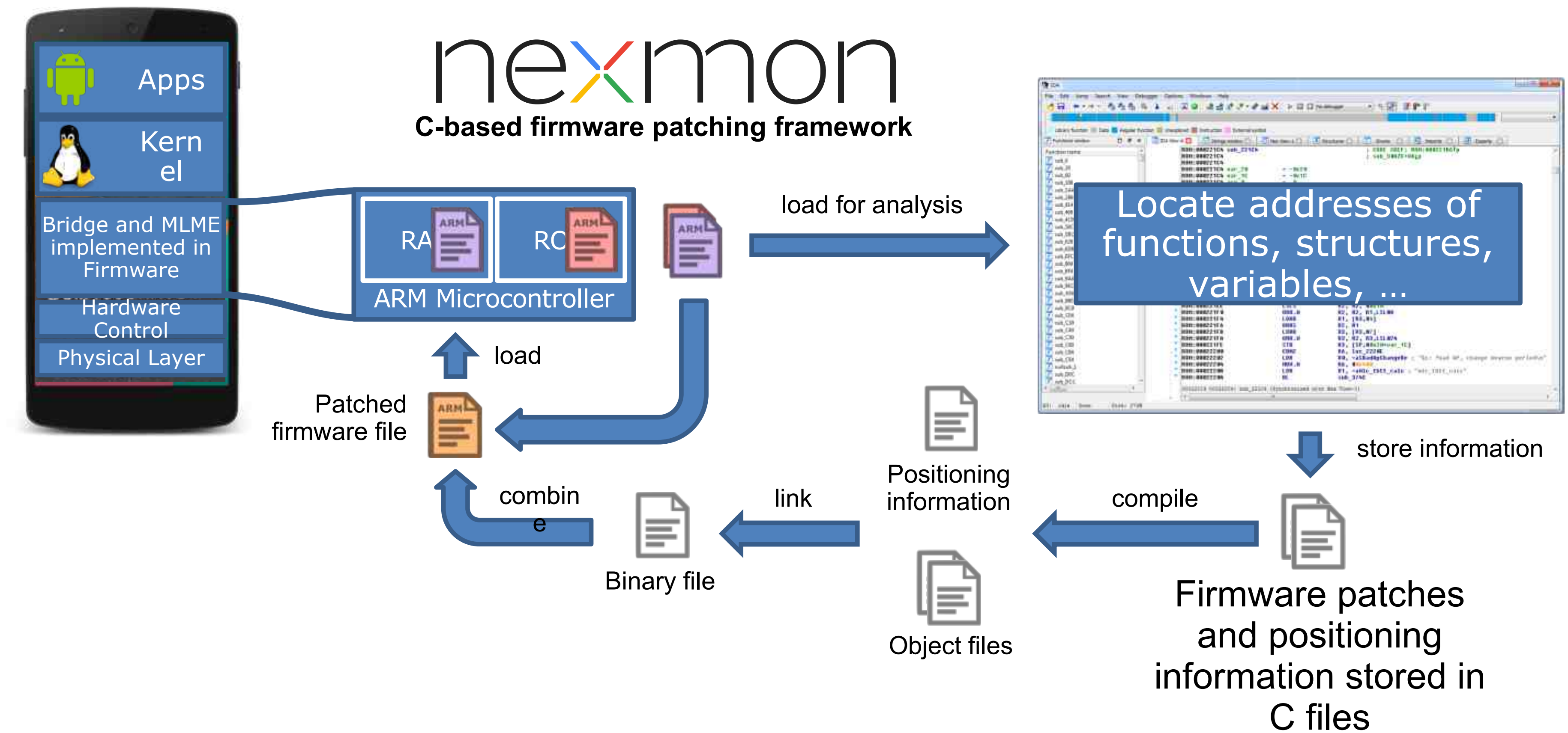
MLME = MAC Sublayer Management Entity

Energy Consumption and Delay

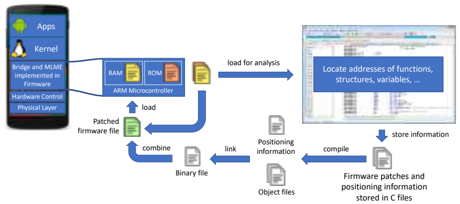


MLME = MAC Sublayer Management Entity

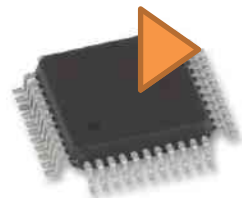
Analyzing the Wi-Fi Firmware



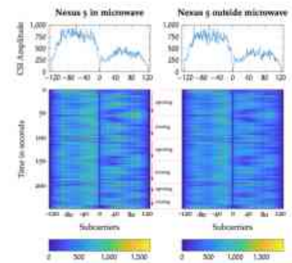
Framework and Toolset



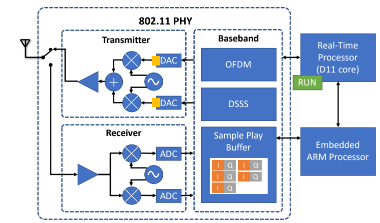
nexmon
C-based firmware patching framework



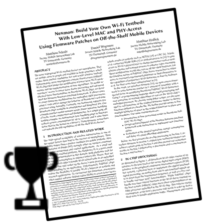
Programmable Debugger



Channel State Information Extractor



Software-defined Radios on Wi-Fi Chips



WiNTECH'17 Paper

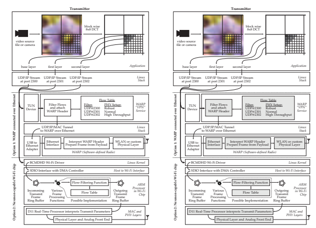
COMCOM '18 Article

MobiSys'18 Paper
WiNTECH'19 Paper

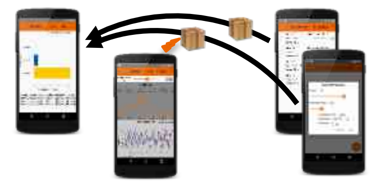
WiSec'17 Paper,
MobiSys'18 Paper

Framework and Toolset

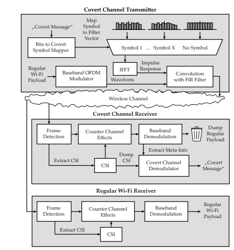
Ping Offloading



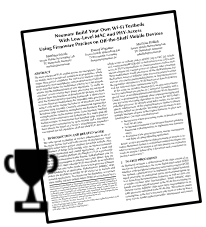
Software-defined Wireless Networking for Scalable Video Streaming



Reactive Wi-Fi Jammer on Smartphones



Wi-Fi-based Covert Channel



WiNTECH'17 Paper

NetSys'15 Paper

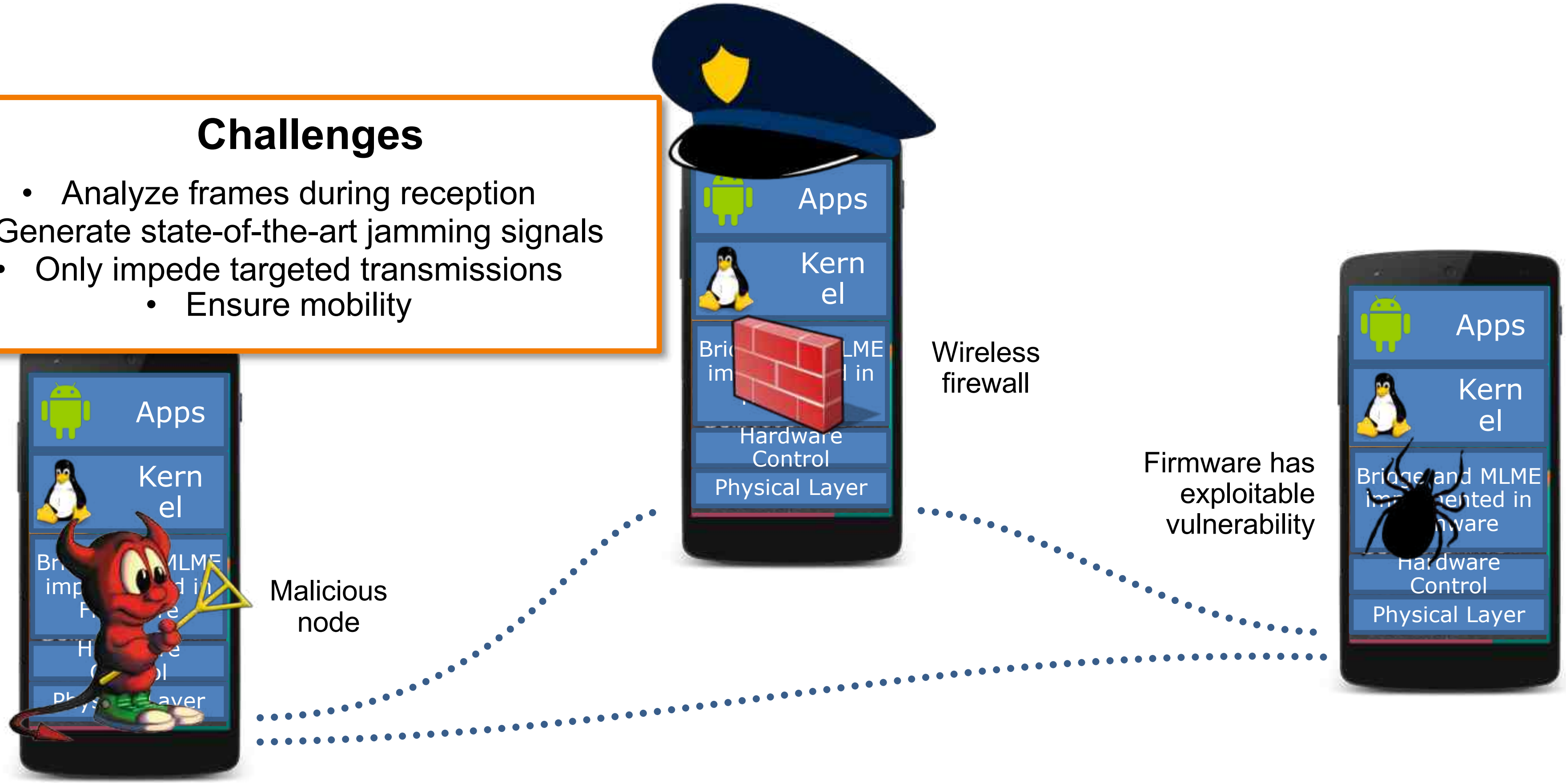
WiSec'17 Paper

MobiSys'18 Paper

Applications

Challenges

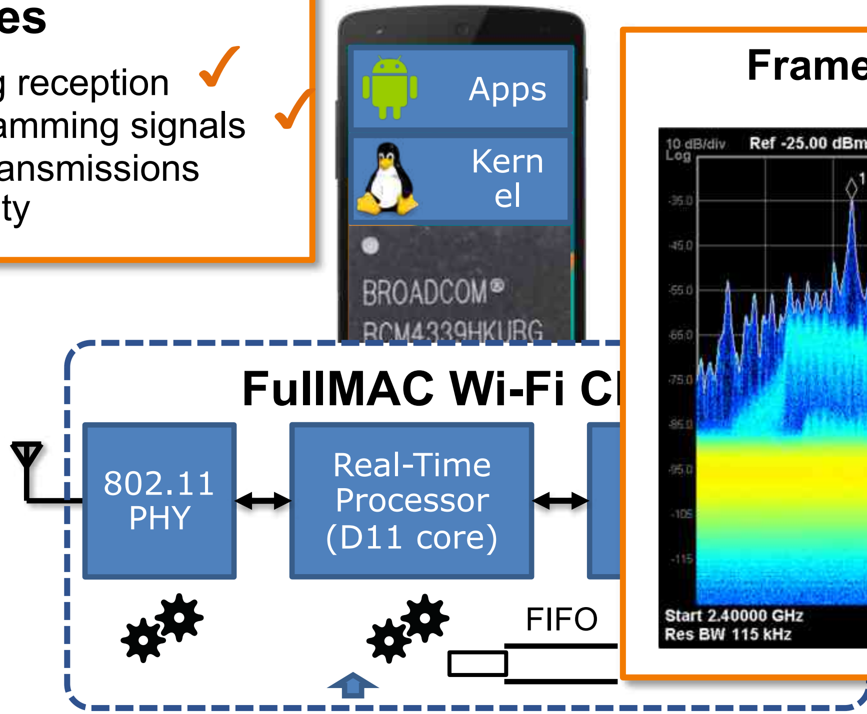
- Analyze frames during reception
- Generate state-of-the-art jamming signals
- Only impede targeted transmissions
- Ensure mobility



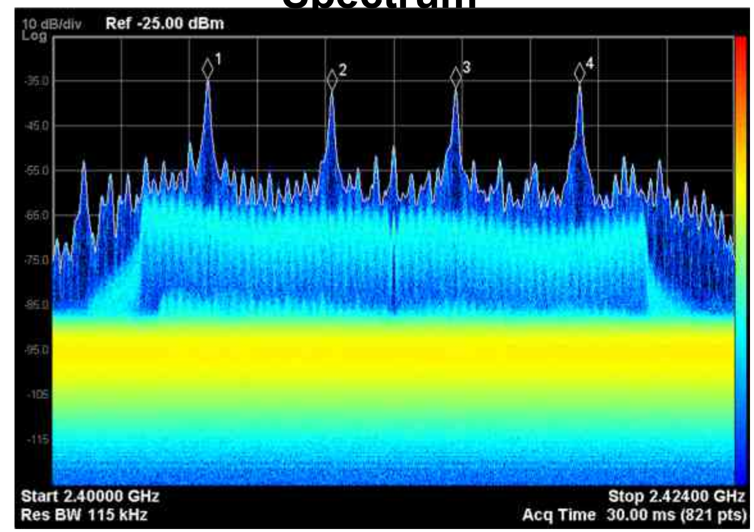
Wi-Fi Jamming on Smartphones

Challenges

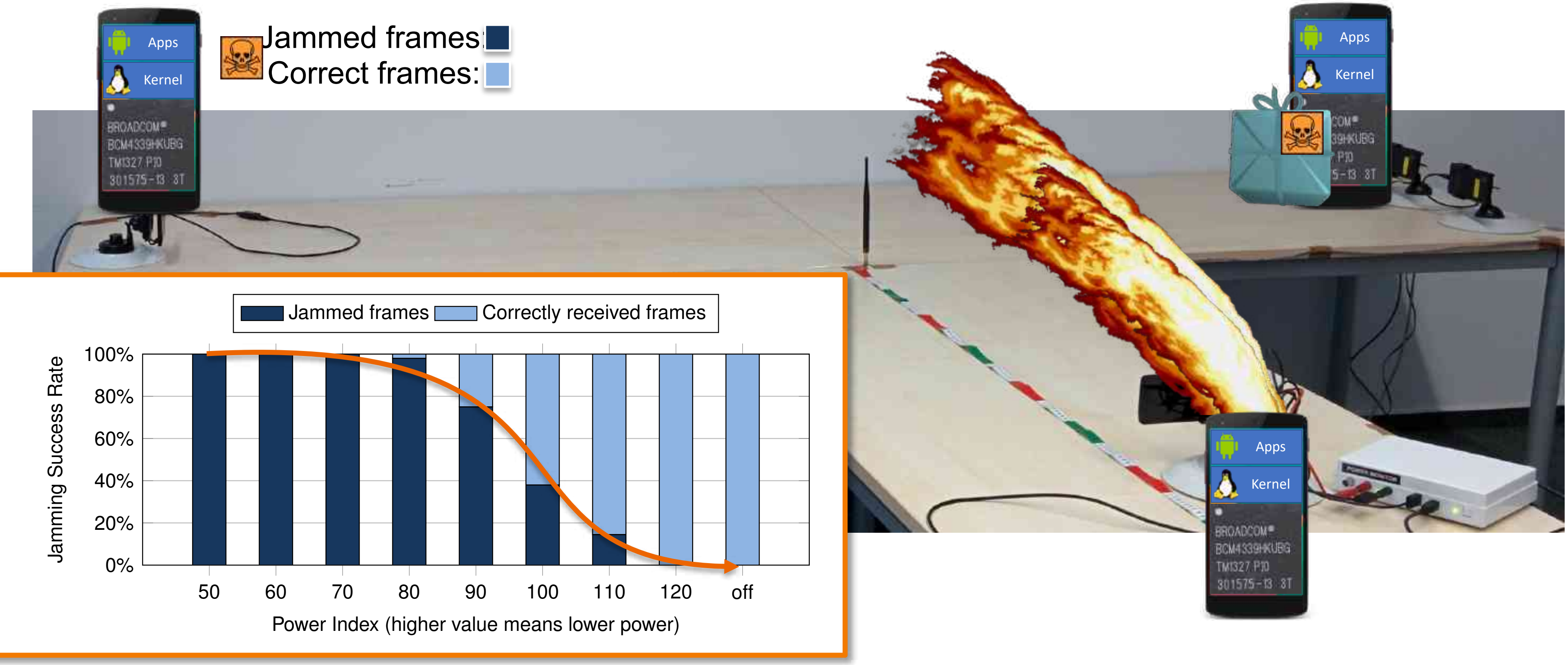
- Analyze frames during reception ✓
- Generate state-of-the-art jamming signals ✓
- Only impede targeted transmissions
- Ensure mobility



Frame and Jamming Signal Spectrum



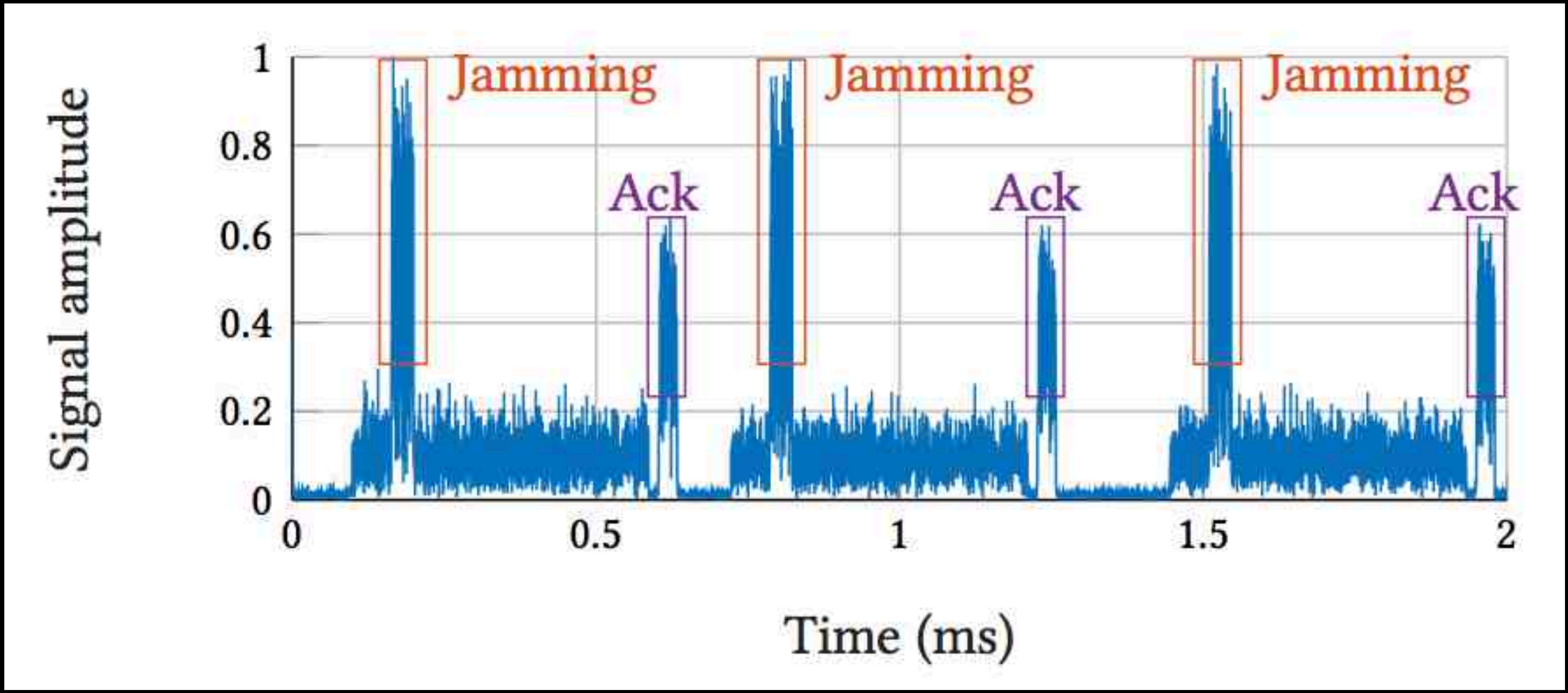
Implementing the Reactive Jammer



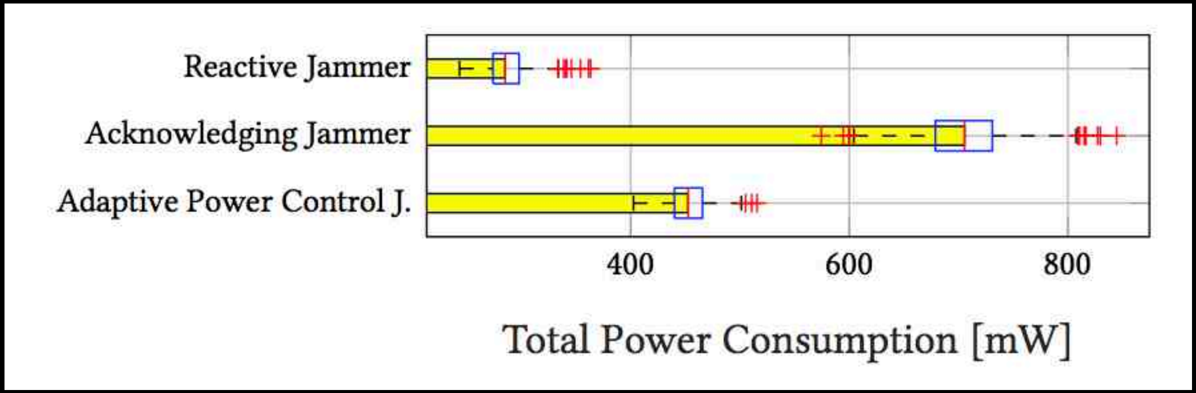
Evaluating the Reactive Jammer

Ack. Jammer

(approx. 20us delay after parsing UDP port)



Energy?



This translates to

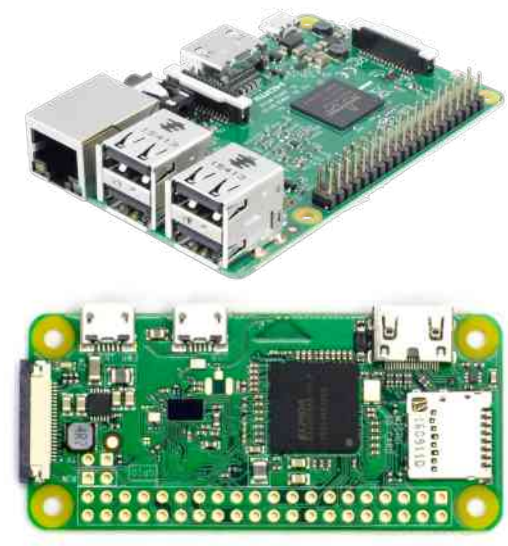
- 30.7 hours runtime as reactive jammer
- 12.4 hours runtime as acknowledging jammer
- 19.3 hours runtime as adaptive power control jammer

Supported Platforms

Nexus 5

SGS 2

Nexus 6P



Raspberry Pi 3
Raspberry Pi Zero W

And More

We ❤

BROADCOM®

Who Uses It?

Project Zero

Google

News and updates from the Project Zero team at Google

Tuesday, April 4, 2017

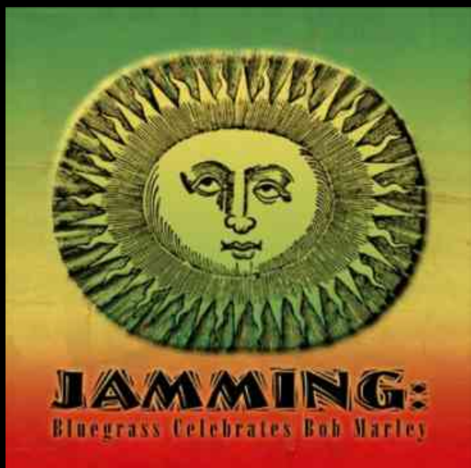
Over The Air: Exploiting Broadcom's Wi-Fi Stack (Part 1)

Posted by Gal Beniamini, Project Zero

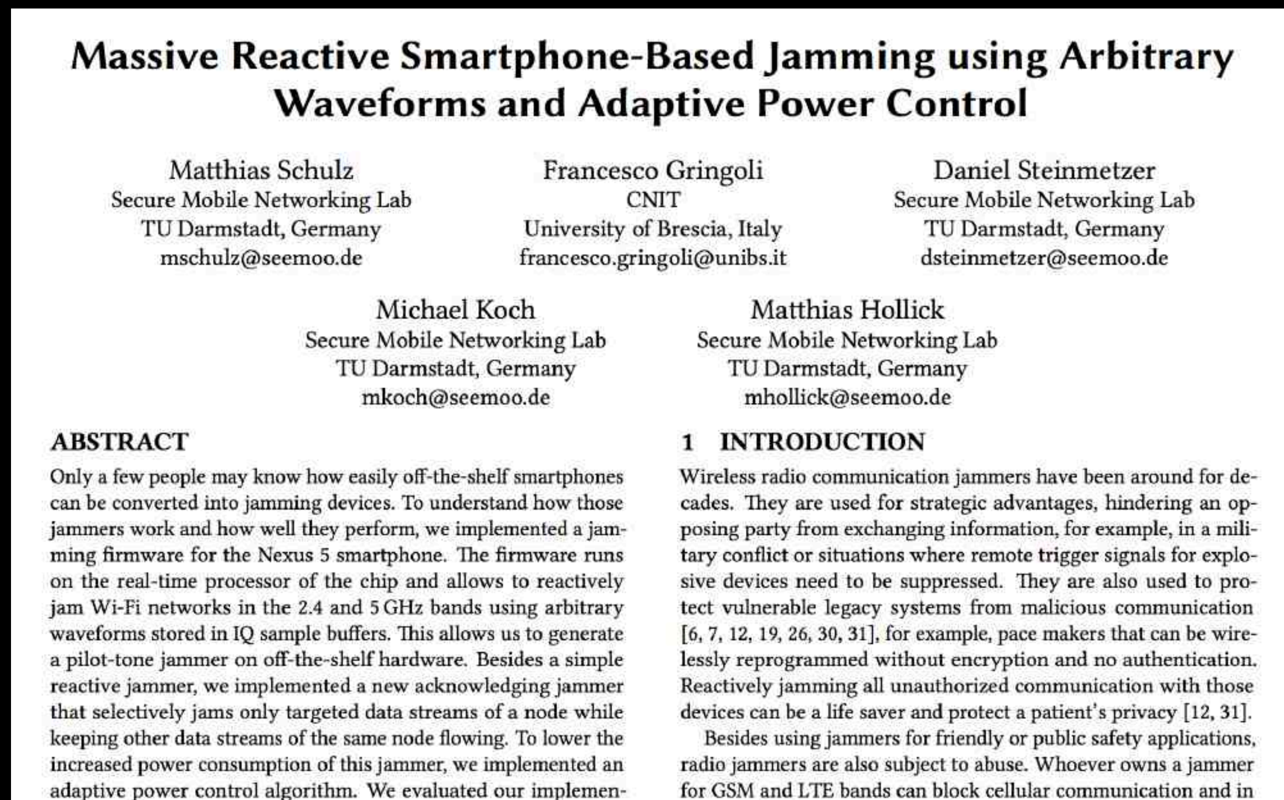
It's a well understood f... ...urity of complex systems. For ...m mobile platforms include multiple processing units,

one last question we need to answer first - *where* is the ROM located? According to the great research done by the folks behind NexMon, the ROM is loaded at address 0x0, and the RAM is loaded at address

Breaking Wi-Fi (For Good)



Source: http://www.cmhrecords.com/



Massive Reactive Smartphone-Based Jamming using Arbitrary Waveforms and Adaptive Power Control

Matthias Schulz
Secure Mobile Networking Lab
TU Darmstadt, Germany
mschulz@seemoo.de

Francesco Gringoli
CNIT
University of Brescia, Italy
francesco.gringoli@unibs.it

Daniel Steinmetzer
Secure Mobile Networking Lab
TU Darmstadt, Germany
dsteinmetzer@seemoo.de

Michael Koch
Secure Mobile Networking Lab
TU Darmstadt, Germany
mkoch@seemoo.de

Matthias Hollick
Secure Mobile Networking Lab
TU Darmstadt, Germany
mhollick@seemoo.de

ABSTRACT

Only a few people may know how easily off-the-shelf smartphones can be converted into jamming devices. To understand how those jammers work and how well they perform, we implemented a jamming firmware for the Nexus 5 smartphone. The firmware runs on the real-time processor of the chip and allows to reactively jam Wi-Fi networks in the 2.4 and 5 GHz bands using arbitrary waveforms stored in IQ sample buffers. This allows us to generate a pilot-tone jammer on off-the-shelf hardware. Besides a simple reactive jammer, we implemented a new acknowledging jammer that selectively jams only targeted data streams of a node while keeping other data streams of the same node flowing. To lower the increased power consumption of this jammer, we implemented an adaptive power control algorithm. We evaluated our implemen-

1 INTRODUCTION

Wireless radio communication jammers have been around for decades. They are used for strategic advantages, hindering an opposing party from exchanging information, for example, in a military conflict or situations where remote trigger signals for explosive devices need to be suppressed. They are also used to protect vulnerable legacy systems from malicious communication [6, 7, 12, 19, 26, 30, 31], for example, pace makers that can be wirelessly reprogrammed without encryption and no authentication. Reactively jamming all unauthorized communication with those devices can be a life saver and protect a patient's privacy [12, 31].

Besides using jammers for friendly or public safety applications, radio jammers are also subject to abuse. Whoever owns a jammer for GSM and LTE bands can block cellular communication and in

WiSec 2017

NEXMON fully utilized

- Transmission of arbitrary waveforms on any frequency supported by the Broadcom Wi-Fi chip
- Implementation of a smartphone-based reactive jammer for Wi-Fi systems that can jam all receivable rates supported by Nexus 5 smartphones (e.g., 80 MHz SISO 802.11ac frames)
- Enhanced jammers:
  - By sending acknowledgements to the frame transmitter to avoid retransmissions and the blockage of other non-targeted traffic
  - By using an adaptive power control algorithm to adjust the transmission power depending on the jamming success

Demo

DEMO: Demonstrating Reactive Smartphone-Based Jamming

Matthias Schulz, Efstathios Deligeorgopoulos, Matthias Hollick
Secure Mobile Networking Lab
TU Darmstadt, Germany
{mschulz,edeligeorgopoulos,mhollick}@seemoo.de

Francesco Gringoli
CNIT - DII
University of Brescia, Italy
francesco.gringoli@unibs.it

ABSTRACT

The practicability of reactive Wi-Fi jammers using commercial off-the-shelf (COTS) hardware has only been shown recently. For instance, it can serve to facilitate mobile friendly jamming applications. Until now, no demonstrators existed to reproduce the results obtained with these systems, hence, inhibiting re-use for further research or other applications. Moreover, there is a lack of practical jammers that can be used for educational purposes. In this work, we present an Android app that allows to create advanced jamming scenarios with at least three Nexus 5 smartphones. We use one or more of them to generate and transmit Wi-Fi frames with UDP

Nexmon framework [4, 5], that makes it possible to modify Wi-Fi firmwares running in Broadcom FullMAC chips on smartphones. Their jammer not only supports single-stream 802.11ac frames, but also allows to design arbitrary jamming waveforms by writing I/Q samples into a buffer that can be played back whenever a jamming condition matches. This combines the flexibility of an SDR with the ubiquitous availability of low-cost Wi-Fi chips. Additionally, the authors presented a new jamming attack that sends fake acknowledgements to the transmitter of a target frame. This avoids blocking the transmission of other non-targeted frames at the transmitter.

In this work, we present an Android app that demonstrates and

WISEC 2017

Demo App To Explore

Demo App To Explore

Total Cost:

300 EUR
(1 Smartphone)

We just updated and re-released the CSItool for Broadcom chipsets

Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets

Francesco Gringoli
CNIT/University of Brescia
Italy
francesco.gringoli@unibs.it

Matthias Schulz
TU Darmstadt
Germany
mschulz@seemoo.tu-darmstadt.de

Jakob Link
TU Darmstadt
Germany
jlink@seemoo.tu-darmstadt.de

Matthias Hollick
TU Darmstadt
Germany
matthias.hollick@seemoo.tu-darmstadt.de

ABSTRACT

Modern wireless transmission systems heavily benefit from knowing the channel response. The evaluation of Channel State Information (CSI) during the reception of a frame preamble is fundamental to properly equalizing the rest of the transmission at the receiver side. Reporting this state information back to the transmitter facilitates mechanisms such as beamforming and MIMO, thus boosting the network

ACM Reference Format:
Francesco Gringoli, Matthias Schulz, Jakob Link, and Matthias Hollick. 2019. Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets. In *13th International Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization (WiNTECH '19), October 25, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3349623.3355477

ACM WiNTECH 2019

README.md

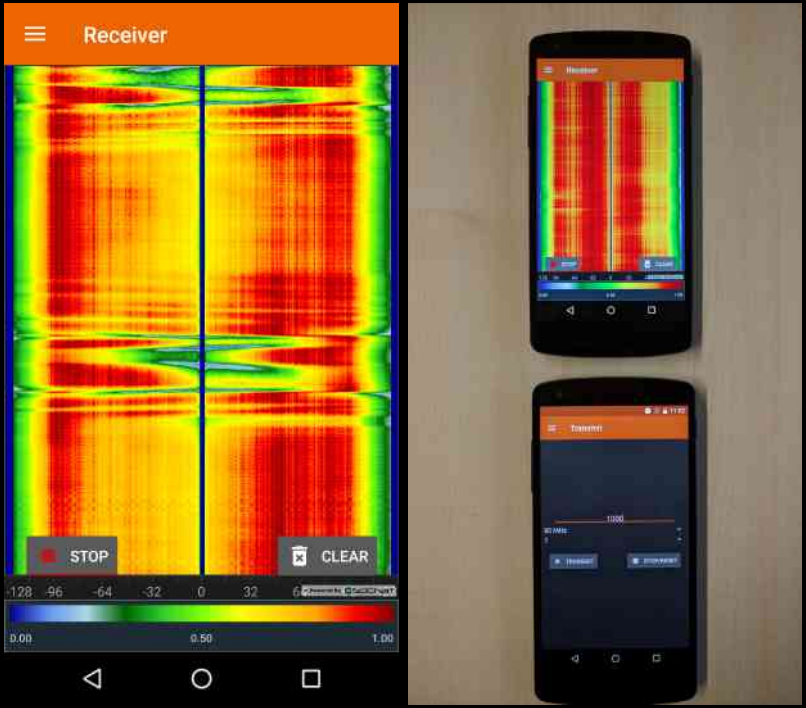
Nexmon Channel State Information Extractor

This project allows you to extract channel state information (CSI) of OFDM-modulated Wi-Fi frames (802.11a/(g)/n/ac) on a per frame basis with up to 80 MHz bandwidth on the Broadcom Wi-Fi Chips listed below.

WiFi Chip	Firmware Version	Used in
bcm4339	6_37_34_43	Nexus 5
bcm43455c0	7_45_189	Raspberry Pi B3+/B4
bcm4358	7_112_300_14_sta	Nexus 6P
bcm4366c0	10_10_122_20	Asus RT-AC86U

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

Tool	Open Source	Device	Supp. Chipset	max. BW	NSS×NRX	Supp. Std.	NSC	Res.
nexmon CSI Extractor	yes	Router, PCIE e.g. Asus RT-AC86U	BCM43{65, 66}	80 MHz	4×4	VHT/11ac	242	12 bit (f)
	yes	Smartphone, IoT e.g. Nexus 5/6P, RPi3B+/4B	BCM43{39, 58, 455}	80 MHz	1×1	VHT/11ac	242	14 bit (i) or 10 bit (f)



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

Total Cost:

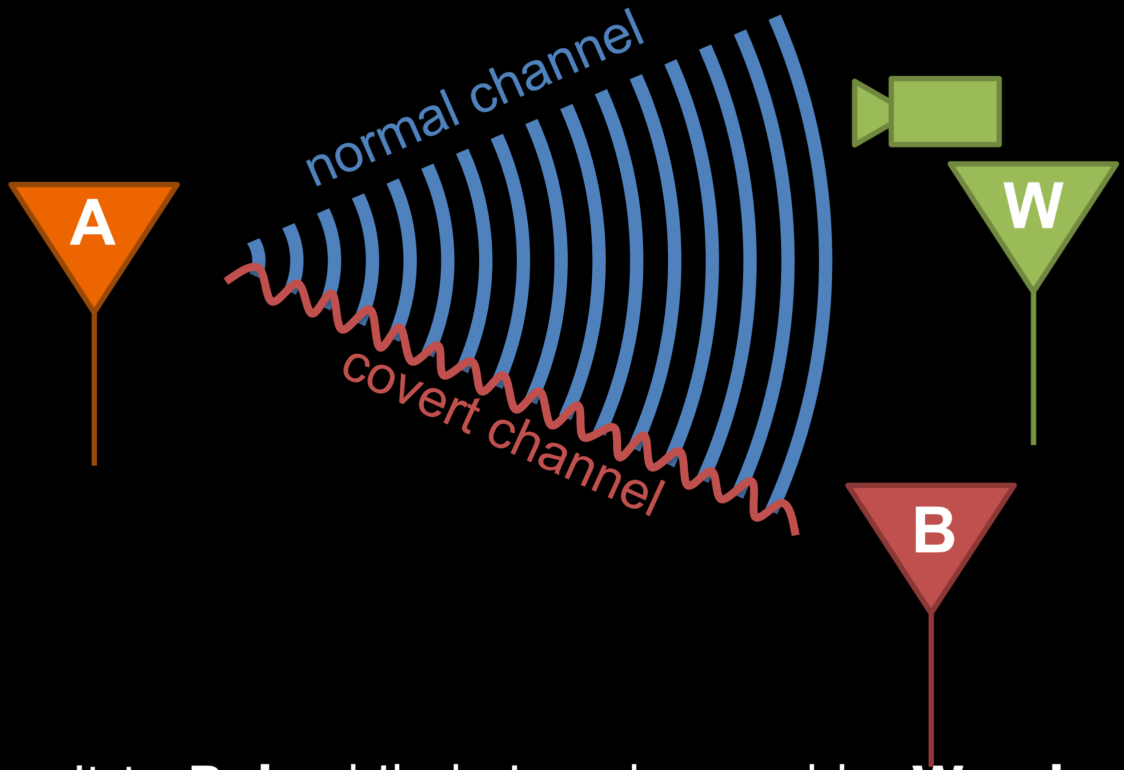
30 EUR (1 RaspPI)
to
130 EUR (1 ASUS AP)

More On Wi-Fi

Quiz: What Do You See? What Was Its Purpose?

‘The Thing’ Or ‘The Great Seal Bug’ … In The Past Exfiltration Of Data Was A Tedious Task: Expensive, Low Bandwidth Channel

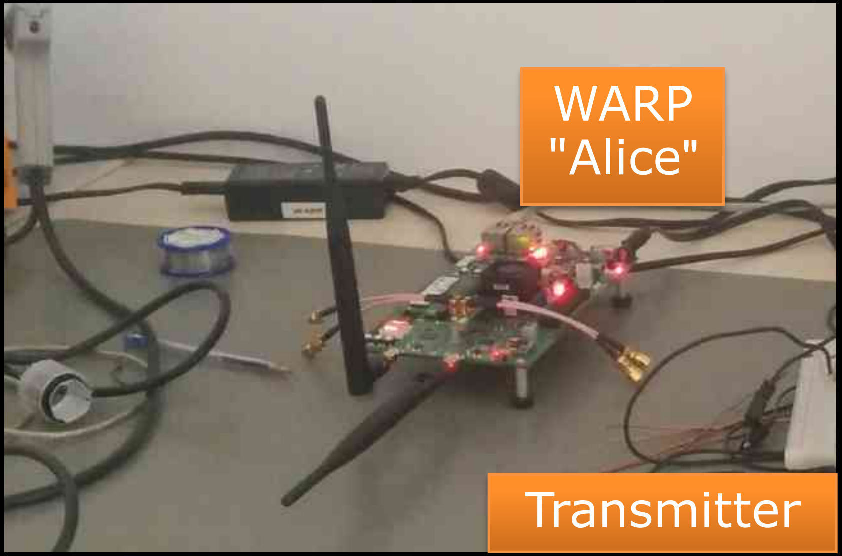
Application: Covert Communication

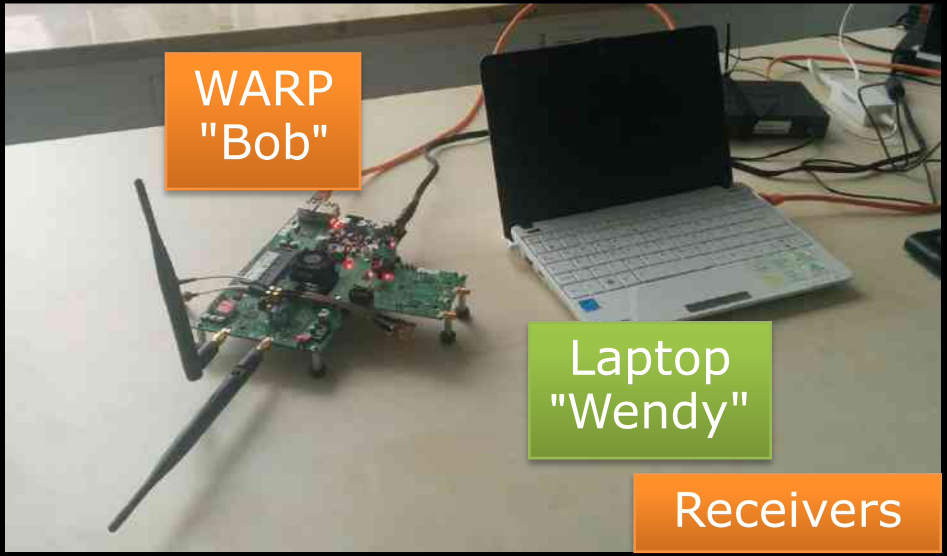


Toy scenario

- **Alice** wants to transmit to **Bob** while being observed by **Wendy**
- Alice and Bob aim at staying **undetected**

Setup

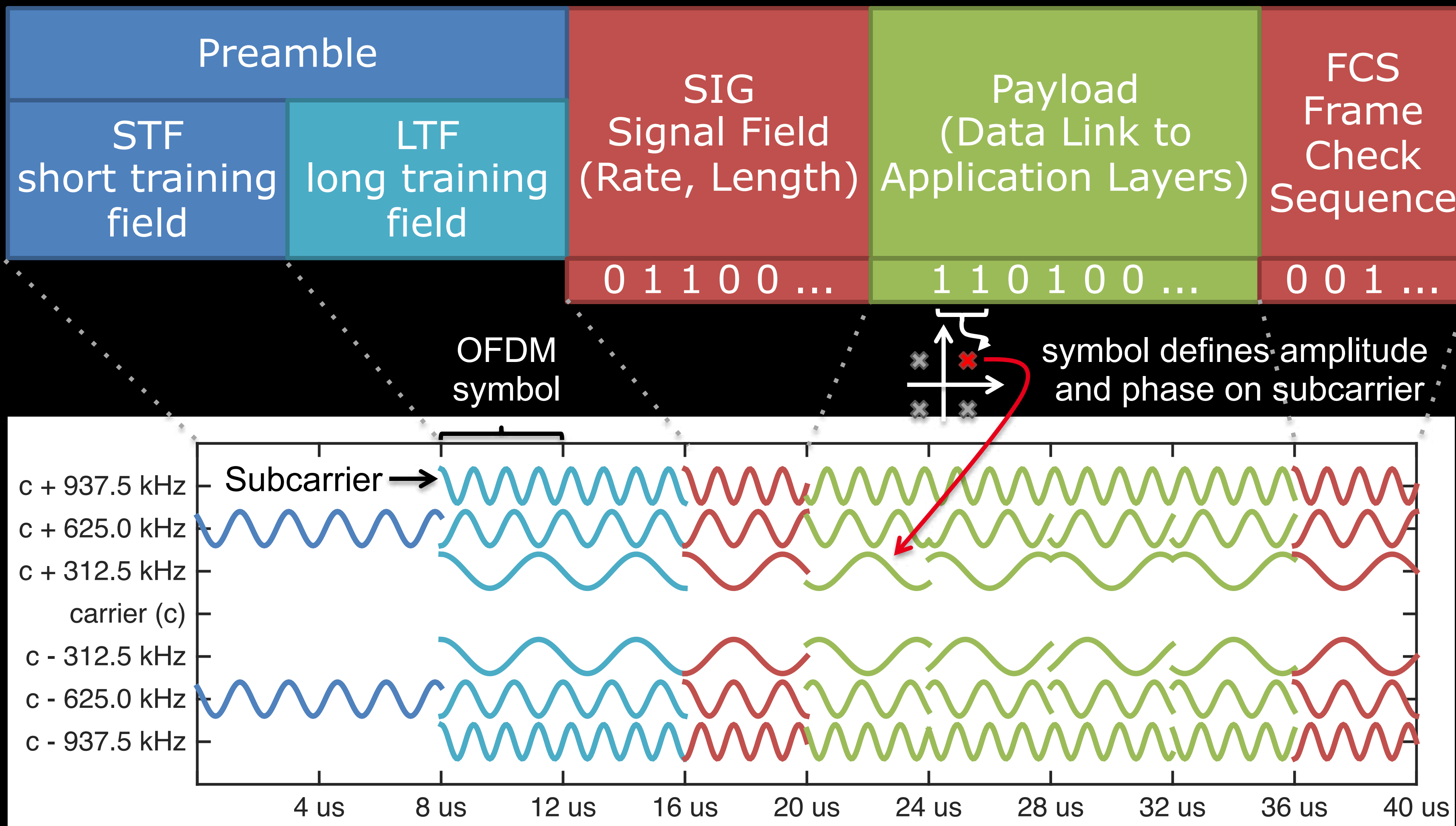

WARP
"Alice"
Transmitter


WARP
"Bob"
Laptop
"Wendy"
Receivers

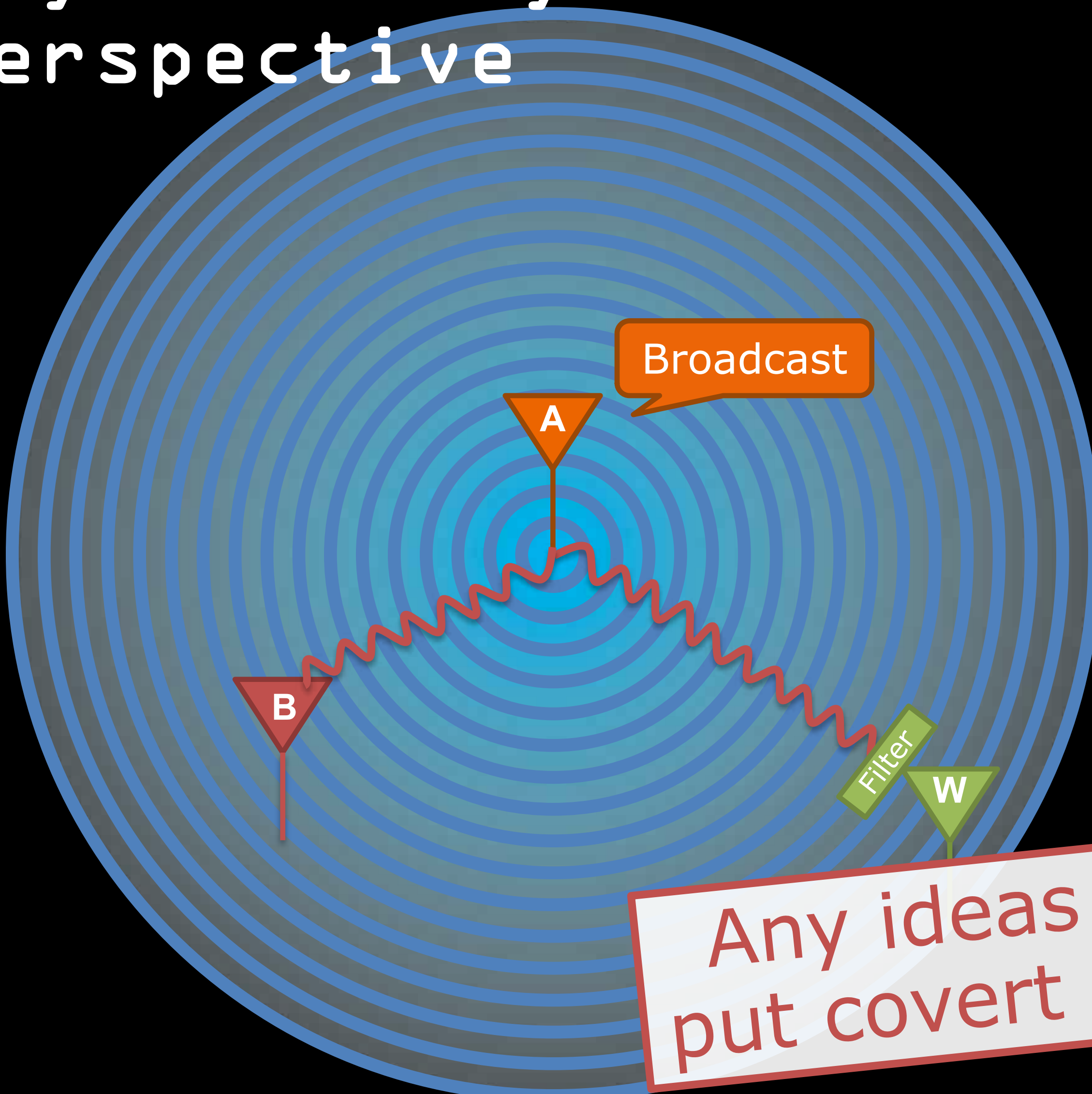

WARP
Receiver
5m
WARP
Transmitter
Laptop
Receiver

Background on Wi-Fi

How Wi-Fi frames look like ...



Background on Wi-Fi
Physical Layer Perspective



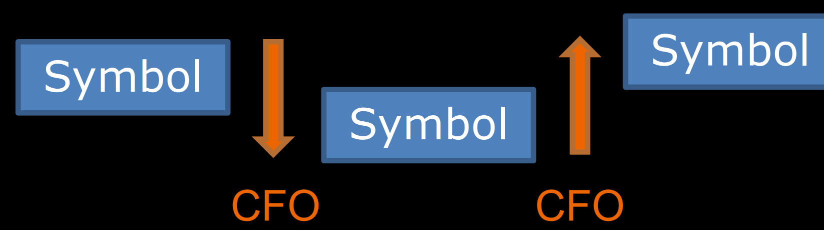
- Random **noise** at each receiver
- **Receivers are designed to compensate** for distortions
- „Unnecessary“ information is added to the signal to increase robustness

Any ideas on where to put covert PHY channels?

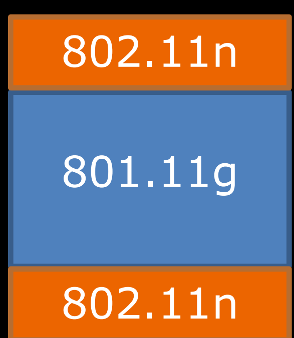
Covert Channel Overview

Short Training Field with Phase Shift Keying

Carrier Frequency Offset with Frequency Shift Keying

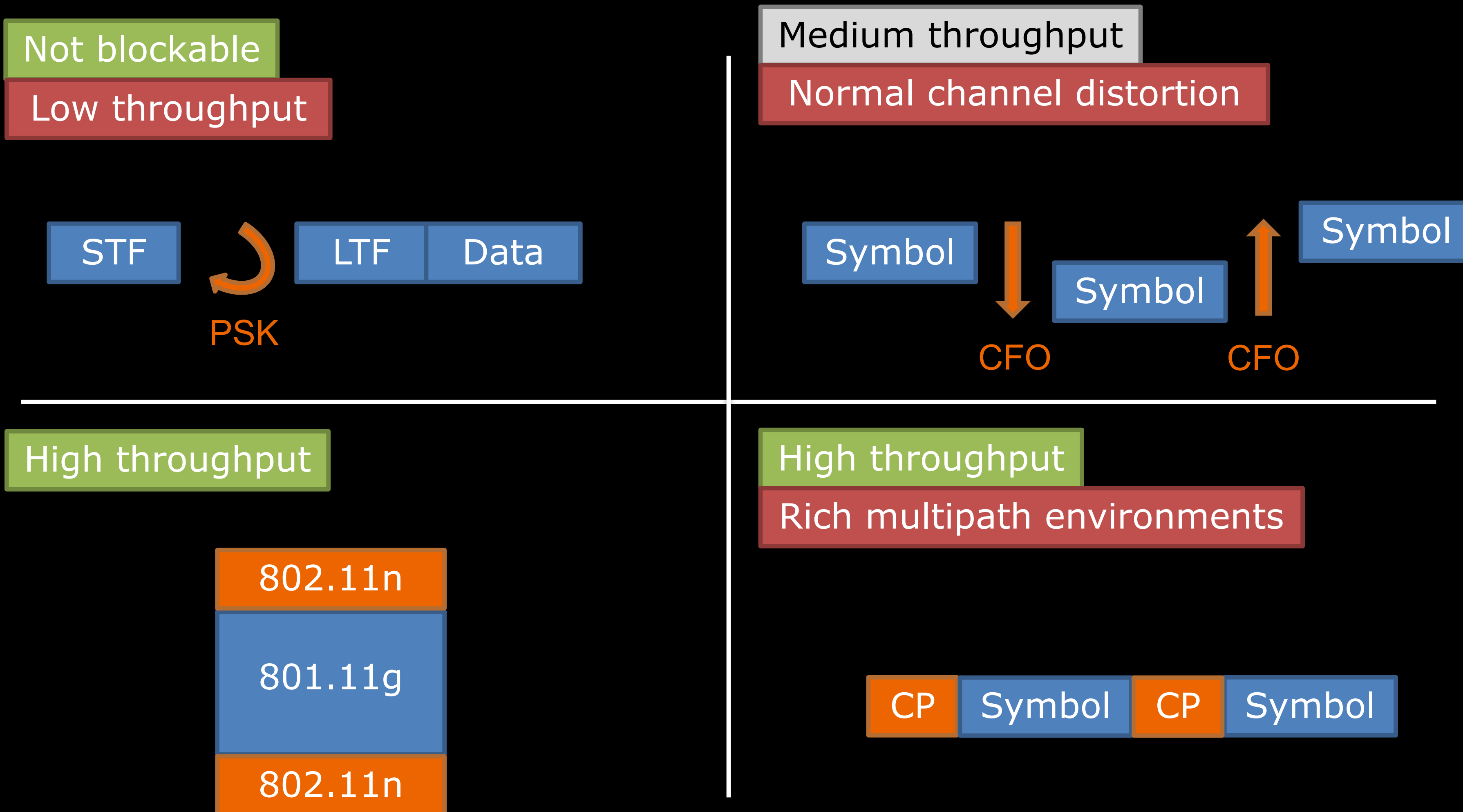


Camouflage Subcarriers



Cyclic Prefix Replacement

Summary and Conclusion: Covert Channel Overview



Practical Covert Channels for WiFi Systems

Jiska Classen *, Matthias Schulz *, and Matthias Hollick
Secure Mobile Networking Lab
Technische Universität Darmstadt
{jclassen, mschulz, mhollick}@seemoo.tu-darmstadt.de

***Abstract*—Wireless covert channels promise to exfiltrate information with high bandwidth by circumventing traditional access control mechanisms. Ideally, they are only accessible by the intended recipient and—for regular system users/operators—indistinguishable from normal operation. While a number of theoretical and simulation studies exist in literature, the practical aspects of WiFi covert channels are not well understood. Yet, it is particularly the practical design and implementation aspect of wireless systems that provides attackers with the latitude to establish covert channels: the ability to operate under adverse conditions and to tolerate a high amount of signal variations. Moreover, covert physical receivers do not have to be addressed within wireless frames, but can simply eavesdrop on the transmission. In this work, we analyze the possibilities to establish covert channels in WiFi systems with emphasis on exploiting physical layer characteristics. We discuss design alternatives for selected covert channel approaches and study their feasibility in practice. By means of an extensive performance analysis, we compare the covert channel bandwidth. We further evaluate the possibility of revealing the introduced covert channels based on different detection capabilities.**

instance, an online banking application could establish a secure connection to a server but maliciously publish login data over a covert wireless physical channel.

WiFi covert channels have been mostly studied in theory and simulation [10]. Practical evaluations are scarce due to the complexity of modifying existing network interface cards (NICs), the work of Dutta et al. [6] being an exception. We close this gap: in our work, we evaluate practical covert channels on the Wireless Open-Access Research Platform (WARP)[2] as well as off-the-shelf wireless NICs as legitimate receivers. Using WARP, we are able to utilize the same orthogonal frequency-division multiplexing (OFDM) modulation schemes as in 802.11a/g. Our covert channels can be easily adapted to OFDM-based wireless communication systems such as LTE, DVB-T, and upcoming standards like LTE Advanced. We aim at remaining compatible with the 802.11a/g standard and having little to no performance decrease on off-the-shelf receivers. Our contributions are as follows:

1) We analyze the IEEE 802.11a/g physical layer with

IEEE CNS 2015

Total Cost:

12 kEUR

(2 WARPs)

…

but can be much cheaper

Shadow Wi-Fi: Teaching Smartphones to Transmit Raw Signals and to Extract Channel State Information to Implement Practical Covert Channels over Wi-Fi

Anonymous Author(s)

ABSTRACT

Wi-Fi chips offer vast capabilities, which are not accessible through the manufacturers' official firmwares. Unleashing those capabilities can enable innovative applications on off-the-shelf devices. In this work, we demonstrate how to transmit raw IQ samples from a large buffer on Wi-Fi chips. We further show how to extract channel state information (CSI) on a per frame basis. As a proof-of-concept application, we build a covert channel on top of Wi-Fi to stealthily exchange information between two devices by prefiltering Wi-Fi frames prior to transmission. On the receiver side, the CSI is used to extract the embedded information. By means of experimentation, we show that regular Wi-Fi clients can still demodulate the underlying Wi-Fi frames. Our results show that covert channels on the physical layer are practical and run on off-the-shelf smartphones. By making available our raw signal transmitter, the CSI extractor, and the covert channel application to the research community, we ensure reproducibility and offer a platform for further innovative applications on Wi-Fi devices.

1 INTRODUCTION

Wi-Fi can be regarded as the de-facto standard for wireless

experimentation outside of lab environments. Furthermore, the device specific transmit and receive characteristics are maintained.

While SDRs provide for ample flexibility, the overhead for generating raw signals in software can be prohibitive. In contrast, the use of dedicated hardware for signal processing allows to efficiently modulate sequences of bits into wireless signals and vice versa. Especially on the receiver side, continuous calculations are required to perform correlations to detect incoming frames. Hence, for transforming Wi-Fi chips into SDRs, it is beneficial to use as many of the existing dedicated signal processing units as possible for both the sending as well as the receiving path. For the receiving path the following example illustrates this trade off. During regular reception, every Wi-Fi receiver needs to first extract channel state information (CSI) from the long-term training field (LTF) of a frame's preamble to cancel the effects of the wireless channel and to demodulate the transmitted data. If we aim at implementing applications that rely on CSI, they should avoid performing CSI extraction on a sample buffer on their own, but leverage the already existing information instead. As one part of our solution, we, hence, show how to extract CSI on a per-frame basis for use in advanced applications. For the sending path, the goal is to support sending

IEEE MobySys 2018

Total Cost:

600 EUR
(2 Smartphones)

Part 2: Bluetooth

Bluetooth

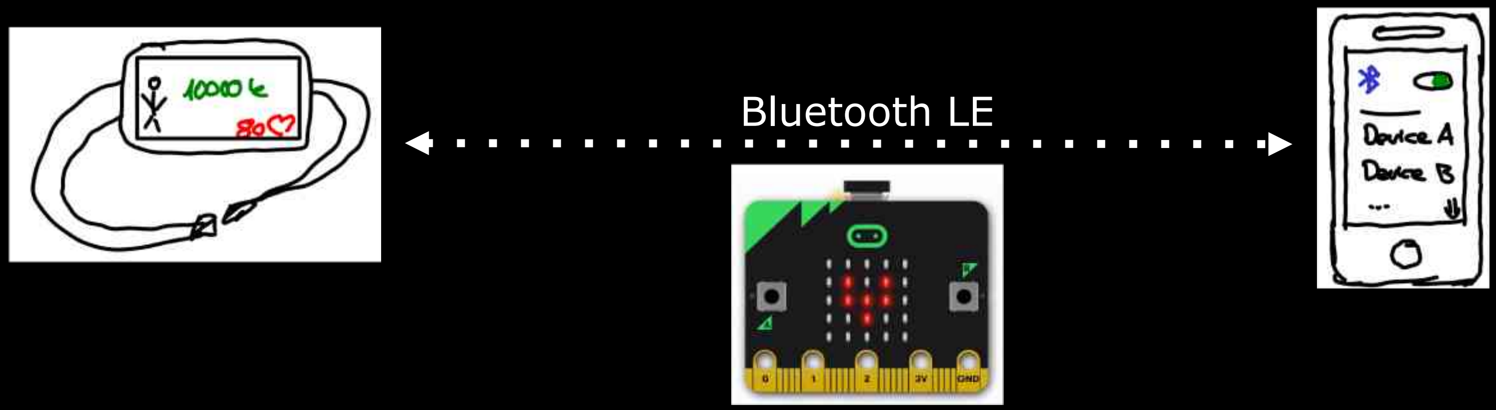
~~SMART~~

Bluetooth

Rather Complex,
Rather Closed,
Rather Legacy,
Tools Expensive

Can We Diagnose BT Lower Layers Using Off-the-Shelf Devices?

Standard Bluetooth Sniffing Setup (BLE)



- Use special (but very cheap) hardware, such as microbit/btlejack or Bluefruit LE.
- Successfully follow the **hopping** pattern and then **overhear the initial pairing** procedure to extract secret keys (*LE Legacy* pairing in Bluetooth 4.0 and 4.1).
- Maybe also active MITM (Bluetooth >= 4.2) required to get encryption keys…
- ...finally find out what these devices do on the lower layers!

Standard Bluetooth Sniffing Setup (Classic Bluetooth)

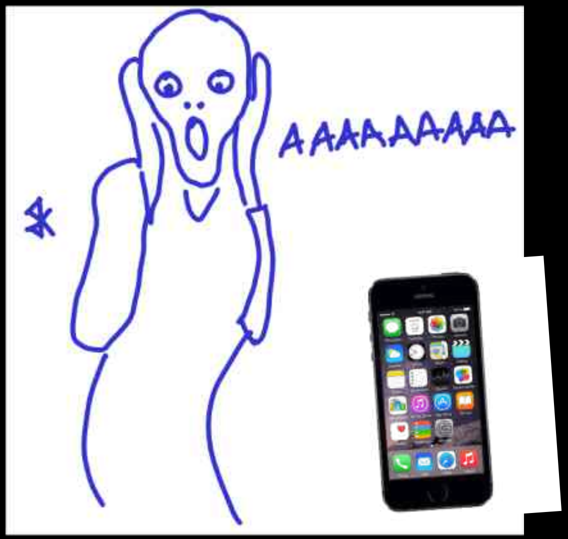
- Use special (> $10k) hardware, i.e. Ellisys.
- Open source solutions such as Ubertooth do not support encrypted traffic…
- Successfully follow the **hopping** pattern and then **modify the initial pairing** procedure to extract secret keys, active MITM required (Bluetooth >=4.0).
- If a stronger mode than “Just Works” is used, user needs to ignore the wrong numeric comparison.
- ...finally find out what these devices do on the lower layers!

Bluetooth Lower Layers: Security Perspective

Bluetooth lower layers are not well-tested.

- If you know the **MAC address**, you can **connect** to a device and get more information, i.e. which LMP version it is running (often equals the **firmware version**).

> *"Hi there, I'm a Broadcom Bluetooth 4.1 chip running an attackable LMP minor version of `0x2203`…"*



Making Bluetooth Lower Layers Accessible

In the shown Bluetooth **sniffing setup**, initial pairing must be overheard,
attacker needs to be in proximity during this pairing that only takes place once.
→ Very artificial setup, typically both **parties are aware of sniffing**,
at least if secure pairing modes are used.

Sniffing does not require MITM, **access on one of the devices** within a connection is sufficient to get all contents of a session despite hopping.
Lower layer traffic is not embedded within HCI (Host Controller Interface / layer 3) information.
→ Bluetooth **layer 1+2 cannot be observed** out of the box.

- **Modify firmware** of existing chipsets to monitor lower layer traffic.

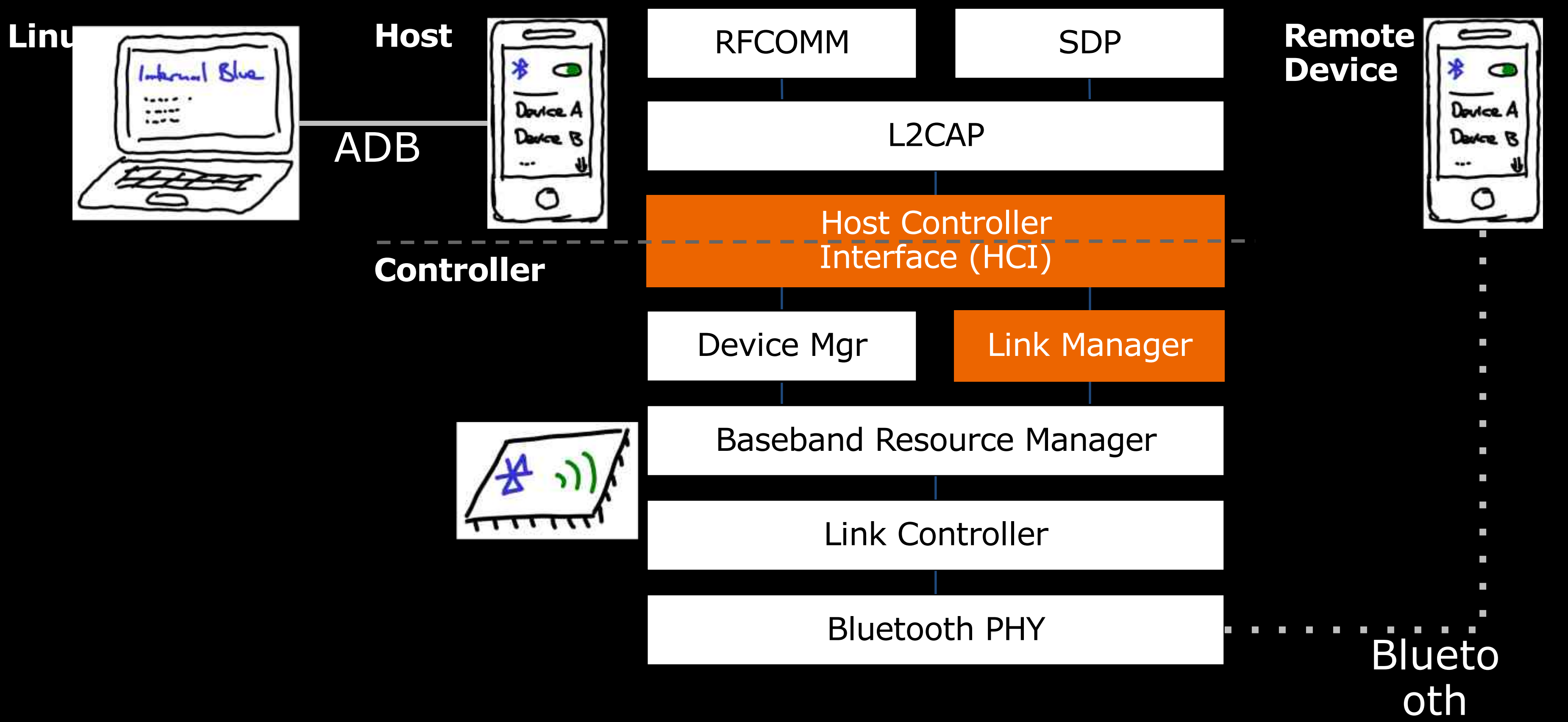
- SEEMOO already did this for monitor mode on Broadcom Wi-Fi chips.

nexmon

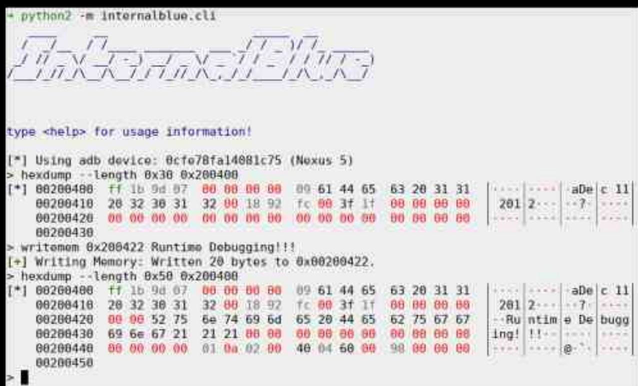
InternalBlue

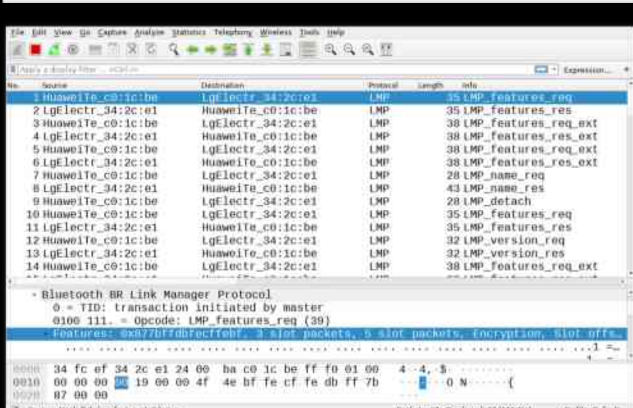
https://github.com/seemoo-lab/internalblue

Platform Overview



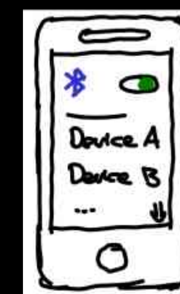
InternalBlue Based on Binary Patching





Vendor specific HCI (local)

Bluetooth



Modify firmware

- Nexus 5 (*BCM4339* firmware)
- Only LMP, no LCP

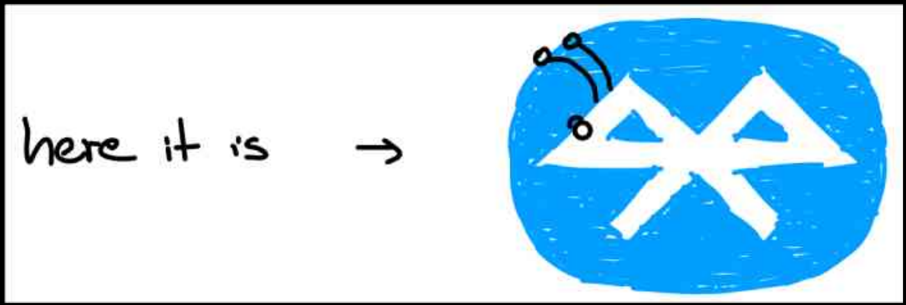
InternalBlue - A Deep Dive into Bluetooth Controller Firmware. Dennis Mantz.
https://media.ccc.de/v/2018-154-internalblue-a-deep-dive-into-bluetooth-controller-firmware

Platform Independence

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…

Finding Bugs in Lower Layers



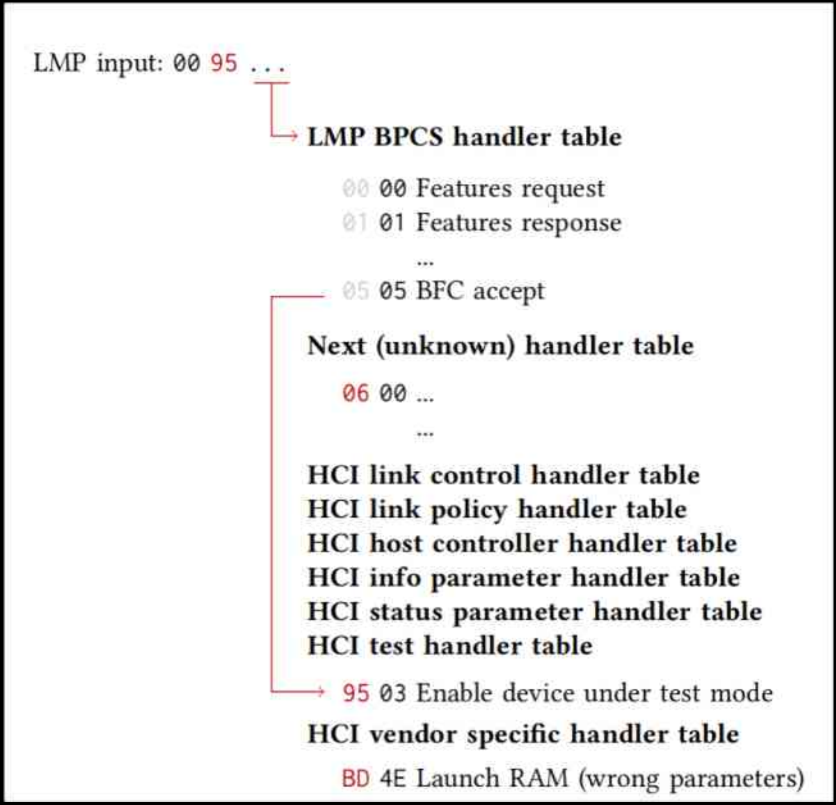
Uninitialized Encryption

- The attacker **initiates** SSP (Secure Simple **Pairing**) with the victim. Only the MAC address must be known for this, the device is not required to be discoverable. The victim is not required to take any action.
- Instead of completing the pairing, the attacker sends an `LMP_`**`start_encryption`**`_req`.
- Bluetooth crashes within the `bignum_xormod` on *Nexus 5*, your mileage might vary on other platforms.

- CVE-2019-6994

Handler Escalation Over the Air: HCI via LMP

- Missing parameter check in a vendor specific LMP handler...
- **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.
- Many more vulnerable devices, such as *iPhone 5...6, Macbook 2012...2017, Raspberry Pi 3*.



- CVE-2018-19860 / **BT-B-g0ne**

Do I have a device with a Broadcom chip?

- Platforms:
  Android 6 and 7, Lineage OS 14.1
  Linux/BlueZ (partially, some parts are work in progress)
  (macOS in progress)

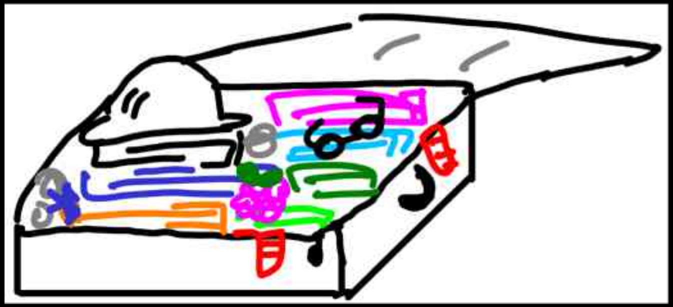
- Devices tested so far:
  Nexus 5, Xperia Z3 Compact, Samsung Galaxy Note 3 (*BCM4339*, best support)
  Nexus 6P, Samsung Galaxy S6, Samsung Galaxy S6 edge (also good support)
  Macbook Pro 2011+2016 (with Ubuntu)
  Raspberry Pi 3/3+
  Thinkpad T420, T430
  Asus USB Dongle
  Any Linux PC with the *CYW20735* evaluation board

Devices with Broadcom chips supported by *InternalBlue*. https://github.com/seemoo-lab/internalblue/tree/master/internalblue/fw/

All Those Bugs, Sure I Can Get A Software Patch

How does patching work?

- **Originally 128 patchram** slots inside **ROM**.
- 4 bytes per slot.
- Sufficient to **branch** (4 byte instruction) into **RAM**.
- **Limitation to 128 changes in program flow.**
- There is some free memory in RAM, but not enough to substitute huge functions.



- One more pair of socks will always fit into my bag!

How bad is it?

Device	Operating System	Slots
Raspberry Pi 3+/4	Raspbian July 2019 (includes “return version 5.0” patch)	128/128
iPhone 6	iOS 12.4 (released September 2014, will not get iOS 13)	128/128
iPhone SE	iOS 12.4 (released March 2016, still supported in iOS 13)	127/128
iPhone 7	iOS 12.4 (released September 2016)	192/192
Samsung Galaxy S8	Android 8, January 2019 (released April 2017)	250/256
iPhone 8/X/XR	iOS 12.4 (released November 2017)	240/256
Nexus 5	Lineage 14.1 / Android 6 (no more updates)	113/128
Samsung Galaxy S10/S10e/S10+	Android 9, June 2019 (released January 2019)	212/256

InternalBlue – Bluetooth Binary Patching and Experimentation Framework

Dennis Mantz
dmantz@seemoo.de
TU Darmstadt, Secure Mobile Networking Lab
Darmstadt, Germany

Jiska Classen
jclassen@seemoo.de
TU Darmstadt, Secure Mobile Networking Lab
Darmstadt, Germany

Matthias Schulz
mschulz@seemoo.de
TU Darmstadt, Secure Mobile Networking Lab
Darmstadt, Germany

Matthias Hollick
mhollick@seemoo.de
TU Darmstadt, Secure Mobile Networking Lab
Darmstadt, Germany

ABSTRACT

Bluetooth is one of the most established technologies for short range digital wireless data transmission. With the advent of wearables and the Internet of Things (IoT), Bluetooth has again gained importance, which makes security research and protocol optimizations imperative. Surprisingly, there is a lack of openly available tools and experimental platforms to scrutinize Bluetooth. In particular, system aspects and close to hardware protocol layers are mostly uncovered.

We reverse engineer multiple *Broadcom* Bluetooth chipsets that are widespread in off-the-shelf devices. Thus, we offer deep insights into the internal architecture of a popular commercial family of Bluetooth controllers used in smartphones, wearables, and IoT platforms. Reverse engineered functions can then be altered with our *InternalBlue Python* framework—outperforming evaluation kits, which are limited to documented and vendor-defined functions. The modified Bluetooth stack remains fully functional and high-performance. Hence, it provides a portable low-cost research platform.

InternalBlue is a versatile framework and we demonstrate its abilities by implementing tests and demos for known Bluetooth vulnerabilities. Moreover, we discover a novel critical security issue affecting a large selection of *Broadcom* chipsets that allows executing code within the attacked Bluetooth firmware. We further show how to use our framework to fix bugs in chipsets out of vendor support and how to add new security features to Bluetooth firmware.

ACM Reference Format:
Dennis Mantz, Jiska Classen, Matthias Schulz, and Matthias Hollick. 2019.
InternalBlue – Bluet
In *The 17th Annual I*
and Services (MobiSy
New York, NY, USA

1 INTRODU

Bluetooth, *the* sta
has been around f
1994. In the early d
hands-free speake
between devices
with the use of w
Energy (BLE) int
Bluetooth 5.0 and
networking and l
will play an impor

Bluetooth secu
selectively, which
Wi-Fi standard ov
attributed to the a
allow easy experir
shelf hardware. W
so-called *monitor*
soon implemente
Fi stack and the
(WEP) standard [
the firmware run

Inside Job: Diagnosing Bluetooth Lower Layers Using Off-the-Shelf Devices

Jiska Classen
Matthias Hollick
jclassen@seemoo.de
mhollick@seemoo.de
TU Darmstadt, Secure Mobile Networking Lab
Darmstadt, Germany

ABSTRACT

Bluetooth is among the dominant standards for wireless short-range communication with multi-billion Bluetooth devices shipped each year. Basic Bluetooth analysis inside consumer hardware such as smartphones can be accomplished observing the Host Controller Interface (HCI) between the operating system's driver and the Bluetooth chip. However, the HCI does not provide insights to tasks running inside a Bluetooth chip or Link Layer (LL) packets exchanged over the air. As of today, consumer hardware internal behavior can only be observed with external, and often expensive tools, that need to be present during initial device pairing. In this paper, we leverage standard smartphones for on-device Bluetooth analysis and reverse engineer a diagnostic protocol that resides inside Broadcom chips. Diagnostic features include sniffing lower layers such as LL for Classic Bluetooth and Bluetooth Low Energy (BLE), transmission and reception statistics, test mode, and memory peek and poke.

ACM Reference Format:

solutions, the latter having less stable implementations to follow encryption and hopping, such as Ubertooth and Bluefruit [1, 8, 11]. None of these run on the analyzed off-the-shelf device itself.

Most likely users of such a setup are aware of sniffing. We assume sniffers are installed to legally observe and analyze traffic between devices the analyst owns. Typical use cases are to inspect security of a proprietary smartphone app communicating with a proprietary IoT device, or to analyze performance on the Physical Layer (PHY) of a smartphone app and IoT firmware under development.

Android devices offer the BTSnoop Log, a developer option, which only covers HCI containing messages exchanged between the operating system and the Bluetooth chip. HCI is the Bluetooth middleware layer, but lower layer packets are not directly encapsulated within HCI and hence cannot be observed this way. In contrast, traces sniffed over the air with an external sniffer only contain Classic Bluetooth Link Manager Protocol (LMP) and BLE Link Control Protocol (LCP) packets. The toolchain implemented

ACM MobySys 2019, ACM WiSec 2019

https://www.youtube.com/watch?v=Cqzww5-K7VA

There is more …

InternalBlue - A Deep Dive into Bluetooth Controller Firmware

https://media.ccc.de/v/2018-154-internalblue-a-deep-dive-into-bluetooth-controller-firmware/related

Part 3: Airdrop

Slides at: https://www.usenix.org/conference/usenixsecurity19/presentation/stute

One Billion Apples' Secret Sauce: Recipe for the *Apple Wireless Direct Link* Ad hoc Protocol

Milan Stute
Secure Mobile Networking Lab
TU Darmstadt, Germany
mstute@seemoo.de

David Kreitschmann
Secure Mobile Networking Lab
TU Darmstadt, Germany
dkreitschmann@seemoo.de

Matthias Hollick
Secure Mobile Networking Lab
TU Darmstadt, Germany
mhollick@seemoo.de

ABSTRACT

Apple Wireless Direct Link (AWDL) is a proprietary and undocumented IEEE 802.11-based ad hoc protocol. Apple first introduced AWDL around 2014 and has since integrated it into its entire product line, including iPhone and Mac. While we have found that AWDL drives popular applications such as AirPlay and AirDrop on more than one billion end-user devices, neither the protocol itself nor potential security and Wi-Fi coexistence issues have been studied. In this paper, we present the operation of the protocol as the result of binary and runtime analysis. In short, each AWDL node announces a sequence of Availability Windows (AWs) indicating its readiness to communicate with other AWDL nodes. An elected master node synchronizes these sequences. Outside the AWs, nodes can tune their Wi-Fi radio to a different channel to communicate with an access point, or could turn it off to save energy. Based on our analysis, we conduct experiments to study the master election process, synchronization accuracy, channel hopping dynamics, and achievable throughput. We conduct a preliminary security assessment and publish

ACM Reference Format:

A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link

Milan Stute, *TU Darmstadt*
Sashank Narain, *Northeastern University*
Alex Mariotto, *TU Darmstadt*
Alexander Heinrich, *TU Darmstadt*
David Kreitschmann, *TU Darmstadt*
Guevara Noubir, *Northeastern University*
Matthias Hollick, *TU Darmstadt*

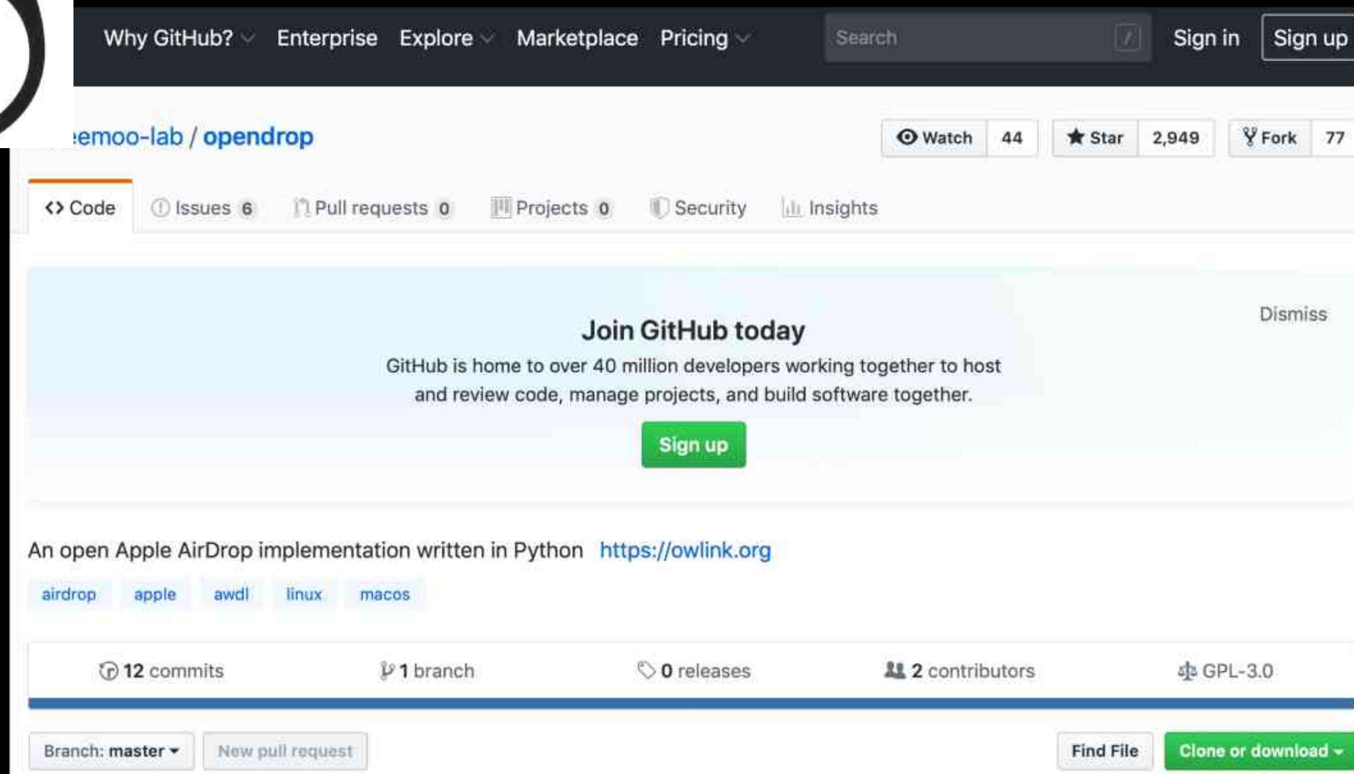
Abstract

Apple Wireless Direct Link (AWDL) is a key protocol in Apple's ecosystem used by over one billion iOS and macOS devices for device-to-device communications. AWDL is a proprietary extension of the IEEE 802.11 (Wi-Fi) standard and integrates with Bluetooth Low Energy (BLE) for providing services such as Apple AirDrop. We conduct the first security and privacy analysis of AWDL and its integration with BLE. We uncover several security and privacy vulnerabilities ranging from design flaws to implementation bugs leading to a man-in-the-middle (MitM) attack enabling stealthy modification of files transmitted via AirDrop, denial-of-service (DoS) attacks preventing communication, privacy leaks that enable user identification and long-term tracking undermining MAC address randomization, and DoS attacks enabling targeted or

peatedly discovered in Bluetooth [7], WEP [74], WPA2 [88], GSM [12], UMTS [57], and LTE [51], the lack of information regarding AWDL security is a significant concern given the increasing number of services that rely on it, particularly Apple's AirDrop and AirPlay. It is also noteworthy that the design of AWDL and integration with Bluetooth Low Energy (BLE) are (1) driven by optimizing energy and bandwidth and (2) the devices do not require an existing Wi-Fi access point (AP) with secure connections but are open to communicating with arbitrary devices, thus, potentially exposing various attack vectors.

We conduct the first, to the best of our knowledge, security analysis of AWDL and its integration with BLE, starting with the reverse engineering of protocols and code supported by analyzing patents. Our analysis reveals several security and

ACM MobyCom 2018, USENIX Sec 2019



OWLINK.ORG → OWL and OpenDrop

Summary + Conclusion

Wi Fi ®
~~CERTIFIED~~

Bluetooth
~~SMART~~

AirDrop

There is more:

Videos

- https://media.ccc.de/v/2018-154-internalblue-a-deep-dive-into-bluetooth-controller-firmware/related
- https://media.ccc.de/v/2018-124-pinky-brain-are-taking-over-the-world-with-vacuum-cleaners
- https://media.ccc.de/v/2018-123-nello-nicht-ganz-allein-zu-haus
- https://youtu.be/bKG8ZZq4oTo

Slides

- "Having fun with IoT: Reverse Engineering and Hacking of Xiaomi IoT Devices" https://dgiese.scripts.mit.edu/talks/DEFCON26/DEFCON26-Having_fun_with_IoT-Xiaomi.pdf

Thank You & Contact

SEEMOO
SECURE MOBILE NETWORKING

TECHNISCHE UNIVERSITÄT DARMSTADT

Prof. Dr.-Ing. Matthias Hollick
Department of Computer Science

SEEMOO
Mornewegstr. 32
64293 Darmstadt/Germany
mhollick@seemoo.tu-darmstadt.de

Phone +49 6151 16-25470
Fax +49 6151 16-25471
www.seemoo.tu-darmstadt.de