DEMO: Extracting Physical-Layer BLE Advertisement Information from Broadcom and Cypress Chips

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

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

ABSTRACT

Multiple initiatives propose utilizing Bluetooth Low Energy (BLE) advertisements for contact tracing and SARS-CoV-2 exposure notifications. This demo shows a research tool to analyze BLE advertisements; if universally enabled by the vendors, the uncovered features could improve exposure notifications for everyone. We reverse-engineer the firmware-internal implementation of BLE advertisements on *Broadcom* and *Cypress* chips and show how to extract further physical-layer information at the receiver. The analyzed firmware works on hundreds of millions of devices, such as all *iPhones*, the European *Samsung Galaxy S* series, and *Raspberry Pis*.

CCS CONCEPTS

• **Security and privacy** → **Systems security**; *Software security engineering*; Software reverse engineering; • **Networks** → **Application layer protocols**.

KEYWORDS

Bluetooth Low Energy, Advertisement, Broadcom, Cypress

ACM Reference Format:
Jiska Classen and Matthias Hollick. 2020. DEMO: Extracting Physical-Layer BLE Advertisement Information from Broadcom and Cypress Chips. In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '20), July 8–10, 2020, Linz (Virtual Event), Austria.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3395351.3401699

1 INTRODUCTION

Specification-compliant BLE advertisements contain little information that can be used for proximity estimation. We assume that the *Apple* and *Google* exposure notification Application Programming Interface (API) uses advertisements nonetheless [2], because they preserve battery and limit Remote Code Execution (RCE) risks. When receiving an advertisement, the chip measures the Received Signal Strength Indicator (RSSI) and includes it in the advertisement report before forwarding it to the operating system. This behavior is specification-compliant and does not need any further modifications [1, p. 2382]. There are multiple ways to enhance proximity measurements. For example, one could include the sender's transmission power in the advertisements [2, p. 4]. The transmission power directly influences the RSSI on the receiver. Not all chips can be set to the same transmission power, and thus, RSSI measurements need to be adjusted on the receiver.

We find that *Apple's* Bluetooth *PacketLogger* goes beyond the Bluetooth specification and displays the channel, antenna, and scan mode of each advertisement—but this information is always set to zero. Further investigation reveals that this information can be enabled with a firmware-internal global variable. As shown in Figure 1, the advertisements are not modified, as the information is measured on the receiver. In the most recent version of *InternalBlue* [3], this feature can be activated with the `adv` command.

Information about the advertisement channel is valuable, as the RSSI varies depending on the channel. The three advertisement channels are spread across the spectrum. Thus, typical interference sources such as a Wi-Fi access point can easily be evaded by blacklisting the interfered channel without taking statistics on multiple advertisements over time.

This demo is structured as follows. We explain the detailed reverse-engineering workflow to uncover proprietary features in Section 2. In Section 3, we conclude our findings.

2 REVERSE-ENGINEERING PROPRIETARY ADVERTISEMENT FEATURES

We use two methods to reverse-engineer vendor-specific additions to the BLE advertisement handler. First, we analyze which information the *PacketLogger* is using to display the channel in Section 2.1. Based on this information, we analyze the *Broadcom* and *Cypress* firmware to enable the output of this information in Section 2.2.

2.1 PacketLogger

The *PacketLogger* is included in the *Additional Tools for Xcode* on *macOS*. It features similar functions as *Wireshark* but is specifically designed for Bluetooth in the *Apple* ecosystem. Thus, it supports various proprietary protocols and features that are a helpful starting point for reverse-engineering. These protocols are otherwise

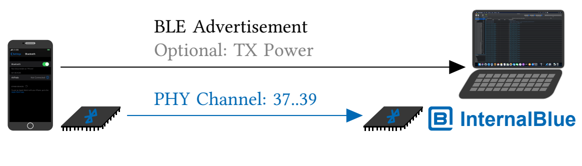


Figure 1: BLE advertisement physical-layer information.

Table 1: BLE advertisement report captured with *PacketLogger*, including the enhanced event type.

Apr 15 17:19:28:281	HCI Event	CA:FE:BA:BE:13:37	▼ LE - Advertising Report - 1 Report - Normal - Random - CA:FE:BA:BE:13:37 -56 dBm - Channel 37
			Parameter Length: 30 (0x1E) Num Reports: 0x01 Report 0 Event Type: Scan Mode: Normal Scan Mode - Channel 37 - Antenna: BT - Connectable Undirected Advertising (ADV_IND) ... RSSI: -56 dBm
Apr 15 17:19:28:281	HCI Event		▶ 00000000: 3E1E 0201 0001 3713 BEBA FECA ...

```
channel    = (event_type >> 4) & 7
antenna    = event_type & 0x80
scan_mode  = (event_type >> 3) & 3
```

Listing 1: Enhanced event type interpretation.

undocumented. We assume that *Apple* uses the *PacketLogger* for developing protocols and debugging and, thus, intentionally includes information about these protocols in their toolchain.

Table 1 shows a BLE advertisement as captured on *macOS Catalina* with *PacketLogger*. Note that by default all advertisements are displayed to be on channel 37, even though they are also received on channels 38 and 39. However, this indicates that there are some means of channel interpretation that are not included in specification-compliant advertisement reports [1, p. 2382].

The *PacketLogger* binary is located in `PacketLogger.app/Contents/Frameworks/PacketDecoder.framework/Versions/A/PacketDecoder`. It contains all the strings displayed within the *PacketLogger* and also most function names, which enables an analysis with *IDA Pro 7.2*. A search for the string `'Channel'`, as it can be seen in the *PacketLogger* output, leads to the function `leAdvertisingEventTypeString`. This function prints the antenna, channel, and scan mode, which are encoded into the upper half byte of the event type as shown in Listing 1. This is possible because the event type is 1 B, but Bluetooth specification only defines the values `0x00–0x04` [1, p. 2383]. The channel values `0x0–0x2` correspond to the Bluetooth channels 37–39. Thus, without this feature enabled on the chip, the channel is always interpreted as 37 by *PacketLogger*.

Note that the Bluetooth specification defines an extended advertisement report [1, p. 2402]. However, this report type also does not contain channel information.

2.2 Bluetooth Firmware

The *PacketLogger* reverse-engineering only indicates that additional event types exist. They still need to be enabled within the firmware. For the firmware analysis, we dump firmware from a selection of chips with *InternalBlue*. *WICED Studio 6.2* contains partial symbols for various *Cypress* chips as well as the *Broadcom BCM20703A2* chip that is in *MacBooks* produced in 2016 and 2017.

The global boolean `bEnhancedAdvReport` changes the behavior of the functions `_scanTaskRxHeaderDone` and `lculp_HandleScanReport`. This is explained in the next two paragraphs.

The firmware is organized in tasks that are called by the Bluetooth Core Scheduler (BCS). The scan task is responsible for receiving advertisements. In general, packet reception tasks are separated into receiving a header and receiving the according payload. The channel is already known when receiving a header, and thus, if `bEnhancedAdvReport` is set, additional information is copied from the raw packet data into `ulp_extraInfo`.

While tasks need to be finished within the strict timings of the Bluetooth clock, handlers asynchronously parse task data and pass it on to the host's operating system driver. In the case of an advertisement, this handler is `lculp_HandleScanReport`. The prefix `lculp` stands for link control in the ultra-low-power protocol, namely BLE. The advertisement handler copies the additional information into the `event_type` field if `bEnhancedAdvReport` is set.

Searching for the variable name `bEnhancedAdvReport` is only possible within a firmware that has partial symbols. However, symbols for most off-the-shelf devices are unknown. Nonetheless, hardware registers are mapped similarly over various firmware generations. Also, the architecture is typically an *ARM Cortex M*, and compiler options are similar. Thus, we can search for equal 4 B and 8 B snippets, which only return a few results within each firmware, to identify the scan task handler across multiple firmware versions. In our case, the scan task handler disassembly includes the line `mov.w r0, #0x650000`, which is a 4 B instruction represented by `0x00caf44f`, and that we used for manual binary diffing.

We find that the comparably old *Nexus 5* firmware with a build date from 2012 does not feature the `bEnhancedAdvReport` flag. However, the *Cypress* evaluation boards *CYW20719*, *CYW20735*, and *CYW20819* support it, as well as the *MacBook* 2016–2017 chip *BCM20703A2*, the *MacBook* 2017–2019 chip *BCM4364B0*, and the *Samsung Galaxy S10/S20* chip *BCM4375B1*. We assume that this feature was introduced by *Broadcom* around 2014 and all newer chips support it. While our *InternalBlue*-based setup can only enable this feature for research, *Broadcom* could also roll it out as a patch for a broad variety of devices.

3 CONCLUSION

The channel reporting within BLE advertisements can be enabled with a single flag on most *Broadcom* and *Cypress* chips. This makes the patch rather simple, as it only needs to set the flag to `0x01`. Future practical tests within contact tracing will show how much channel awareness can improve proximity measurements. Further physical-layer properties might also be available during advertisement reception. However, we did not spot any additional advertisement flags during the reverse-engineering process. Thus, more complex patches are required for further physical-layer insights.

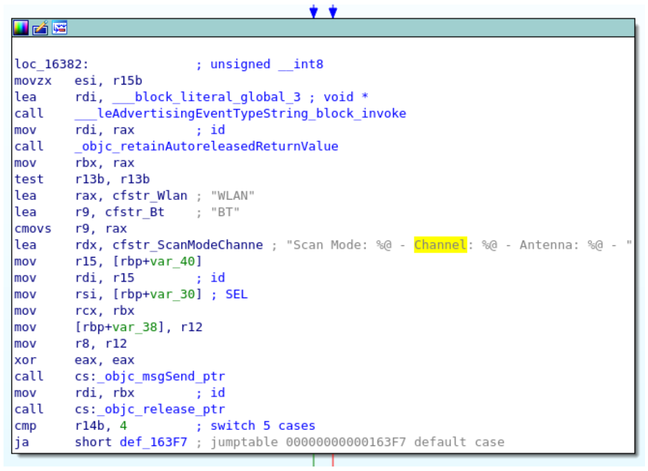
DEMO

This demo consists of an *InternalBlue* addition that enables the enhanced event type within advertisements on various *Broadcom* and *Cypress* chips. *InternalBlue* runs on *Android*, *Linux*, *macOS*, and *iOS*. Thus, the demo can be tested on various devices, as long as they have a supported chip.

InternalBlue is open-source and available on https://github.com/seemoo-lab/internalblue. The extended advertisements can be activated by running the command `adv`. After enabling the enhanced advertisements with `adv` and opening *Wireshark*, the event type field in advertisements contains the masked channel, antenna, and scan mode information.

To show this demo, we will provide a video of receiving the enhanced event type on a *Cypress CYW20819* evaluation board on *Linux*. On the *Linux BlueZ* stack, advertisements can be received by executing `hcitool lescan`.

Moreover, the video will show the workflow of reverse-engineering *PacketLogger* to identify the field where the channel information is included, as depicted in Figure 2. Then, we will search for this field in a partially symbolized *Cypress* firmware. This part of the demo will help other researchers to identify similar proprietary features.



```
loc_16382:              ; unsigned __int8
movzx   esi, r15b
lea     rdi, ___block_literal_global_3 ; void *
call    ___leAdvertisingEventTypeString_block_invoke
mov     rdi, rax        ; id
call    _objc_retainAutoreleasedReturnValue
mov     rbx, rax
test    r13b, r13b
lea     rax, cfstr_Wlan ; "WLAN"
lea     r9, cfstr_Bt    ; "BT"
cmovs   r9, rax
lea     rdx, cfstr_ScanModeChanne ; "Scan Mode: %@ - Channel: %@ - Antenna: %@ - "
mov     r15, [rbp+var_40]
mov     rdi, r15        ; id
mov     rsi, [rbp+var_30] ; SEL
mov     rcx, rbx
mov     [rbp+var_38], r12
mov     r8, r12
xor     eax, eax
call    cs:_objc_msgSend_ptr
mov     rdi, rbx        ; id
call    cs:_objc_release_ptr
cmp     r14b, 4         ; switch 5 cases
ja      short def_163F7 ; jumptable 00000000000163F7 default case
```

Figure 2: *PacketLogger* analysis in *IDA Pro 7.2*.

ACKNOWLEDGMENTS

We thank Alexander Heinrich for pointing out the issue that all advertisements are reported to be received on channel 37.

This work has been funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

REFERENCES

[1] Bluetooth SIG. 2020. Bluetooth Core Specification 5.2. https://www.bluetooth.com/specifications/bluetooth-core-specification.

[2] Apple Google. 2020. Exposure Notification, Bluetooth Specification, v1.1. https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-BluetoothSpecificationv1.1.pdf.

[3] Dennis Mantz, Jiska Classen, Matthias Schulz, and Matthias Hollick. 2019. InternalBlue - Bluetooth Binary Patching and Experimentation Framework. In *The 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '19)*. https://doi.org/10.1145/3307334.3326089