2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) | 978-1-6654-3572-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/DSN48987.2021.00049

WazaBee: attacking Zigbee networks by diverting Bluetooth Low Energy chips

Romain Cayre*‡, Florent Galtier*, Guillaume Auriol*†, Vincent Nicomette*†, Mohamed Kaâniche*, Géraldine Marconato‡
*CNRS, LAAS, 7 avenue du colonel Roche, F-31400
†Univ de Toulouse, INSA, LAAS, F-31400
‡APSYS.Lab, APSYS
Email: *firstname.lastname@laas.fr ‡firstname.lastname@airbus.com

***Abstract*—This paper discusses the security of wireless communication protocols of the Internet of Things (IoT) and presents a new attack targeting these protocols, called *WazaBee*, which could have a critical impact and be difficult to detect. Specifically, *WazaBee* is a pivotal attack aimed at hijacking *BLE* devices, commonly used in IoT networks, in order to communicate with and possibly attack through a different wireless network technology, considering protocols based on *802.15.4*, in particular *Zigbee*. We present the key principles of the attack and describe some real-world experiments that allowed us to demonstrate its practical feasibility. The attack takes advantage of the compatibility that exists between the two modulation techniques used by these two protocols. Finally, the paper briefly discusses possible countermeasures to mitigate the impact of this attack.**

I. Introduction

The massive and fast deployment of IoT devices raises major security concerns. Indeed, several wireless protocols designed to meet the requirements of the connected objects (low power consumption and low complexity protocol stacks) have been deployed in the recent years. Some of these protocols, such as *Zigbee*, *BLE* or *Thread*, are becoming widespread in IoT networks, and their vulnerabilities are actively studied, while other proprietary protocols are seldom analysed because their specifications are not available.

It is generally difficult to analyse security and to implement efficient protection mechanisms in IoT environments, mostly due to some specific characteristics of such environments, heterogeneity, decentralisation and dynamicity, which are difficult to handle from the security point of view.

This situation is problematic because it dramatically increases the attack surface exposed by IoT environments, and opens the opportunity to attackers to set up novel offensive strategies that are difficult to anticipate from a defensive point of view. Several vulnerabilities and attack scenarios have already been discovered in several widely used wireless protocols, like [1], [2], [3] or [4]. However, these studies generally focus on a single wireless technology.

In this paper, we focus on a specific threat that has been seldom studied from an offensive perspective until now. It takes advantage of the co-existence in the same environment of multiple wireless technologies. We investigate the possibility to divert the behaviour of a given device dedicated to a specific radio protocol, to make it communicate through another radio protocol not initially supported by the device, in order to perform malicious activities. The feasibility of such communications between heterogeneous protocols, has been explored in some previous works, commonly called *Cross Technology Communications*. However, existing solutions always assume a cooperation from the surrounding devices to allow such a transmission. This assumption is not realistic from an offensive perspective, which makes such attacks very unlikely. The approach investigated in this paper does not rely on this assumption, thus increasing its practical feasibility. Specifically, the proposed approach, called *pivoting attack*, allows to establish a communication channel between chips supporting *BLE*, which are embedded in many smart devices, and *IEEE 802.15.4* protocols such as *Zigbee*, in order to perform various types of attacks. The ubiquity and wide deployment of BLE devices make these attacks critical as the attack surface of Zigbee networks is significantly increased.

We believe the consequences of such *pivoting attacks*, would be critical, because 1) they open the possibility of new offensive strategies quite difficult to detect, because not considered as for now, and 2) they can be deployed at a large scale because the vast majority of connected objects embed at least one radio technology, that could possibly be remotely diverted. Furthermore, they exploit the use of *BLE* chips that are widely deployed in many environments, because they are embedded in the BLE-connected smartphones and smart devices. This makes the threat stealthier than attacks based on Software Defined Radios (SDR) which require bringing specific and detectable malicious devices inside the environment.

As an example, such a strategy can be used to perform covert channel attacks or to exfiltrate data to an illegitimate remote receiver by means of a corrupted *BLE* object, by communicating through a wireless protocol that is not supposed to be monitored in the targeted environment. It can also be used to perform traditional attacks targeting a radio protocol `RP1` (man in the middle, sniffing, spoofing, etc) from a device supporting another radio protocol `RP2` and that is not considered as a potential source of attack for the `RP1` protocol.

The main objective of this paper is to show the feasibility of such a novel attack strategy, called *WazaBee*, by considering the specific case of *BLE* and *802.15.4*-based (e.g. *Zigbee*) protocols. The main motivation is to increase awareness about

978-1-6654-3572-7/21/$31.00 ©2021 IEEE
DOI 10.1109/DSN48987.2021.00049

the need to develop efficient protection mechanisms to prevent and mitigate this type of attack which could have critical consequences on the security of IoT environments. The attack takes advantage of some characteristics of the *BLE* protocol to allow some *BLE* devices to communicate using *802.15.4*-based protocols not initially supported by these devices.

The consequences of this attack are critical because the vulnerability is not specific to some *BLE* chips but is rather related to the design and implementation of the underlying radio protocols. As a consequence, the attack is not implementation dependent and may potentially be used with the majority of *BLE* chips. In addition, the attack can be implemented easily which increases the level of the threat.

The contribution is threefold:

- We present the theoretical foundations of a novel pivoting attack, *WazaBee*, by demonstrating the compatibility that exists between the two modulation techniques used by *BLE* and *802.15.4*-based protocols and how it is possible to take advantage of this compatibility to carry out various critical attacks.
- We demonstrate its practical feasibility by implementing it on two different chips supporting the *BLE* protocol, and evaluate its performance in a realistic environment.
- We describe the implementation of two realistic attack scenarios exploring the feasibility of exploiting this attack in practice, from several heterogeneous hardware devices and under various assumptions.

The paper is organised as follows. Section II describes some related works allowing to implement pivoting attacks targeting wireless devices. Section III presents the physical layers of *BLE* and *802.15.4*-based protocols. Section IV outlines the theoretical basis of *WazaBee* attack, and also underlines some requirements related to the *BLE* protocol stack and how they may be addressed to implement the attack. Section V describes a practical implementation of *WazaBee* on two different chips and the associated benchmarks. Section VI presents two attack scenarios that we actually conducted in order to illustrate the impact of this attack. Section VII proposes several mitigations to this attack, and finally, the conclusion and future work are presented in section VIII.

II. Related work

This section briefly presents different attack strategies to carry out a pivoting attack. Firstly, the case of IoT devices supporting multiple radio protocols is discussed, then an overview of the few existing research works that considered such an attack on a device supporting a single specific radio protocol is presented.

A. Multi-protocol devices

A pivoting attack aims at taking advantage of the coexistence of multiple protocols in the same environment in order to compromise new objects. The most natural approach for this attack is to compromise an object supporting multiple radio communication protocols, allowing to perform the attack using the provided API. As an example, in [5], Bachy et al. compromise a smart-TV using *HbbTV* communication protocol, then use it to reconfigure the firewall embedded in the ADSL box using LAN protocols (Ethernet or WiFi).

Several hardware devices allow such attacks to be carried out. For instance, *Software Defined Radio* devices are designed for a generic purpose, allowing communications through multiple protocols, regardless of their modulation and frequency bands. However, so far, these devices are only used for prototyping and experimentation purposes.

There are also chips that integrate different wireless devices. For example, *B-L475E-IOT01A* [6], based on the *STM32L4* micro controller intended for IoT devices, supports multiple wireless protocols (such as *Bluetooth*, *WiFi* or *NFC*). Similarly, the *CC2652R* [7] from *Texas Instruments* is compliant to multiple radio technologies in the ISM band. The compromise of such a chip greatly facilitates the implementation of a pivoting attack targeting one of the wireless protocols supported by the chip. However, such chips are expensive and their use is quite specific, which limits their deployment in IoT networks.

B. Single-protocol devices

Since most connected objects only embed one wireless device, the practical implementation of a pivoting attack is much more complex. We are not aware of existing research specifically addressing this issue from an offensive perspective. However, some contributions explored related topics. The most relevant contributions are related to *Cross-Technology Communications* (*CTC*) solutions, that are aimed providing a communication system between two single-protocol devices supporting heterogeneous wireless communication protocols. However, to our knowledge these contributions did not investigate the use of this technology in security or in an offensive perspective. There are two main categories of *CTC*, named *Packet-level CTC* and *Physical layer CTC*.

The *Packet-level CTC* approach relies on some information linked to the packets. As an example, K. Chebrolu et al. use packet duration in order to transmit data [8], while the *FreeBee* [9] approach by S. Min Kim is based on the time interval between beacon frames. From an offensive perspective, these approaches could be interesting to exfiltrate some data, but they are not relevant for pivoting attacks. Other limitations, such as a low data throughput, are inherent to these approaches and hamper their practical use.

Physical layer CTC approaches consist in emulating a technology using the signal generated by another one. As an example, Z. Li et al. simulate a *Zigbee* frame using a WiFi transceiver [10]. Similarly, W. Jiang et al. have presented an approach named *BlueBee* [11], allowing to simulate *Zigbee* frames using a *BLE* transceiver, and another approach called *XBee* [12], enabling to receive *Zigbee* frames from a *BLE* receiver. However, these solutions have major limitations that prevent their use in an offensive perspective. As an example, the selection of a *Zigbee* channel by *BlueBee* is based on the channel hopping algorithm of *BLE* connected mode, so it requires to establish a *BLE* connection with another *BLE* device. Similarly, adding a specific identifier before the data

included in the frame is needed in order to receive a *Zigbee* frame using *XBee*, so it requires the cooperation of the *Zigbee* transmitter. These constraints can be easily addressed if the use of *CTC* is legitimate and deliberate, however they prevent the use of these solutions in a context of attack and especially for pivoting attacks. Our approach overcomes these limits and provides a reliable two-way *CTC* that doesn't require the cooperation of other devices: as a consequence, it may be used in an offensive context.

The *Packet-in-Packet* strategy [13], proposed by T. Goodspeed et al. consists in encapsulating a complete radio frame into an application-level *payload*: a misidentification of the beginning of the encapsulating frame by the receiver (e.g., due to interferences causing bitflips during the demodulation) can lead to the interpretation of the encapsulated frame. This strategy is particularly interesting for bypassing software checks performed at the protocol layer, and may thus allow attackers to access and control the lower layers of the radio device. The authors highlight a possible use of this attack to perform a pivoting attack, e.g., to inject radio traffic corresponding to a wireless protocol different from the protocol natively supported by the radio device, under certain specific conditions. However, this strategy can only be applied to a limited number of protocols, and can only be achieved if the modulations used have similar characteristics (frequency bands, bandwidth, etc). For instance, M. Millian and V. Yadav discuss the possibility of encapsulating *802.15.4* traffic into 802.11 frames [14]. However, they stress the difficulty of such a strategy due to the differences between the two technologies.

T. Goodspeed has also discovered a vulnerability in the *nRF24L01+* chip, that facilitates sniffing and frame injection on a set of protocols (such as *Bluetooth Low Energy* or *Enhanced ShockBurst*) using *Gaussian Frequency Shift Keying* modulation. He was able to divert the use of a register dedicated to the address selection to select an arbitrary preamble [15]. Exploiting this vulnerability allowed him to add a promiscuous mode for the *Enhanced ShockBurst*, which is not natively supported by the chip. However, it is also possible to divert the use of this register to detect specific preambles used by different wireless technologies, as long as similar modulations and bit rates are used. This vulnerability has been used by M. Newlin to develop a firmware aiming to add advanced sniffing capabilities for the *Enhanced ShockBurst* and *Mosart* protocols to the *nRF24* chip [16].

D. Cauquil has also disclosed a similar vulnerability in other *Nordic Semiconductors* chips [17], and has developed a similar tool for the *nRF51*. He was then able to implement communication primitives for a proprietary protocol not initially supported by the chip, allowing it to control a mini-drone [18]. An implementation of these primitives has been integrated into the *radiobit* [19] project.

These research works present some first techniques and experimental results that illustrate the practical feasibility of pivoting attacks targeting wireless protocols. However, these techniques have several limitations which strongly restrict their use: they require an active cooperation of other devices, or the modulation of the native protocol and the pivoting protocol must be similar and sometimes depend on the use of specific chips (such as *Nordic SemiConductors nRF24* and *nRF51* chips). Our main contribution is to present a pivoting attack strategy that overcomes some of these constraints, allowing the implementation of communication primitives targeting a wireless technology using a modulation different from the one natively supported by the chips that doesn't require the cooperation of other nodes, and that could possibly be generalised to multiple hardware devices from different manufacturers.

III. Overview of wireless protocols

In this section, we introduce some definitions that are useful to understand the pivoting attack presented in this paper. Then, we briefly present some background information about the *BLE* and *Zigbee* protocols lower layers.

A. Digital modulation

Digital modulation is defined as the process of transforming a digital signal (the modulating signal) to adapt it to the transmission channel. This transformation consists in modifying the characteristics of a sine wave, called a carrier, according to the data to be transmitted. The resulting signal is called the modulated signal.

The modulated signal is defined by the following equation:

$$s(t) = A(t)\cos(2\pi f_c t + \varphi(t)) \tag{1}$$

where $A(t)$, f_c, and $\varphi(t)$ represent the amplitude, frequency and phase of the signal, respectively.

The state of a modulated signal at a given time can be represented by a vector in the complex plan: the norm of the vector represents the amplitude of the signal, while its argument corresponds to its phase. Indeed, formula (1) can be written as follows:

$$s(t) = I(t)\cos(2\pi f_c t) - Q(t)\sin(2\pi f_c t) \tag{2}$$

- $I(t) = A(t)\cos(\varphi(t))$ "In-phase component"
- $Q(t) = A(t)\sin(\varphi(t))$ "Quadrature component"

Note also that equation (2) demonstrates that it is possible to control the instantaneous phase, the instantaneous frequency and the amplitude of a carrier wave by manipulating the amplitude of I and Q signals. This property is the basis of a so-called I/Q modulator.

B. Bluetooth Low Energy (BLE)

The *Bluetooth Low Energy* protocol, or *BLE*, is a simplified variant of the *Bluetooth* protocol, introduced in version 4.0 of the *Bluetooth* specification [20]. In particular, it is optimised for energy saving and is commonly used in IoT networks, due to its low complexity and its wide deployment. It is also supported by default by most smartphones and computers.

In this paper, we focus on the lower layers of the protocol, notably the physical layer. The physical layer of the protocol (*PHY* layer) describes a single packet format composed of the following fields:

- **Preamble:** one byte field corresponding to series of alternating bits (0x55), used to synchronise the receiver at the start of the frame,
- **Access Address:** 4 bytes field, allowing to identify a specific connection or an advertisement,
- **Protocol Data Unit (PDU):** field of variable size made up of a link-layer header (*LL Header*) and the data to be transmitted,
- **Cyclic Redundancy Check (CRC):** 3 bytes field for integrity checking based on cyclic redundancy code.

When a frame is transmitted, the data from the upper layers is prefixed with a header by the link layer (*LL*), and is encapsulated into the PDU field. The corresponding CRC is appended to the PDU. A transformation called *whitening* is then applied, allowing the generation of a pseudo-random sequence, in order to avoid the presence of long repeated sequences of 1 or 0, which could alter the transmission of the modulated signal. Finally, the preamble and the Access Address are included before the PDU and the frame is then processed by the modulator.

The physical layer of the *BLE* protocol is based on a frequency modulation, called *Gaussian Frequency Shift Keying* (*GFSK*), operating in the ISM band (from 2.4 to 2.5 GHz). It is a variant of the *2-Frequency Shift Keying* (*2-FSK*) modulation in which a gaussian filter is applied to the modulating signal to avoid abrupt changes in frequency upon symbol changes.

A *2-FSK* modulation consists in encoding two symbols (0 and 1 for binary data) by two different frequencies defined by the following formulas:

$$F_0 = f_c - \Delta f = f_c - \frac{m}{2T_s} \tag{3}$$

$$F_1 = f_c + \Delta f = f_c + \frac{m}{2T_s} \tag{4}$$

- f_c is the frequency of the carrier, called central frequency,
- Δf is the modulation deviation (defined as the lag between the frequency encoding the symbol and the frequency of the carrier),
- m is the modulation index (a value between 0 and 1 characterizing the modulation),
- T_s is the symbol duration (the inverse of the data rate).

This modulation provides a modulated signal whose signal envelope amplitude is constant and its phase is continuous over time. In addition, the instantaneous phase $\varphi(t)$ and the instantaneous frequency $f(t)$ are linked as follows:

$$f(t) = \frac{1}{2\pi}\frac{d\varphi(t)}{dt} \tag{5}$$

Thus, the variation of instantaneous frequency can be inferred by observing the direction of rotation of the instantaneous phase: an increase in frequency (encoding the value 1) will cause a counter-clockwise rotation of the phase, while a decrease in frequency (encoding the value 0) will cause the phase to rotate clockwise. Such a modulation can thus be represented in the complex plan by observing the direction of rotation of the phase, as illustrated in Figure 1.

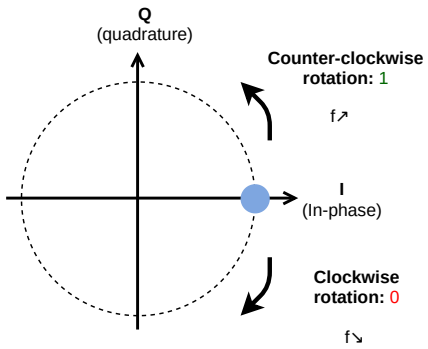

Fig. 1: I/Q representation of a *2-FSK* modulation

BLE specification states that the modulation index must be set between 0.45 and 0.55. The symbol duration T_s depends on the mode in use. Indeed, the first versions of the specification required a data rate of 1 Mbit/s (i.e., $T_s = 10^{-6}s$). However, version 5 introduced two new operating modes for the physical layer: *LE Coded*, that is out of the scope of the paper, and *LE 2M*, operating at 2 Mbits/s (i.e., $T_s = 5 \times 10^{-7}s$).

The central frequency depends on the communication channel. Indeed, the specification proposes 40 communication channels in the ISM frequency band (from 2.4 to 2.5 GHz), each with a bandwidth of 2 MHz. Three of these channels (37, 38 and 39) were initially dedicated to the broadcasting of announcement messages (advertising channels) while the other 37 channels were dedicated for data exchange in connected mode (data channels). However, the addition of new modes, *LE Coded* and *LE 2M*, introduces the possibility to use data channels as secondary advertising channels. Each channel being identified by a number $k \in [0..39]$. The channels 37, 38 and 39 respectively use the frequencies 2402, 2426 and 2480 MHz. The other channels, from 0 to 36, are spaced of 2MHz from 2404MHz skipping those frequencies.

C. Zigbee

Zigbee is one of the most widespread wireless protocols in *IoT* networks. Its low power consumption, the low cost of radio devices and the ability to build complex topologies make it particularly attractive for IoT systems. It is compliant with the IEEE *802.15.4* standard [21] which defines the physical and link layers. Its specification mainly describes the upper layers of the protocol stack (i.e., the network and application layers). In this paper, we focus on the lower layers, and more specifically on the *802.15.4* standard physical layer. This layer (called *PHY*) defines the format of the frames (named *Physical Protocol Data Unit*, or *PPDU*), as follows:

- **Preamble:** 4 consecutive null bytes field (0x00 0x00 0x00 0x00), used to synchronise the receiver with the beginning of the frame,
- **Start of Frame Delimiter (SFD):** one byte field of value 0x7A, indicating the beginning of the frame,
- **Length (PHR):** one byte field encoding the size in bytes of the Protocol Service Data Unit,

- **Protocol Service Data Unit (PSDU):** field of variable length, encapsulating the frame at link layer (or *MAC*). This frame is composed of a header, (*MHR*), the data to be encapsulated, transmitted by the upper layers, as well as a two bytes field, the Frame Check Sequence (*FCS*), used to check the integrity of the received frame.

Acccording to the *802.15.4* standard, a spread spectrum technique (*Direct Sequence Spread Spectrum* or *DSSS*) is applied to the generated frame before it is processed by the modulator. Each byte is split into two blocks of 4 bits, the *Least Significant Bits (LSB)* and the *Most Significant Bits (MSB)*. Each of these blocks is then substituted by a pseudo-random sequence of 32 bits, called *PN* sequence (*Pseudorandom Noise*) according to the correspondences presented in Table I. The bits of this sequence are also called *chips*.

TABLE I: Block/PN sequence correspondence table

Block ($b_0b_1b_2b_3$)	PN Sequence (c_0c_1 ... $c_{30}c_{31}$)
0000	11011001 11000011 01010010 00101110
1000	11101101 10011100 00110101 00100010
0100	00101110 11011001 11000011 01010010
1100	00100010 11101101 10011100 00110101
0010	01010010 00101110 11011001 11000011
1010	00110101 00100010 11101101 10011100
0110	11000011 01010010 00101110 11011001
1110	10011100 00110101 00100010 11101101
0001	10001100 10010110 00000111 01111011
1001	10111000 11001001 01100000 01110111
0101	01111011 10001100 10010110 00000111
1101	01110111 10111000 11001001 01100000
0011	00000111 01111011 10001100 10010110
1011	01100000 01110111 10111000 11001001
0111	10010110 00000111 01111011 10001100
1111	11001001 01100000 01110111 10111000

PN sequences are then provided as input of the modulator. The physical layer of the *802.15.4* standard is based on a phase modulation called *Offset Quadrature Phase Shift Keying* (or *O-QPSK*) with half sine pulse shaping in the ISM band. This modulation corresponds to a variant of the *Quadrature Phase Shift Keying* phase modulation, which consists in encoding the binary input information by modulating the phase of the carrier. Four phase values are used to transmit four symbols, each symbol being composed of two consecutive bits. In the specific case of *Zigbee* and *O-QPSK* modulation, each symbol is composed of 2 chips.

To generate a *802.15.4* compliant signal, it is necessary to independently control the In-phase and Quadrature components used to modulate the even bits and the odd bits, respectively. The first step consists in transforming the binary message to be modulated into two sequences of half sine pulses of duration $T_s = 2T_b$ (where T_b corresponds to half the duration of a symbol): a 1 bit is encoded by a positive half sine pulse while a 0 is encoded by a negative half sine pulse. As a result, $I(t)$ is a sequence of half sine pulses representing the even bits while $Q(t)$ is a sequence of half sine pulses representing the odd bits. The Quadrature component is also temporally delayed of T_b in order to avoid some drawbacks linked to *QPSK* modulation.

Then, the modulated signal $s(t)$ can be generated from the In-Phase and Quadrature signals using formula 2. This

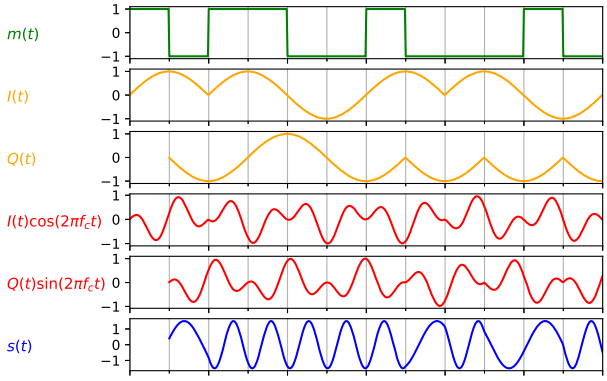

Fig. 2: Temporal representation of *O-QPSK* modulated signal with half sine pulse shaping

modulation generates a signal with continuous phase jumps, evolving linearly during the period of a T_b chip: the instantaneous phase of the modulated signal thus becomes continuous as a function of time and the amplitude of the signal's envelope remains constant, as shown in figure 2. Thus, at each sampling instant, there are only two possible transitions to the following state: $+\frac{\pi}{2}$ and $-\frac{\pi}{2}$. The transition to be made depends on: 1) the value of the previous bit, 2) whether an even bit or an odd bit is currently modulated, and 3) the current state. For instance, if the current state corresponds to symbol 11 and if one wishes to modulate an odd bit set to 1, one will take the transition to state 01, which will cause a linear increase of $+\frac{\pi}{2}$ in the instantaneous phase during the period T_b. The constellation diagram is represented in Figure 3.

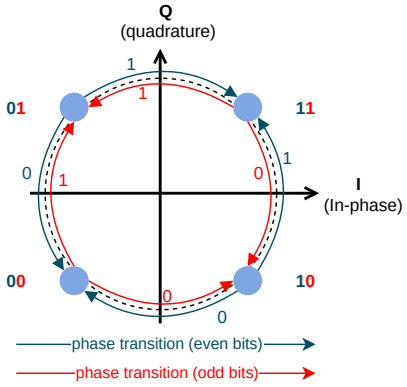

Fig. 3: I/Q representation of *O-QPSK* modulation with half sine pulse shaping

The *802.15.4* standard specification indicates a data rate of 2 Mchips/s in the ISM band, which corresponds to $T_b = 5 \times 10^{-7} s$. Consequently, the data rate corresponding to the bits of the *PPDU* before the substitution of the PN sequences corresponds to 250 kbits/s. The carrier wave frequency (called

central frequency as in *BLE*) depends on the communication channel used. The *802.15.4* standard proposes use of 16 communication channels, from 11 to 26 with a 2 MHz bandwidth per channel. Two consecutive channels are spaced 3 MHz apart. The following formula gives the relationship between the central frequency f_c (in MHz) and the channel number k (from 11 to 26):

$$f_c = 2405 + 5(k - 11) \quad (6)$$

IV. The WazaBee attack

This section describes the *WazaBee* attack and its architecture, which aims to divert the use of the radio device embedded in the *BLE* chip in order to send and receive *802.15.4* frames (in particular *Zigbee* frames). We first describe the attack principle and its theoretical foundations, then we detail the various requirements related to the legitimate operation of the chip that must be taken into account for the attack to be successful and we provide some solutions to fulfil these requirements.

A. Assumptions

We consider that the attacker has already compromised a *BLE* chip and is able to run arbitrary code on it. This chip compromise may be performed using various techniques, such as network attacks (e.g. attack of an *Over The Air* update process [22]), exploitation of vulnerabilities inherent to the object itself and its firmware allowing some remote code execution [23], [24], or physical attacks allowing to flash the device [25]. This compromise is considered as a prerequisite to the *WazaBee* attack, and is out of the scope of this paper.

B. Attack overview

The *Wazabee* attack relies on the existence of a close relationship between *GFSK* and *O-QPSK*, the modulations used by *BLE* and *Zigbee* protocols. The following subsections explain how to switch from one modulation to another.

1) From GFSK to MSK modulation: As explained in previous sections, *BLE* uses a *Gaussian Frequency Shift Keying* modulation with a modulation index m between 0.45 and 0.55. This characteristic allows us to assimilate the *BLE* modulation to a specific case of *GFSK*, called *GMSK* (*Gaussian Minimum Shift Keying*) with a modulation index $m = \frac{1}{2}$. The signal generated by a *GMSK* modulation has a constant amplitude and a phase evolving continuously over time. Moreover, a *GMSK* modulation is a *MSK* modulation (*Minimum Shift Keying*) whose modulating signal is shaped by a Gaussian filter. If we neglect the effect of the Gaussian filter, *BLE* modulation can be assimilated to *MSK* modulation, which changes linearly and continuously the phase of $-\frac{\pi}{2}$ when modulating a 0-bit and of $+\frac{\pi}{2}$ when modulating a 1-bit.

2) From MSK to O-QPSK modulation: As explained in previous sections, an *O-QPSK* modulation with half sine pulse shaping shares with the *MSK* modulation the property of a constant amplitude and a continuous phase. Moreover, the modulation of each bit generates a $\pm\frac{\pi}{2}$ continuous and linear phase transition. Both *MSK* and *O-QPSK* modulations are thus very close. In a more formal way, the research work of [26] shows the theoretical equivalence between *MSK* and *O-QPSK* with half sine pulse shaping, if an encoding strategy is purposely chosen, such as $T_{s(MSK)} = T_{b(OQPSK)}$.

3) From BLE to Zigbee: If we neglect the effect of the Gaussian filter (which will result in more progressive phase transitions), we can make the hypothesis that *BLE* modulation can be approximated by a *MSK* modulation, which is close to the *O-QPSK* modulation used by the *Zigbee* devices. To sum up, we can make the following hypotheses:

- It should be possible to control the input message of a *GFSK* modulator compatible with the *BLE* specification to generate a modulated signal corresponding to a binary sequence that can be interpreted by a *O-QPSK* demodulator (with half sine pulse shaping) compatible with the *802.15.4* standard.
- An arbitrary message modulated by an *O-QPSK* modulator (with half sine pulse shaping) compatible with the *802.15.4* standard should generate a modulated signal corresponding to a binary sequence interpretable by a *GFSK* demodulator compatible with *BLE* specification.

In the following, we outline how these hypotheses can be verified.

C. Correspondence table generation

The first problem to be addressed consists in establishing a correspondence table between the PN sequences used by the *802.15.4* standard (which results from an interpretation of the signal as a phase modulation, the *O-QPSK* with half sine pulse shaping) and their interpretation by a *MSK* frequency modulation. From this correspondence table, it will then be possible to build a binary sequence to be provided as input to a *BLE* compliant modulator to generate a modulated signal close to the one expected by a *802.15.4* demodulator, but also possible to interpret an *802.15.4* frame as a frequency modulated signal that can be demodulated by a *BLE* demodulator.

The generation of *MSK* sequences consists in encoding each phase transition of the *O-QPSK* modulation with a 1-bit if it corresponds to a counter-clockwise rotation of the vector representing the signal in the complex plan ($+\frac{\pi}{2}$ increase of the instantaneous phase) or with a 0-bit if it corresponds to a clockwise rotation ($-\frac{\pi}{2}$ decrease in the instantaneous phase).

Algorithm 1 illustrates this encoding technique. By applying this algorithm to the 16 PN sequences, it is possible to build the correspondence table.

Let us note that a sequence of length n encoded in *O-QPSK* has an equivalent of length $n - 1$ encoded in *MSK*, because this one represents the transitions between phases.

D. Requirements

The practical implementation of such an attack requires to take into account a number of requirements, related to *BLE* physical layer characteristics described in previous sections. Our objective is to implement primitives to send and receive *802.15.4* frames on a chip supporting *BLE* 5.0 specification. For that purpose, we have to control the following elements:

Output: $mskSequence[31]$
Input: $oqpskSequence[32]$;
$evenStates[4] \leftarrow \{1, 0, 0, 1\}$;
$oddStates[4] \leftarrow \{1, 1, 0, 0\}$;
$currentState \leftarrow 0$;
$i \leftarrow 1$;
while $i < 32$ **do**
  if i *is odd* **then**
    if $oqpskSequence[i]$ = $oddStates[(currentState + 1) \mod 4]$ **then**
      $currentState \leftarrow (currentState + 1) \mod 4$;
      $mskSequence[i - 1] \leftarrow 1$;
    else
      $currentState \leftarrow (currentState - 1) \mod 4$;
      $mskSequence[i - 1] \leftarrow 0$;
    end
  else
    if $oqpskSequence[i]$ = $evenStates[(currentState + 1) \mod 4]$ **then**
      $currentState \leftarrow (currentState + 1) \mod 4$;
      $mskSequence[i - 1] \leftarrow 1$;
    else
      $currentState \leftarrow (currentState - 1) \mod 4$;
      $mskSequence[i - 1] \leftarrow 0$;
    end
  end
  $i \leftarrow i + 1$;
end

Algorithm 1: Algorithm of a PN sequence conversion

- **Data rate:** the duration of one symbol encoded in the *MSK* modulation must be identical to the duration of one bit encoded by the *O-QPSK* modulation, i.e., $T_{s(MSK)} = T_{b(OQPSK)}$. It is thus necessary to configure the modulator and the demodulator used by the chip in order to use a 2 Mbits/s data rate, the same data rate as the 2Mchips/s of the *802.15.4* standard,
- **Central frequency:** *BLE* used channel central frequency must match the frequency of the *Zigbee* channel,
- **Modulator input:** to implement an emission primitive, it is necessary to control (directly or indirectly) the data sent to the modulator of the chip, in order to be able to provide the PN sequences encoded in *MSK*,
- **Demodulator output:** to implement a reception primitive, it is necessary to detect the reception of a *802.15.4* frame and to retrieve (directly or indirectly) the data output from the demodulator of the chip.

Controlling the data rate is quite easy since the introduction in version 5.0 of a new *LE 2M* mode for *BLE* physical layer, which allows to use a data rate of 2Mbits/s, which perfectly corresponds to our needs. Therefore, it should be possible to satisfy this first requirement on any chip implementing version 5.0 of the Bluetooth specification.

The second requirement is to control the *BLE* central frequency according to the *Zigbee* channel targeted by the attack. Several solutions can be implemented to solve this problem according to the possibilities offered by the chip and the available API. Indeed, most of the chips supporting *BLE* version 5.0 allow to arbitrarily choose a frequency in the 2.4 to 2.5 GHz band, in this case, it is possible to directly select the central frequency of the targeted *Zigbee* channel. If the chip does not allow such a functionality, it is then possible to select a *BLE* channel whose central frequency corresponds to a *Zigbee* channel: only a subset of the *Zigbee* channels will then be available, those defined in the Bluetooth specification. These channels are indicated in Table II. Such diversion of the use of *BLE* channels is made possible because both *Zigbee* and *BLE* channels share the same characteristics (2MHz bandwidth) and because the *LE 2M* mode allows the use of data channels as secondary advertising channels, thus allowing a direct transmission or reception on the channel via the advertising mode (the connected mode indeed implements a channel hopping algorithm that complicates a lot the implementation of this attack and requires the cooperation of another device).

TABLE II: Zigbee and BLE common channels

Zigbee Channels	*BLE* Channels	central frequency (f_c)
12	3	2410 MHz
14	8	2420 MHz
16	12	2430 MHz
18	17	2440 MHz
20	22	2450 MHz
22	27	2460 MHz
24	32	2470 MHz
26	39	2480 MHz

The third requirement is to be able to control the data provided as an input to the chip modulator: an arbitrary succession of PN sequences (encoded in *MSK*) must be provided in order to implement a transmission primitive. The main difficulty is related to the *whitening* process, which applies a transformation algorithm on the data to be transmitted, thus modifying the frame before its modulation. This functionality can be disabled on some chips, thus allowing a direct control on the bits transmitted to the modulator. However, even in the absence of this possibility, the *whitening* algorithm is reversible because it is based on a simple linear feedback shift register: it is thus possible to build a sequence of bits which, once the transformation has been applied, corresponds to the PN sequences, by first applying the de-whitening algorithm on the sequences that must be transmitted. In these two cases, the PN sequences to be transmitted to generate the expected *802.15.4* frame can be encapsulated in the payload of a *BLE* packet, for instance in the advertising data (the *LE 2M* mode allows the transmission of large advertising packets with a payload of up to 255 bytes).

The fourth requirement, which is crucial to build a reception primitive, is to detect *802.15.4* frames and to decode these frames to retrieve the symbols corresponding to PN sequences. For that purpose, the *Access Address* of the *BLE* chip must be configured: this Access Address is used as a pattern to detect a legitimate *BLE* frame. The *Access Address* value can be set with the PN sequence (encoded in *MSK*) corresponding to the 0000 symbol, in order to detect the preamble of a *802.15.4* frame (this preamble is composed of 4 null-bytes, i.e., eight 0000 symbols). The integrity check must be deactivated, because the *802.15.4* frames are not valid *BLE* frames (the chip must allow this deactivation so that a reception primitive can be implemented) and to configure the size of the frame to the maximum available size. At this stage, the *dewhitening* problem has to be solved: it must be ideally disabled, and if this is not possible, a *whitening* algorithm must be applied to

the frame in order to extract the output bits of the demodulator. The conversion to the original *Zigbee* symbols can be done very simply by using Hamming distance. Each received packet is split into 31-bits blocks and for each block, a Hamming distance is calculated in order to find which PN sequence encoded in *MSK* fits the best the received block. The use of the Hamming distance allows here to cope with two difficulties: bit errors caused by the approximation presented previously, but also interference due to the channel, that may generate bitflips during transmission.

Note that the equivalence of *O-QPSK* modulation with half sine pulse shaping and *MSK* modulation should in theory enable a "symmetric" pivoting attack, i.e, to also divert the use of *Zigbee* chips to attack the *BLE* protocol. However, this strategy is quite difficult to implement, because *Zigbee* protocol stack prevents us from finely controlling the *802.15.4* modulator input or demodulator output, mainly due to the *Direct Sequence Spread Spectrum* functionality, which performs the operation of transforming symbols into chip sequences. It would be necessary to be able to control the input of the modulator and the output of the demodulator, which does not seem to be easily achievable with existing devices.

V. Benchmarks

It is important to validate the *WazaBee* attack on chips from different manufacturers. We have chosen two different chips, *nRF52832* designed by *Nordic SemiConductors* and *CC1352-R1* designed by *Texas instruments*. Let us note that the attack does not depend on the chips we used, as it only exploits similarities between the physical layers used by the protocols themselves. Additionally, we are aware that the *TI CC1352-R1* chip natively supports 802.15.4-based protocols, however, of course, we only used its BLE capabilities during our experiments. In this section, we describe the proof of concept implementations, and present the experiments carried out to evaluate the quality of the *Zigbee* communications achieved with *WazaBee*.

The first implementation was carried out on the *nRF52832* chip, which chip offers great flexibility in the configuration of the embedded radio component *BLE 5.0*, and is compliant with the *LE 2M PHY* layer. Its radio *API* is similar to the *nRF51* one. This *nRF51 API* is well known to the security community for having been massively hijacked in recent years in order to develop offensive tools dedicated to *BLE* and *Enhanced ShockBurst* (*BTLEJack*, *radiobit*, ...). The prototype was implemented on a development board proposed by *AdaFruit* integrating this chip, the *Adafruit Feather nRF52 Bluefruit LE*. The second implementation was carried out on the *CC1352-R1* chip manufactured by *Texas Instruments*. The main motivation was to test the approach on a chip offering less configuration possibilities than the *nRF52* chip. The chip natively supports several protocols, including *BLE* and *802.15.4*. However, only the Bluetooth API was used for the implementation. This API being common to several chips from *Texas Instruments*, the implementation of the attack should be similar on other systems from the same manufacturer.

Two experiments were carried out in order to assess the reception and transmission primitives previously described. The first experiment, dealing with reception, consisted in transmitting one hundred *802.15.4* frames with a payload including a counter (incremented with each frame) using a *Zigbee* transmitter (AVR RZUSBStick Atmel). The development board implementing the *WazaBee* attack, spaced from the transmitter by a distance of 3 meters, received and decoded the corresponding frames, then calculated the FCS corresponding to the received frame to assess its integrity. For each *Zigbee* channel, the frames were classified into three categories: not received, received with integrity corruption, received without integrity corruption. The results are shown in table III.

It can be seen that the reception primitive of *WazaBee* has a very satisfactory reception rate for the two implementations on all channels, with an average of 98.625 % of the frames received without integrity corruption for *nRF52832* and 99.375 % for *CC1352-R1*. In both cases, there is a slight decrease in the reception rate for channels 17, 18, 21, 22 and 23, which can be explained by the interference with WiFi channels 6 and 11, used in our experimental environment. It can also be observed that the *CC1352-R1* presents a more stable reception than the *nRF52832*, without any integrity corruption of the received frames while the *nRF52832* missed 0.6875 % of the frames.

The transmission primitive was assessed under similar conditions: the development board implementing *WazaBee* was configured to transmit one hundred frames including a counter, and a *802.15.4* receiver (the RZUSBStick) was placed 3 meters away. Each transmitted frame could also be classified into three categories: not received, received with integrity corruption and received without integrity corruption. The experiment was performed on all *Zigbee* channels, and the corresponding results are shown in table III.

In both cases and for all channels, the rate of valid frames received without integrity corruption by the RZUSBStick is very satisfactory, with an average of 97.5% for *nRF52832* and 99.438 % for *CC1352-R1*. We observe a similar phenomenon to the one observed during the assessment of the reception primitive for channels 17 and 18, related to the simultaneous use of WiFi channel number 6 in our experimental environment. The rate of corrupted frames received is also slightly higher for *nRF82832* (with an average of 0.8125 % while the *CC1352-R1* did not miss any frame).

VI. Attack scenarios

In this section, we demonstrate the *WazaBee* attack by describing two attack scenarios we actually carried out. Two main attack scenarios, considering various devices, have been implemented. The first scenario illustrates the implementation of a subset of the *WazaBee* primitives on an unrooted Android phone, using an high level API. The second scenario presents the implementation of *WazaBee* on a commercial *BLE* tracker device in order to perform complex *Zigbee* attacks. We purposely chose these devices in order to illustrate the critical impact of the *WazaBee* attack. Indeed, Android phones and *BLE* trackers are very common devices, that anyone may

TABLE III: Reception and transmission primitives assessment results

Channels	Reception primitive				Transmission primitive			
	nRF52832		CC1352-R1		nRF52832		CC1352-R1	
	valid	corrupted	valid	corrupted	valid	corrupted	valid	corrupted
11	100	0	100	0	98	0	100	0
12	100	0	100	0	100	0	100	0
13	100	0	100	0	95	1	100	0
14	100	0	100	0	97	3	100	0
15	99	1	100	0	100	0	100	0
16	100	0	97	0	90	3	100	0
17	98	1	99	0	94	3	96	0
18	95	2	100	0	91	2	95	0
19	100	0	100	0	97	0	100	0
20	100	0	100	0	100	0	100	0
21	98	2	100	0	100	0	100	0
22	95	2	98	0	100	0	100	0
23	97	0	96	0	100	0	100	0
24	99	1	100	0	100	0	100	0
25	100	0	100	0	100	0	100	0
26	97	2	100	0	98	1	100	0

possess. The successful implementation of *WazaBee* on these devices shows that this attack may actually be deployed easily and massively.

Short videos of these two attack scenarios are respectively available at: https://youtu.be/a16LYLwvcZw and https://youtu.be/uDxD1jTNnoE.

A. Experimental setup

A main experimental setup is used for the two attack scenarios, based on a simple domotic *Zigbee* network with the *PANID 0x1234*. This network is composed of two *XBee* (a commercial implementation of *ZigBee*) transceivers. The first one (*16-bits address 0x0063*) is an end device simulating a sensor transmitting an integer (e.g. the temperature) every two seconds while the second one (*16-bits address 0x0042*) is a coordinator which acknowledges the data and displays it on a HTML graph. The channel 14, which matches the 2420 MHz frequency, is used.

B. Scenario A: injecting 802.15.4 frames using a smartphone

The first attack scenario was the injection of arbitrary *802.15.4* frames into our network, using an unrooted Android smartphone. For instance, an attacker could use a malware installed on an employee's phone to launch such an attack remotely, allowing him to perform multiple active attacks targeting *Zigbee* networks. It could also allow to exfiltrate discreetly sensitive data using a protocol that is not monitored.

As mentioned earlier, implementing the two primitives of the *WazaBee* attack requires the attacker to gain control over the lowest layers of the *BLE* protocol stack. However, the aim of the experiment is to test if an attacker that can only interact via an high level API could be able to implement at least a subset of the attack. As a consequence, this scenario was evaluated with the following contraints: 1) the smartphone is unrooted; 2) the attacker has only access to standard Android API with common permissions, and 3) the attack should be compliant with any *BLE* 5-compliant device, without the need to divert specific hardware components (e.g. InternalBlue [27]).

According to the specification, the received frames including a wrong CRC are dropped at the controller level and are not delivered to the host. Therefore, the received *802.15.4* frames are considered as invalid *BLE* frames and are filtered in the controller and not forwarded to the host. As a consequence, the implementation of the reception primitive is not possible without a low-level access allowing to collect invalid frames. The implementation of the transmission primitive is also tricky, because we only have an indirect control over the frequency and the payload content using a high level API. However, the extended advertising feature allows a partial implementation of the transmission primitive. Indeed, this feature has some interesting properties: it allows the transfer of large amount of data, it can use the 37 data channels without the need to initiate a *BLE* connection, it can use the *LE 2M* physical layer and it is based on predictable frame formats.

If the device uses *LE 1M* as primary physical layer and *LE 2M* as secondary physical layer, it initially transmits *ADV_EXT_IND* advertisements at 1 MBits/s on the primary advertising channels (37,38 and 39), indicating on which secondary advertising channel and the offset to the start time the extended advertisement will be transmitted. The channel selection is based on a pseudo-random algorithm named *Channel Selection Algorithm #2* [20], and is not directly controllable by the user. Then, the advertiser transmits the extended advertisement embedding the data provided by the user (*AUX_ADV_IND*) at 2 Mbits/s on the selected channel.

Diverting this feature in order to transmit *802.15.4* frames can be achieved using the strategy mentioned above to forge the advertising data. We first need to choose the PN sequences (encoded in *MSK*) corresponding to the frame to transmit. Then, we need to add some padding bytes before the frame (because of the multiple headers included before the data) and apply the dewhitening function to the resulting data. As this operation depends on the channel, it allows to select a specific *Zigbee* channel: in our case, we want to transmit data at 2420 MHz (Zigbee channel 14), which corresponds to *BLE* channel 8, so we perform the dewhitening operation using this *BLE* channel as input. The output is then cropped to remove the padding bytes, then the result is provided as advertising data. We use a manufacturer data field to encapsulate our forged

frame, resulting in a padding size of 16 bytes. Then, the extended advertising can be enabled using the smallest time interval in order to increase the probability that the channel selection algorithm picks our target channel.

We implemented this approach in an android application running on an unrooted OnePlus 6T smartphone, that fully supports the extended advertising feature. We were able to inject forged data packets to our target zigbee network, as illustrated in figure 4.

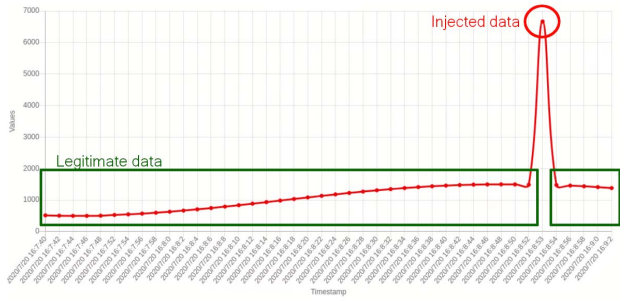

Fig. 4: Forged data packets injection from a OnePlus 6T smartphone

This approach is entirely compliant with the specification and only uses an high level API, meaning every *BLE* 5 device is able to inject *802.15.4* frames into at least eleven channels (especially those which have common frequencies with *BLE* data channels) in the 2.4-2.5GHz ISM band. As a result, it increases the attack surface of *802.15.4*-based protocols.

As we have chosen to implement the attack on a smartphone with limited permissions, it was not possible to implement the reception primitive. However, let us note that attackers with higher privileges may be able to gain a low level access and easily implement the two primitives. For example, *Internal-Blue* [27] allows to patch firmwares of *Broadcom* and *Cypress* controllers, which are common in off-the-shelf devices. If the attackers are able to reverse engineer the target firmware to identify the functions allowing to match the requirements mentioned in IV-D, they can easily write malicious patches and add custom code to the firmware implementing *WazaBee* primitives.

C. Scenario B: performing complex Zigbee attacks from a BLE tracker device

The second attack scenario illustrates the possibility to perform complex *Zigbee* attacks by abusing a *BLE* smart object. The impact of such an attack could be significant, as it may allow an attacker to build complex attacks involving legitimate *BLE* devices, that will not be identified as a potential threat to *802.15.4* networks. For example, an employee's mobile device (e.g. a smart watch, a tracker ...) could be infected outside the company in order to carry out a complex attack when the device is within range of the company's *Zigbee* network.

Our attack was performed on a commercial *BLE* tracker device, called Gablys Lite, which is based on a *nRF51822* chip. It requires a physical access to the device, as we used some unprotected debug pins providing a *Serial Wire Debug* (*SWD*) to flash a new firmware. Note that a similar attack could be performed using *BLE* vulnerabilities such as OTA updates abuse, which do not require this physical access.

The *nRF51822* chip is similar to *nRF52832*, but it doesn't support *LE 2M*, which is a key requirement of *WazaBee* attack. However, as the *Enhanced ShockBurst* protocol at 2 Mbits/s is supported by the chip, it can be used as an alternative for *LE 2M* physical layer. This solution has a direct impact on the reception quality, but it is sufficient to successfully conduct a complex active attack.

The main goal of this attack is to perform a denial of service targeting the sensor, in order to spoof it and inject fake data into the display interface. We used an existing attack targeting *XBee* nodes in order to perform a denial of service, allowing to inject a new configuration through remote AT commands [28]. The attack is divided into four main steps:

- **Active scanning:** the device transmits a *Beacon Request* on a channel and waits for a *Beacon* from the coordinator. If no *Beacon* is received before the timeout expires, the device selects the next channel. If a *Beacon* is received, the channel, the PanID and the coordinator's address are collected and saved,
- **Eavesdropping:** the device sniffs the legitimate frames in order to collect the sensor's address,
- **Remote AT command injection:** the device forges a remote AT command, using coordinator's address as source and sensor's address as destination. It allows to force the sensor to use another channel,
- **Fake data injection:** the device transmits fake data frames, mimicking the sensor's behaviour.

This attack was implemented successfully, as illustrated in figure 5. This experiment shows that WazaBee's primitives can be combined to conduct complex attack scenarios, and also that a legitimate commercial device can be modified and used to perform this kind of offensive strategies.

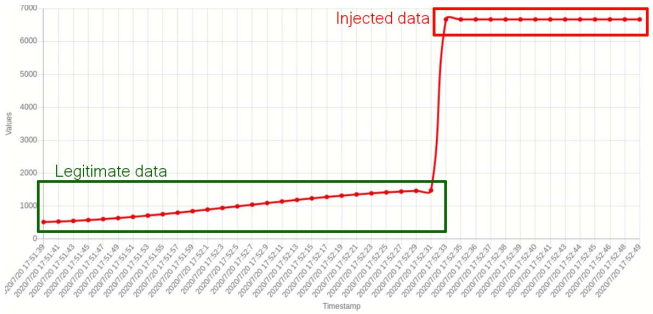

Fig. 5: Complex attack workflow from a BLE tracker

D. Conclusion

The two attack scenarios illustrated in the previous subsections are not exhaustive, but they illustrate the critical impact of *WazaBee* attack, and especially the considerable number of legitimate devices that may be diverted in order to

attack *802.15.4* networks. Depending on the corrupted device and the privilege level gained, an attacker may be able to implement the two primitives or only a subset of *WazaBee* attack. However, scenario A underlines the fact that even with a partial implementation, an attacker would be able to achieve interesting objectives, such as leaking sensitive data or disrupting legitimate nodes. These offensive strategies could also be combined in order to perform complex attack scenarios. Finally, these two scenarios also underline that the attack is easy to implement on various devices and may be used easily in the wild.

VII. Counter-measures

WazaBee attack is inherent to the wireless protocols and their modulations, even if some conditions need to be fulfilled to be implemented on some specific chips. As a consequence, we should consider every *BLE* 5 device as potentially vulnerable and the environments exposed to *BLE* devices should be designed and monitored with the assumption that some attacks could potentially be carried out through *802.15.4* networks. Several counter-measures could be investigated either to limit the impact of the attack, or to prevent or detect it.

As explained in section IV-D, the practical implementation of *WazaBee* requires controlling some features of the *BLE* chips. Making it difficult or impossible for an attacker to control these features (such as the deactivation of the CRC or the setting of a precise channel frequency), by chip manufacturers, would complicate the task of the attacker, and especially the implementation of the reception primitive. However, such counter-measures should only be considered as a first barrier for an attacker and not as efficient adequate solutions, as illustrated in our scenario A which only uses an high level API in order to implement the transmission primitive.

Some other common counter-measures, such as cryptographic techniques, that most of the *802.15.4*-based protocols provide, should be systematically used. If these techniques are implemented, even if the *WazaBee* attack is still possible, the task of the attacker would be much more complicated. Unfortunately, the correct implementation of these counter-measures is not trivial and it highly depends on the protection of the keys. Note that some known attacks [29] aiming at breaking the *802.15.4* encryption can be performed using *WazaBee* and also that the attacker can still perform denial of service attacks [30].

Finally, some defensive solutions dedicated to the IoT context, to monitor and detect in real time attacks targeting wireless protocols, can also be considered. Indeed, the existence of such offensive strategies motivates the deployment of intrusion and prevention detection systems based on the analysis of radio communications. Such systems could simultaneously monitor multiple wireless protocols (even those which are not deployed in the legitimate environment) such as the solution proposed in [31], or could be protocol agnostic, such as the intrusion detection approaches proposed in [32], [33]. These intrusion detection systems are designed to monitor the physical layers of communication protocols (by monitoring signal strength on different frequency bands) and are based on the modeling of legitimate communications and therefore detect accidental faults (in [33]) or malicious activities (in [32]) by identifying deviations from legitimate behavior.

More generally, the wireless attacks investigated in this paper may impact other protocols, depending on the compatibility between their modulations and channel coding, along with the programmability of the underlying hardware. Indeed, if the frequencies overlap, while the modulations are similar enough to be able to control what is received by one protocol from an emission of the other, the two protocols are by design vulnerable to pivoting techniques. Let us note that evaluating accurately the similarities between two modulations is an open challenge. We argue that such attacks should be included in the threat assumptions when different wireless technologies are deployed in connected environments. We also argue that protocol designers should consider such possibilities of cross-protocol interactions when creating new wireless standards, to reduce the risks of pivoting attacks using their protocol as basis.

VIII. Conclusion and prospective work

In this paper, we have presented a new pivoting attack strategy, called *WazaBee*, allowing the legitimate operation of a chip intended to communicate via *BLE* to be diverted in order to send and receive *Zigbee* communications (actually, our approach is compliant with all *802.15.4* frames). A proof of concept is also presented by implementing the attack on two chips with different architectures and from different manufacturers. The performance results measured during the two evaluations carried out for the transmission and reception primitives were very stable and reliable. The direct consequence is a considerable increase in the attack surface in IoT environment, each system communicating via a protocol based on the *802.15.4* standard (*Zigbee*, *6LoWPan* ...) being potentially accessible from a component supporting *BLE*, a particularly widespread technology in *IoT* environments. The impact of *WazaBee* has been illustrated in two attack scenarios, highlighting the variety of uses an attacker could make of it.

The study of this type of attacks has been seldom explored and it is very important to develop suitable countermeasures. We plan to focus our future work on formalizing this type of attack and exploring its feasibility with other wireless communication protocols. In this way, we plan to further investigate the similarities between different existing modulation techniques that could be exploited to perform *WazaBee* like attacks. Defining a metric to measure such similarities could be useful to anticipate, for example, which protocols based on amplitude modulation could be diverted to other protocols based on frequency modulation. From a more defensive perspective, the existence of this type of attack argues in favor of protection solutions capable of monitoring multiple protocols in real time, and detecting potential illegitimate behaviours. Consequently, we plan to study the design and implementation of a multi-protocol and multi-layer intrusion detection and prevention system that can cope with such attacks.

REFERENCES

[1] T. Zillner and S. Strobl, "Zigbee Exploited : The Good, the Bad and the Ugly," https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf, 2015.

[2] M. Ryan, "How Smart is Bluetooth Smart ?" 2013.

[3] ——, "Bluetooth: With Low Energy comes Low Security," 2013.

[4] E. Ronen, A. Shamir, A. Weingarten, and C. O'Flynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 195–212.

[5] Y. Bachy, F. Basse, V. Nicomette, E. Alata, M. Kaâniche, J. Courrège, and P. Lukjanenko, "Smart-tv security analysis: Practical experiments," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2015, pp. 497–504.

[6] "B-L475E-IOT01A Data Brief," ST Microelectronics, 2018, https://www.st.com/resource/en/data_brief/b-l475e-iot01a.pdf.

[7] "CC2652R Data Sheet," Texas Instruments, 2019, http://www.ti.com/lit/ds/symlink/cc2652r.pdf.

[8] K. Chebrolu and A. Dhekne, "Esense: Communication through energy sensing," in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 85–96. [Online]. Available: https://doi.org/10.1145/1614320.1614330

[9] S. M. Kim and T. He, "Freebee: Cross-technology communication via free side-channel," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 317–330. [Online]. Available: https://doi.org/10.1145/2789168.2790098

[10] Z. Li and T. He, "Webee: Physical-layer cross-technology communication via emulation," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 2–14. [Online]. Available: https://doi.org/10.1145/3117811.3117816

[11] W. Jiang, Z. Yin, R. Liu, Z. Li, S. M. Kim, and T. He, "Bluebee: A 10,000x faster cross-technology communication via phy emulation," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3131672.3131678

[12] W. Jiang, S. M. Kim, Z. Li, and T. He, "Achieving receiver-side cross-technology communication with cross-decoding," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 639–652. [Online]. Available: https://doi.org/10.1145/3241539.3241547

[13] T. Goodspeed, S. Bratus, R. Melgares, R. Shapiro, and R. Speers, "Packets in packets: Orson welles' in-band signaling attacks for modern radios," 08 2011, pp. 7–7.

[14] M. C. Millian and V. Yadav, "Packet-in-packet exploits on 802 . 15 . 4," 2015.

[15] T. Goodspeed, "Promiscuity is the nrf24l01+'s duty," http://travisgoodspeed.blogspot.com/2011/02/promiscuity-is-nrf24l01s-duty.html, 2011.

[16] M. Newlin, "MouseJack : White Paper," 2016, https://github.com/BastilleResearch/mousejack/blob/master/doc/pdf/DEFCON-24-Marc-Newlin-MouseJack-Injecting-Keystrokes-Into-Wireless-Mice.whitepaper.pdf.

[17] D. Cauquil, "Sniffing btle with the micro:bit," *PoC or GTFO*, vol. 17, pp. 13–20, 2017.

[18] ——, "Weaponizing the BBC Micro:Bit," 2017, https://media.defcon.org/DEFCON25/DEFCON25presentations/DEFCON25-Damien-Cauquil-Weaponizing-the-BBC-MicroBit-UPDATED.pdf.

[19] ——, "Radiobit, a BBC Micro:Bit RF firmware," 2017, https://github.com/virtualabs/radiobit.

[20] *Bluetooth Core Specification*, Bluetooth SIG, 12 2019.

[21] "Ieee standard for low-rate wireless networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, April 2016.

[22] M. Bettayeb, Q. Nasir, and M. A. Talib, "Firmware update attacks and security for iot devices: Survey," in *Proceedings of the ArabWIC 6th Annual International Conference Research Track*, ser. ArabWIC 2019. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3333165.3333169

[23] Armis, "Blueborne Technical White Paper," https://go.armis.com/hubfs/BlueBorne%20Technical%20White%20Paper.pdf, 2017.

[24] ——, "BleedingBit Technical White Paper," https://go.armis.com/hubfs/BLEEDINGBIT%20-%20Technical%20White%20Paper.pdf, 2018.

[25] G. Vishwakarma and W. Lee, "Exploiting jtag and its mitigation in iot: A survey," *Future Internet*, vol. 10, p. 121, 12 2018.

[26] S. Pasupathy, "Minimum shift keying: A spectrally efficient modulation," *IEEE Communications Magazine*, vol. 17, no. 4, pp. 14–22, July 1979.

[27] D. Mantz, J. Classen, M. Schulz, and M. Hollick, "Internalblue - bluetooth binary patching and experimentation framework," *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, Jun 2019. [Online]. Available: http://dx.doi.org/10.1145/3307334.3326089

[28] I. Vaccari, E. Cambiaso, and M. Aiello, "Remotely exploiting at command attacks on zigbee networks," *Security and Communication Networks*, vol. 2017, pp. 1–9, 10 2017.

[29] N. Vidgren, K. Haataja, J. L. Patiño-Andres, J. J. Ramírez-Sanchis, and P. Toivanen, "Security threats in zigbee-enabled systems: Vulnerability evaluation, practical experiments, countermeasures, and lessons learned," in *2013 46th Hawaii International Conference on System Sciences*, 2013, pp. 5132–5138.

[30] X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen, "Ghost-in-zigbee: Energy depletion attack on zigbee-based wireless networks," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 816–829, 2016.

[31] S. Siby, R. R. Maiti, and N. Tippenhauer, "Iotscanner: Detecting and classifying privacy threats in iot neighborhoods," 2017.

[32] J. Roux, E. Alata, G. Auriol, M. Kaâniche, V. Nicomette, and R. Cayre, "RadIoT: Radio Communications Intrusion Detection for IoT - A Protocol Independent Approach," in *17th IEEE International Symposium on Network Computing and Applications (NCA 2018)*, Cambridge, Massachusetts, United States, Nov. 2018, p. 8p. [Online]. Available: https://hal.laas.fr/hal-01914981

[33] S. Rajendran, W. Meert, V. Lenders, and S. Pollin, "Unsupervised wireless spectrum anomaly detection with interpretable features," *IEEE Transactions on Cognitive Communications and Networking*, vol. PP, pp. 1–1, 04 2019.