New Replay Attacks on ZigBee Devices for Internet-of-Things (IoT) Applications

Mohammad Shafeul Wara
Dept. of Electrical and Computer Engineering
University of New Hampshire
Durham, NH 03824, USA
mw1260@wildcats.unh.edu

Qiaoyan Yu
Dept. of Electrical and Computer Engineering
University of New Hampshire
Durham, NH 03824, USA
qiaoyan.yu@unh.edu

Abstract—**The ZigBee wireless technologies have been widely applied in Internet of Things (IoT) applications. Among all identified security threats, replay attack is one of the major challenges. Although the firmware and authentication algorithms for ZigBee devices are being updated over the time, the security measures for ZigBee enabled IoT devices cannot efficiently thwart all replay attacks. As new sniffing devices are available on the market, new replay attacks will further challenge the integrity of the communication between ZigBee devices. This work proposes a new replay attack to reveal the security vulnerabilities of the existing commercial ZigBee devices, Phillips Hue bulbs and Xbee S1 and S2C modules. Two case studies performed in this work confirm that the proposed replay attack can successfully transmit the captured packets to the victim device in spite of the built-in countermeasures.**

Index Terms—**ZigBee security, replay attack, Xbee, Philips Hue, IoT, IEEE 802.15.4, KillerBee, wireless security.**

I. Introduction

ZigBee/IEEE 802.15.4 is one of the widely used protocols in Internet-of-Things (IoT) applications. Although ZigBee provides some end-to-end security via the AES-CCM algorithm and 128-bit symmetric encryption [1], [5], [8], the ZigBee protocol is still vulnerable to many threats, such as ZigBee network key sniffing, ZigBee End Device (ZED) sabotage, and Denial-of-Service (DoS), and replay attacks [1], [6], [8].

The key sniffing attack in a ZigBee network result in key leakage during the process of network commissioning. When a new node without a preinstalled network key tries to connect to a network, an unencrypted network key will be shared with the device in the network. An attacker could sniff the unencrypted transaction and recover the network key with the help of some special tools (e.g., API-Mote, KillerBee and Wireshark). The ZED sabotage attack aims for expeditiously draining the battery embedded in IoT ZED devices. An attacker may impersonate a legitimate device to continuously send signals to the victim device, stopping the device from transitioning into sleep mode [8]. Consequently, the ceaseless communication causes the battery to drain completely within a short period of time. The DoS attack prohibits its target device from accepting new incoming packets. An attacker can succeed in the DoS attack either by setting the device frame counter to a higher than normal value (so that the incoming packets with lower frame counter will be discarded during authentication) or by maliciously transmitting frames over the ZigBee network to disrupt the communication.

This work is partially supported by the NSF grant #1652474.

Due to the network discovery and identification threats [7], replay attacks still challenge the ZigBee security. The high security standard and 128 bit AES encryption are not sufficient to stop the sniffing devices from eavesdropping the ZigBee network traffic. In spite of many firmware updates on ZigBee devices, ZigBee data authentication could be nullified by newly emerged attacks. There are no robust and sustainable solutions available yet to thoroughly protect the ZigBee communication. To facilitate effective countermeasure designs, this work contributes to discover new replay attacks that could be performed on ZigBee devices.

The rest of this work is organized as follows. In Section II, the related work is summarized and our main contributions are highlighted. The preliminaries for the replay attack and experimental devices/tools are introduced in Section III. We propose a new reply attack flow in Section IV. A case study is provided in Section V to introduce the detailed attack implementation and our key observation. We conclude this work in Section VI.

II. Related Work and Our Contributions

A. Related Work

Olawumi et al. [7] introduced three attacks on ZigBee network. The first attack can detect the availability of ZigBee devices and identify their identities in the network. The second attack is capable of sniffing the data transmitted by a particular device. The third attack leverages Atmel RZ Raven USB stick and KillerBee software tool to capture ZigBee packets. To thwart the first attack, Olawumi et al. suggested integrating an intrusion detection and prevention technique to regularly monitor the beacon frame process in a ZigBee network. To protect the system from the second attack, the authors proposed to replace the standard security level with the high security level in safety-critical ZigBee-enabled systems. They further advocated to apply time stamps in ZigBee encryption process so that the replayed frames will be discarded upon reception. The work [8] proposed an attack to sabotage ZigBee end devices by prematurely draining the battery of a ZigBee device. A possible countermeasure presented in the work [8] is using a 32-bit frame counter embedded in each frames to thwart replay attacks. However, no validation was performed in the work [8].

In the work [2], Fan et al. performed a replay attack on a ZigBee device (undisclosed brand). The ON/OFF command packets were captured and then replayed with Atmel RZUSB stick. Their replay attack was not successful because the counter associated with each replayed frame was less than the counter value of the receiver device. Consequently, the replayed packets were rejected by the receiver. The work [2] hypothesized that if an attacker could overwrite the packet counter a larger number, the replay attack can succeed theoretically.

Gupta demonstrated a replay attack on Philips Hue bulb in the handbook [3]. While a legitimate user controlled the bulb's ON/OFF and color, the packets were captured through a sniffing device. Next, the dump file was replayed by using a RZUSB stick and Attify ZigBee framework, which is a KillerBee based graphical user interface (GUI) toolkit. With that approach, the Philips bulb successfully reacted to the replayed packets. The author concluded that the replayed packets were accepted because the cyclic redundancy check (CRC) based verification mechanism was not implemented in the Philips Hue system. If the bulb uses frame counters and message integrity code (MIC) to authenticate incoming packets to enhance the resilience against the replay attack, a straightforward replay attack will fail.

Towards the interception of packets attack, the work in [7] suggests using high security standard to allow ZigBee devices to share the encryption key. That proposal requires ZigBee device manufactures to pre-install a master key. Unfortunately, as reported in [11], the master key was leaked via twitter. The leaked master key will jeopardise the safety of all the ZigBee devices. Moreover, an encrypted network alone cannot thwart replay attack. Although a concept of time stamping mechanism is proposed in [7], the authors do not validate that method.

B. *Our Contributions*

The main contributions of this work are as follows:

- We evaluate the ZigBee/IEEE 802.15.4 protection mechanisms against replay attacks and discover a weakness that allows an attacker to turn ON/OFF Hue smart bulbs.
- We propose a new relay attack on Philips Hue smart bulb that provides an adversary with a partial control. Our noise removal technique is capable of identifying and extracting noise from the captured packets of Xbee communication.
- We examine the countermeasure proposed in [2] and prove that editing the frame counter to a higher value is not a practical method against message integrity check. Our work also confirms that the method of increasing the packet counter for incoming packet authentication [8], [10] cannot thwart our replay attack.

III. PRELIMINARIES

The company Digi International [4] has developed ZigBee-enabled Xbee modules, Xbee *S1* and *S2C*, to execute the operations defined in the IEEE 802.15.4 and ZigBee protocols. Xbees are small radio devices and they act as transceivers. Additional devices/microcontrollers/computers are required to configure and monitor Xbees when we deploy them in an application. Digi International provides users with software to choose and configure the firmware for Xbees.

A. *Security Levels in Xbee*

Through the graphical user interface *XCTU*, users can enable or disable ZigBee security features such as 128-bit AES encryption and select their own security key. Xbee has three ZigBee security level: residential, standard and high security. Residential security was initially supported in ZigBee 2006 standard and required a network key to be shared between the connected devices. By using a link key (APS layer), standard security provides more options for the security improvement over the previous one. High security offers a wide range of security features including message integrity, confidentiality, and authentication [4]. Typically, Xbee devices are implemented at standard security level.

An Xbee module may include message integrity code (MIC), but Xbee/Xbee Pro S2C model does not support MIC-based message integrity check [4]. Instead, Xbee modules maintain the data integrity by adding one byte checksum at the end of each frame. All bytes of previously received frame except the start delimiter (first byte of the frame) and the length (the next two bytes) are utilized to calculate the checksum. Any frame with a wrong checksum sent to an Xbee will not be processed further and that frame will be discarded by the receiver [4].

B. *Attack Tools*

The framework KillerBee [9] is designed to simplify the process of sniffing packets from the air interface or dump packets in the format of libpcap (packet capture library), which is readable in Wireshark. We can also use KillerBee to inject arbitrary packets [7]. The current KillerBee framework supports multiple hardware devices for threats investigation, including the River Loop Api-Mote, Atmel RZ RAVEN USB Stick, MoteIV Tmote Sky, TelosB mote, Sewino Sniffer, and other hardware running Silicon Labs Node Test firmware [7].

IV. PROPOSED REPLAY ATTACK FLOW

Replay attacks are easy to implement but could cause a severe consequence, especially in security-critical applications such as hospitals, power plants, and smart homes [7]. A ZigBee device employs 128-bit AES encryption to protect its data integrity; however it cannot completely thwart replay attacks. This is because replay attacks do not require the knowledge of the crypto key. Once the target packet is captured, that packet could be re-transmitted over the air. Due to the low technical challenge, attackers prefer using replay attacks to sabotage ZigBee enabled systems. Thus, it is imperative to expand the

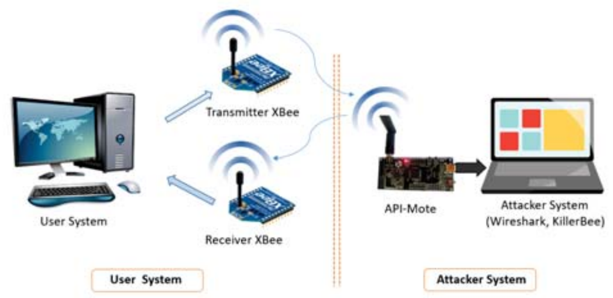
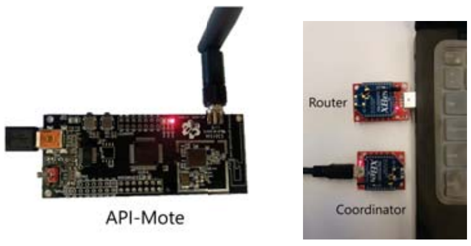


Fig. 1: Our experimental setup.



(a) (b)

Fig. 2: Devices used in proposed attack. (a)Sniffing device (API-Mote), and (b) two Xbee devices connected to a user system.

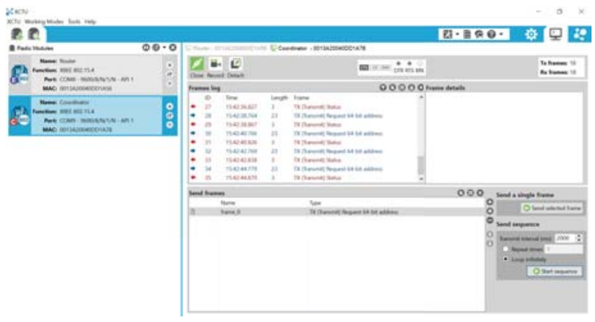


Fig. 3: Screenshot of XCTU.

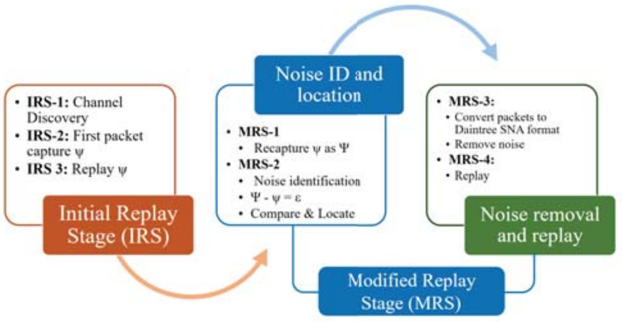


Fig. 4: Overview of proposed replay attack flow.

knowledge of how new replay attacks can be formed to harm ZigBee applications. In this work, we present a new replay attack that can be implemented with low-cost hardware and open-source software tools.

A. *Hardware and Software Setup for Proposed Attack*

To perform a sniffing attack [1], [7] on a ZigBee network, two Xbee S1 and S2C devices are connected to each other. One Xbee serves as a coordinator and another one operates as a router. Our attack system is depicted in Fig. 1. In the attacker system, an API-Mote V4 Beta is connected to a Ubuntu Linux OS based computer system. The API-Mote shown in Fig. 2 is the hardware to support the exploration of ZigBee security vulnerabilities.

The software XCTU installed in a Windows laptop is used to configure the Xbee devices, generate and transmit packets to the router Xbee, and visualize the transmitted and received messages in real time. Figure 3 shows the XCTU interface for message monitoring. In the upper-left corner of the interface, we can find the Xbee devices, their roles in the communication, and their MAC addresses. In the middle of the interface, we can track the frames transferred between the router and coordinator. More details for each frame is available in the middle-bottom screen. We can also configure the transmission frequency for each selected frame.

B. *Overview of Proposed Attack Flow*

Our attack process is split into two stages **Initial Replay Stage (IRS)** and **Modified Replay Stage (MRS)**. IRS is a general replay approach and it consists of three major steps: (IRS-1) *channel discovery*, (IRS-2) *first packet capture* and (IRS-3) *first packet replay*. In the phase of MRS, our attack will be performed according to the following steps: (MRS-1) *second packet capture*, (MRS-2) *noise identification*, (MRS-3) *noise removal* and (MRS-4) *final replay*. The overview of proposed attack flow is depicted in Fig. 4.

The main facilities for our replay attack include two API-Mote devices (i.e., API-Mote-1 and API-Mote-2) and five KillerBee tools (i.e., *zbid, zbstumbler, zbdump, zbconert* and *zbreplay*). The API-Mote devices enable the communication between KillerBee and ZigBee devices. Inside each API-Mote, a Future Technology Devices International (FTDI) chip is used to establish the connection to a host system. The core function for each KillerBee tool is listed in Table I.

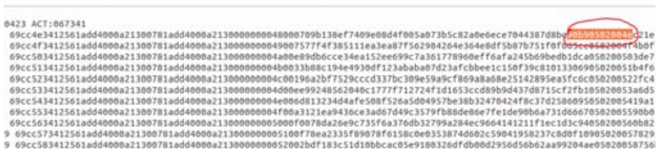
C. *Details for Each Step in Proposed Attack*

In this subsection, we introduce the main operations that will be performed in each step. The three steps in the first stage IRS are straightforward. The novelty of our proposed attack is in the second stage MRS. During MRS, we conduct noise detection and removal operations so that the replayed message will be as same as the original one.

IRS-1: To identify the communication channel of a victim device and its personal area networks (PAN) identification, the

TABLE I: Core Functions of KillerBee Tools

Tool Name	Description of function
zbstumbler	Discover communication channel
zbdump	Capture packets
zbreplay	Replay packets
zbconvert	Packet format conversion



```
0423 ACT:067341
 69cc4e3412561add4000a21300781add4000a21300000000048000709b138ef7409e08d4f085a073b5c82a0e6ece7044387d8bc0b9050200040c21e
 69cc4f3412561add4000a21300781add4000a21300000000049007577f4f385111ea3ea87f562984264e364e8df5b87b751f0f005ce0502000f4b0f
 69cc503412561add4000a21300781add4000a2130000000004a00e89db6cce34ea152ee699c7a3617778960eff6afa245b69bedb1dca0502000503de7
 69cc513412561add4000a21300781add4000a2130000000004b0033b88c194e4930df123ababa07d23afcbbee1c150f39c810133069050200051b4f6
 69cc523412561add4000a21300781add4000a2130000000004c00196a2bf7529cccd337bc309e59a9cf869a8a68e25142895ea5fc6c050200522fc4
 69cc533412561add4000a21300781add4000a2130000000004d00ee99248562040c1777f712724f1d1653ccd89b9d437d8715cf2fb105020053a6d5
 69cc543412561add4000a21300781add4000a2130000000004e006d813234d4afe508f526a5d04957be38b32470424f8c37d258609505020005419a1
 69cc553412561add4000a21300781add4000a2130000000004f00a3121ea9436ce3ad67d49c3579fb88de86e7fe1de90b6a731d66670502005590b0
 69cc563412561add4000a21300781add4000a21300000000050000f0078da26e9c735f6a376db32799a284ec9664141211f1ec1d3c94050200560b82
9 69cc573412561add4000a21300781add4000a2130000000005100f78ea2335f89078f6158c0e0353874d602c59041958237c8d0f10905020057829
9 69cc583412561add4000a21300781add4000a2130000000005202bdf183c51d10bbcac05e9180326dfdb00d2956d56b62aa99204ae050200587560
```

Fig. 5: Converted Daintree SNA packet. Note, the highlighted area represents for the 6-byte replay attack noise that needs to be removed before replay.

first step of our proposed attack begins with sending beacon requests to all available ZigBee channels (i.e., channel 11 to channel 26). The zbstumbler tool in the KillerBee software is leveraged to acquire the ZigBee device's PAN identification, which will be utilized in the later steps.

IRS-2: When the target device is communicating with other ZigBee devices, the communication traffic is captured by the zbdump tool and API-Mote-1. We save the traffic as the first packet capture and call it Dump-1 (denoted as ψ) amid our attack.

IRS-3: In this step, we simply re-transmit the captured packet ψ to the target device. However, the replayed message would differ from the message received in the course of a legitimate transmission. This is because some noise will be added to the first capture by the API-Mote-1.

MRS-1: We use API-Mote-1 to replay the packet ψ dumped from IRS-2 and then re-capture it through API-Mote-2. To differentiate the newly captured packet from ψ, we name the replayed and re-captured (Dump-2) packet as Ψ. The replay noise ϵ is defined as the difference between ψ and Ψ. Equation (1) expresses the relation between the first and second captured packets.

$$\Psi = \psi + \epsilon \tag{1}$$

MRS-2: In this step, we compare two packet dumps ψ and Ψ to locate the replay noise ϵ. We further zoom in ϵ to analyze the noise associated with the first packet capture ψ. This reply noise should be removed in order to defeat the message integrity check that may be implemented in the advanced ZigBee device. After noise removal, ψ becomes ψ*.

MRS-3: To remove the replay noise ϵ, we take one extra step to convert the captured packet from the libpcap format to the Daintree SNA format. The former format is not easy to edit; in contrast, the latter one can be modified in a text editor. After text editing, we can retrieve the message close to the original one. Figure 5 shows the identified noise.

MRS-4: In the last step, we replay the modified packet ψ* again. Because of our noise removal technique, the victim device will be deceive by the replayed packet as a legitimate message. Thus, our proposed replay attack will succeed even though the victim ZigBee device may have the message authentication mechanism as a countermeasure against replay attacks.

V. A Case Study

A. Experimental Setup

We evaluated the threats of replay attack on Xbee devices (S1 and S2C) and Philips Hue light bulbs that employ IEEE 802.15.4/ZigBee protocol for communication. The communication was encrypted with AES-CCM 128-bit encryption algorithm. The API-Mote together with the KillerBee tool set and the Wireshark packet analyser were used to sniff data. A frame containing a message "ZigBee Encrypted CtoR" was generated by the XCTU frame generator and sent through the coordinator Xbee to the router Xbee at every two seconds.

B. Implementation of Proposed Replay Attack

1) Limitation on Current Countermeasure Used in Philips Hue: The Philips Hue verifies MIC as a part of defense mechanism to thwart certain replay attacks. If the packet's MIC does not match with bulb's MIC or the packet counter is tempered, the packet (likely replayed packet) will be rejected. However, there is a loop hole that debilitates this countermeasure. When the bulb turns ON after a physical shut-down (either by a manual switch or removing the bulb from the socket), the bulb can be turned ON or OFF by a replayed packet. In fact, any ON/OFF packet can be replayed as many times as the attacker wants if the bulb can be disconnected physically and then reconnected back to the network. The replay attack above will succeed regardless of whether the bulb is being controlled by a legitimate command or not.

2) Our Attack Implementation on Xbee: The general attack flow chart proposed in Section IV-B needs some modifications to adapt to different ZigBee devices. First, we conducted the proposed replay attack on Xbee devices to validate the feasibility of our attack and noise removal technique. The process of proposed replay attack is shown in Fig. 6.

Dumped packets: Each dumped packet is composed of data payload, MIC, Frame Check Sequence (FCS), destination mode and address, and PAN ID. Depending on the capture moment, the size of a dumped packet varies. Table II compares the packets dumped from three rounds. The first dump was captured while the original message was being sent from one Xbee to another. The second and third dumps were captured during the first and second dumps were replayed, respectively. As shown in Table II, MIC and FCS become part of the frame payload. The frame size increases with the dump timing. For instance, the packet captured by the third dump contains 35 bytes; in contrast, the packet at the first dump only has 23 bytes. Note, only data and MIC are mentioned in the table and other common information is excluded.

Comparison of packet size: In our experiment, the replayed packets were successfully received by the Xbee S1 and S2C. However, there were 6-byte data (four bytes for MIC and two bytes for FCS) appended to the packet captured

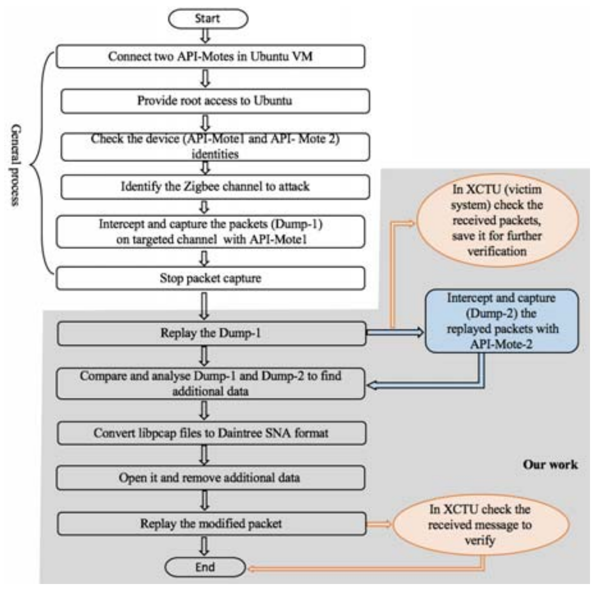
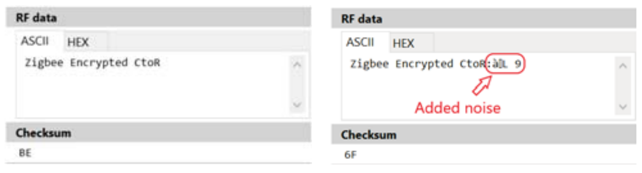

Start
Connect two API-Motes in Ubuntu VM
Provide root access to Ubuntu
Check the device (API-Mote1 and API- Mote 2) identities
Identify the Zigbee channel to attack
Intercept and capture the packets (Dump-1) on targeted channel with API-Mote1
Stop packet capture
General process
Replay the Dump-1
Intercept and capture (Dump-2) the replayed packets with API-Mote-2
In XCTU (victim system) check the received packets, save it for further verification
Compare and analyse Dump-1 and Dump-2 to find additional data
Convert libpcap files to Daintree SNA format
Open it and remove additional data
Replay the modified packet
In XCTU check the received message to verify
Our work
End

Fig. 6: Flow chart for proposed replay attack on Xbee.

TABLE II: Comparison of packets from different dumps.

Dump ID	1^{st} Dump	2^{nd} Dump	3^{rd} Dump
Data (hex)	07 09 b1 38 ef 74 09 e0 8d 4f 00 5a 07 3b 5c 82 a0 e6 ec e7 04 43 87	07 09 b1 38 ef 74 09 e0 8d 4f 00 5a 07 3b 5c 82 a0 e6 ec e7 04 43 87 d8 bc 90 b9 05 02	07 09 b1 38 ef 74 09 e0 8d 4f 00 5a 07 3b 5c 82 a0 e6 ec e7 04 43 87 d8 bc 90 b9 05 02 00 4e 95 70 05 02
MIC	d8 bc 90 b9 05 02 00 4e	00 4e 95 70 05 02 00 4e	00 4e 95 70 05 02 00 4e
Length (bytes)	23	29	35

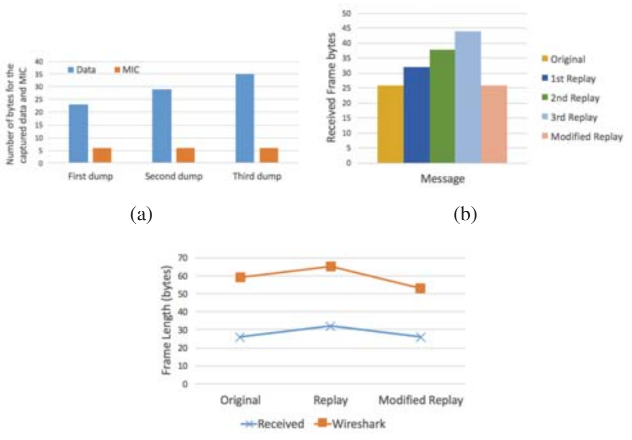

RF data
ASCII HEX
Zigbee Encrypted CtoR
Checksum
BE

(a)

RF data
ASCII HEX
Zigbee Encrypted CtoR:à□L 9
Added noise
Checksum
6F

(b)

Fig. 7: Replayed packets that are received by the Xbee devices. (a) Without and (b) with noise at the target device.



(a) (b)

(c)

Fig. 8: Impact of Xbee replay attacks and proposed noise removal technique on the packet size. (a) The number of bytes in captured payload and MIC at different dumping rounds, (b) the size of received message in XCTU with different versions of replay, and (c) the packet size measured by Wireshark and XCTU.

previously. Despite of the increased packet size, the Xbee receiver still accepted the replayed packet. For instance, an Xbee enabled light bulb needs a command "Turn On" to activate the bulb. If the packet that the light bulb receives is a replay one, the command will be "Turn OnXyz", in which the additional "Xyz" is the replay noise. In most cases, the replay noise is represented by some non-readable characters. Figures 7(a) and (b) show the the message "ZigBee Encrypted CtoR" without and with the replay noise.

The comparison of the packet size dumped at different rounds is presented in Fig. 8(a). The size of MIC remains the same. Although the packet size increases due to the replay noise induced in the process of packet dumping, our noise removal technique can eliminate the noise. We first converted the captured packet into a format of Daintree SNA (Sensor Network Analyzer), which allowed us to edit in a text editor. Next, the six bytes of MIC in the first dump were identified as the noise. We erased the MIC from the dumped packet and then re-transmitted it to the victim Xbee router. Thus, the router received the replay packet as if it was a clean original message. As shown in Fig. 8(b), our noise removal technique successfully made the size of the received replayed packet return to the original length. Another interesting fact is that the frame size captured by Wireshark is larger than what was observed in XCTU. This is because XCTU does not count the extended source and destination addresses, frame counter and sequence number. As illustrated in Fig. 8(c), the modified and original packets consist the same number of bytes (26). On the contrary, the replay packet has a few more bytes. Similarly, Wireshark can detect more bytes in the replayed packet than what was practically received.

3) Our Attack Implementation on Philips Hue: We repeat our replay attack flow on a Philips Hue bulb with the flow shown in Fig. 9. First, the bulb is connected to a ZigBee network created by a bridge. That bridge acts as a gateway between the Hue mobile application (user) and the bulb. A user can send a command to turn ON and OFF the bulb. Once the bridge receives the command from the user, it transmits the packets to the destination bulb. We (attacker) maliciously capture the ON/OFF packets transmitted by the bridge with the intention to take control of the bulb illegitimately. Next, we identify and extract those ON/OF packets and replay them to the target device. Although the bulb does not react to the ambience and intensity control, it responds to the ON/OFF packets once per each power-up period.

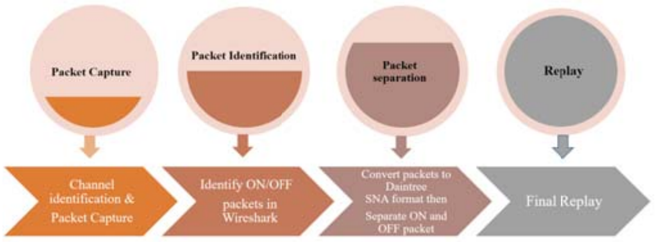
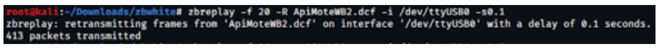
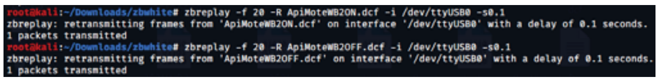

Packet Capture
Packet Identification
Packet separation
Replay
Channel identification & Packet Capture
Identify ON/OFF packets in Wireshark
Convert packets to Daintree SNA format then Separate ON and OFF packet
Final Replay

Fig. 9: Flow chart for proposed replay attack on Philips Hue.



```
root@kali:~/Downloads/zbwhite# zbreplay -f 20 -R ApiMoteWB2.dcf -i /dev/ttyUSB0 -s0.1
zbreplay: retransmitting frames from 'ApiMoteWB2.dcf' on interface '/dev/ttyUSB0' with a delay of 0.1 seconds.
413 packets transmitted
```

(a)



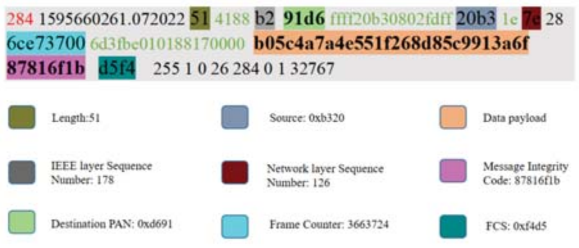
```
root@kali:~/Downloads/zbwhite# zbreplay -f 20 -R ApiMoteWB2ON.dcf -i /dev/ttyUSB0 -s0.1
zbreplay: retransmitting frames from 'ApiMoteWB2ON.dcf' on interface '/dev/ttyUSB0' with a delay of 0.1 seconds.
1 packets transmitted
root@kali:~/Downloads/zbwhite# zbreplay -f 20 -R ApiMoteWB2OFF.dcf -i /dev/ttyUSB0 -s0.1
zbreplay: retransmitting frames from 'ApiMoteWB2OFF.dcf' on interface '/dev/ttyUSB0' with a delay of 0.1 seconds.
1 packets transmitted
```

(b)

Fig. 10: Snapshot of replayed packets on Philips Hue bulb (a) Unseparated replay has more number of packets, while (b) separated replay can have as low as one packet.

Different than those in the scenario of Xbee, the captured packets in Philips Hue do not have noise. Hence, we do not need to remove any noise content from the packet. However, the captured packets contain other network traffic. In order to make the attack more efficient, the ON/OFF packets should be separated. As shown in Fig. 10(a), if we replay an unseparated packet, 413 packets will be transmitted. The total number of packets for replay depends on the nature and type of the captured packets. In contrast, once we filter out the ON/OFF packets from the entire traffic, the number of packets for replay attack transmission is reduced to 1 as shown in Fig. 10(b). As a result, our attack can efficiently take over the control of the target Philips Hue bulb.

Next, we examine the effectiveness of the attack methods proposed in literature [2], [10] on Philips Hue light bulbs. We assume that the replay attack aims for the known packet turning off the light bulb. The work [10] envisions that increasing the frame counter for that packet will make the replay attack succeed. To validate their method, we implemented the countermeasure and used Wireshark to capture the frame turning OFF the light bulb. The top half of Fig. 11 shows the capture frame and the bottom half of the same figure summarizes the different fields. As can be seen, the frame counter contains a value of 3663724. We first increased the frame counter by one, and then replayed the turn-OFF frame. However, the reply attack suggested by the work [10] did not make any effect on the bulb. We further increased the frame counter to other higher values (including its maximum value 0xFFFFFFFF), but all replay attacks failed. More aggressive packet editing was performed so that the frame counter, sequence numbers and MIC were modified either separately or collectively. However, those efforts were not able to make a successful attack.



```
284 1595660261.072022 51 4188 b2 91d6 ffff20b30802fdff 20b3 1e 7e 28
6ce73700 6d3fbe010188170000 b05c4a7a4e551f268d85c9913a6f
87816f1b d5f4 255 1 0 26 284 0 1 32767
```

Length:51
Source: 0xb320
Data payload
IEEE layer Sequence Number: 178
Network layer Sequence Number: 126
Message Integrity Code: 87816f1b
Destination PAN: 0xd691
Frame Counter: 3663724
FCS: 0xf4d5

Fig. 11: Characteristics of captured frame for turning OFF the Philips Hue bulb.

VI. Conclusion

Due to its low-cost implementation, replay attack is a critical challenge on maintaining the integrity and security of ZigBee communication network. Although message authentication and encryption based countermeasures have been introduced in literature, those methods cannot thwart all replay attacks. In this work, we propose a new replay attack flow, which only needs cheap commercial off-the-shelf devices API-Mote and open-source software KillerBee to conduct the attack. A noise removal technique is proposed to retrieve the original message transferred between ZigBeed devices. Our case studies show that our attack can be successfully implemented in Xbee and Philips Hue devices and the victim ZigBee device is not able to differentiate a replayed packet from a legitimate message.

References

[1] Hezam Akram Abdul-Ghani and Dimitri Konstantas. A comprehensive study of security and privacy guidelines, threats, and countermeasures: An iot perspective. *Journal of Sensor and Actuator Networks*, 8(2):22, 2019.

[2] Xueqi Fan, Fransisca Susan, William Long, and Li Shangyan. Security analysis of zigbee. *MIT.edu*, 2017.

[3] Aditya Gupta. *The IoT Hacker's Handbook: A Practical Guide to Hacking the Internet of Things*. Apress, 2019.

[4] Digi International Inc. *XBee/XBee-PRO S2C 802.15.4 RF Module User Guide*.

[5] Hongwei Li, Zhongning Jia, and Xiaofeng Xue. Application and analysis of zigbee security services specification. In *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, volume 2, pages 494–497. IEEE, 2010.

[6] Ivan Medvediev, Oleg Illiashenko, Dmytro Uzun, and Anastasiia Strielkina. Iot solutions for health monitoring: analysis and case study. In *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, pages 163–168. IEEE, 2018.

[7] O. Olawumi, K. Haataja, M. Asikainen, N. Vidgren, and P. Toivanen. Three practical attacks against zigbee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned. In *2014 14th International Conference on Hybrid Intelligent Systems*, pages 199–206, 2014.

[8] 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*, pages 5132–5138, 2013.

[9] Joshua Wright. Killerbee: practical zigbee exploitation framework. In *11th ToorCon conference, San Diego*, volume 67, 2009.

[10] Joshua Wright and Johnny Cache. *Hacking exposed wireless: Wireless security secrets & solutions*. McGraw-Hill Education Group, 2015.

[11] Tobias Zillner and Sebastian Strobl. Zigbee exploited: The good, the bad and the ugly. *Black Hat–2015 [Electronic resource].–Access mode: https://www. blackhat. com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly. pdf*, 2015.