Don't Kick Over the Beehive: Attacks and Security Analysis on Zigbee

Xian Wang
University of Texas at Dallas
Richardson, Texas, USA
xxw161030@utdallas.edu

Shuang Hao
University of Texas at Dallas
Richardson, Texas, USA
shao@utdallas.edu

ABSTRACT

The blooming of the Internet of Things (IoT) has led to the demand for new connectivity technologies. Zigbee, with its low-power consumption and flexible network structure, has become one of the essential wireless communication protocols in the IoT ecosystem and is adopted by many major companies. There have been over thousands of certified Zigbee products, ranging from sensors, to link light products, to smart energy devices. A common belief is that Zigbee is comparatively secure, due to the nature of closed networks and the use of encryption.

However, in this paper, we find Zigbee-related attacks in which adversaries are outside of the target network and have no knowledge of the encryption keys. The low attack requirements pose high threats. The attacks mislead devices to accept forged packets, which will jeopardize the normal operations of Zigbee networks. We further develop a framework to efficiently identify such problems. The framework supports flexible modification of address and network information, and bit-level manipulations from the MAC layer. To accelerate analysis, we design semantic-aware fuzzing to generate packet candidates that are more likely to produce meaningful results.

We conduct experiments on 10 real-world Zigbee systems. Using our approach, we have identified five types of practical attacks, ranging from communication disruption to security key leakage. We show proof-of-concept attacks on industry Zigbee products and systems. Our findings bring to light new security issues of Zigbee and further motivate possible mitigation approaches.

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**.

KEYWORDS

Security; Zigbee; Semantic-aware Fuzzing

ACM Reference Format:
Xian Wang and Shuang Hao. 2022. Don't Kick Over the Beehive: Attacks and Security Analysis on Zigbee. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3548606.3560703

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9450-5/22/11…$15.00
https://doi.org/10.1145/3548606.3560703

1 INTRODUCTION

Internet of Things (IoT) technologies extend network connectivities and lead to the promise of new-level convenience and utility, such as home automation [2, 12, 34, 35, 41], remote healthcare [4, 22, 24, 29, 31], self-driving vehicles [75, 78], and smart manufacturing and agriculture [55, 79]. A recent report estimates that the number of deployed IoT devices will reach 27 billion in 2025, and the global market will be valued at three trillion US dollars [37]. The thriving of IoT heightens the demand for new wireless communication technologies. In particular, Zigbee, due to its low-power consumption and security features, has developed into a critical standard for IoT communications and is adopted by many leading companies. There have been over 4,000 certified products listed on the Connectivity Standards Alliance website [21] (previously known as Zigbee Alliance website), ranging from sensors and door locks, to link light products, to smart energy devices. Therefore, understanding and analyzing Zigbee security is an imperative mission.

Challenges. Existing Zigbee-related security studies focus on attacks exploiting key leakage [60, 81] or old-generation designs (such as replay or DoS attacks [16, 57], which are not exploitable in the latest Zigbee protocol). Zigbee network is a closed network. Except during the commissioning step (typically requiring human interactions), the Zigbee network will not allow nodes to join. It lacks systematic approaches to holistically examine Zigbee security, such as with adversaries outside of the network during the closed normal operations.

Moreover, it is challenging to efficiently find meaningful test cases and identify Zigbee vulnerabilities. Zigbee has varied packet formats, which have not been deeply studied yet. Especially the Zigbee protocol uses 128-bit AES-CCM* encryption and generates a 32-bit authentication code (details in Section 2.2). Thus, it is impractical to directly explore the Zigbee packet space for security analysis.

Novel solutions and findings. Different from prior work, we find attacks that target closed Zigbee networks and bypass security mechanisms (threat model in Section 3): specifically, the adversary is outside of the target Zigbee network, has no knowledge of the encryption keys, and does not rely on the short exploitation time window of the commissioning phase (i.e., attacking at arbitrary time). To address the research challenges, we develop a framework (Section 5) to flexibly support modification of address and network information, and generate packets with fine-granularity

manipulation. Other researchers can adapt our framework to conduct further Zigbee security experiments. In addition, we develop semantic-aware fuzzing (Section 6) to prioritize packets with formats following general structures of the Zigbee protocol, which are more likely to produce meaningful results. We examine semantic correlations across multiple Zigbee protocol layers and handle encryption and varied packet structures. Our approaches efficiently explore potential vulnerabilities in Zigbee.

We conducted experiments on 10 real-world Zigbee systems. Using our analysis approach, we have identified five types of practical attacks and demonstrated a variety of Zigbee systems susceptible to similar attacks. These Zigbee systems use different chipsets and different software. Our identified attacks can be launched from outside of the closed Zigbee networks (at arbitrary time during their normal operations, instead of just the short commissioning window). In two types of the attacks, adversaries not authorized by the network can interrupt normally operational IoT devices, consequently stopping intended functionalities (e.g., motion sensors cannot send alarms when there is a break-in). The third type of attack triggers the target device to expose the encryption key, which allows the adversary to control all devices in a network (as the encryption key is used by all devices in a network). In the last two types of attacks, the vulnerable Zigbee systems will accept and process invalid packets. We further investigate the causes of the vulnerabilities in Zigbee systems and provide mitigation suggestions.

Compared to the attacks that we reveal, the other previously known attacks are more difficult to take effect. Jamming attack [54] is a typical threat, which uses physical signal interference and disrupts wireless connections. The Zigbee protocol has already adopted and standardized effective countermeasures against jamming, including DSSS (Direct Sequence Spread Spectrum) [7, 63] using signal spreading and DCA (Dynamic Channel Assignment) [18, 20] providing dynamic channel changes. During our experiments, we have observed that DCA was used in practical devices, where the Zigbee networks will switch to different signal channels when sensing jamming. Compared with jamming which requires sending hundreds of packets per second, our identified attacks send tens of packets per second and find upper layer vulnerabilities.

Another typical class of Zigbee attacks exploit the commissioning phase [49, 51, 52], when new devices are joining the Zigbee networks. Adversaries need to wait until the Zigbee network is activated (by benign users) to accept joining requests, which will be a small time window to launch attacks. In contrast, the attacks that we find can jeopardize Zigbee networks and devices at arbitrary time (not relying on commissioning windows).

In summary, our paper makes the following contributions:

- We find Zigbee-related attacks that can be launched from outside of the target network and do not require the encryption keys. The attacks can exploit Zigbee networks at arbitrary time during closed normal operations, while previous studies mostly target the commissioning phase (which requires human interactions to enable and has a small time window) [49, 51, 52] or old-generation designs [16, 57].
- We develop a framework to efficiently identify previously unknown Zigbee vulnerabilities. We design semantic-aware fuzzing to generate protocol-compliant packets as candidates, which provides fine-granularity manipulation and higher chances to reach meaningful corner cases.
- We conduct experiments on 10 real-world Zigbee products and systems and identify five types of attacks. The attack vectors are rooted from the network layer and can affect upper Zigbee protocol layers. We analyze the causes of the problems and propose possible approaches to mitigate the threats.

2 BACKGROUND

Zigbee is designed for low-power low-bandwidth needs and can transmit data through a mesh wireless network. We provide background on Zigbee network architecture, protocol stack, commissioning process, and its security mechanisms.

2.1 Zigbee Protocol

Zigbee network overview. A node in the Zigbee network refers to physical communication equipment. Every node has a 64-bit globally unique MAC address (also called extended address) set by the manufacturer. In addition, a 16-bit network address (also called logical address or short address) is assigned to a node when it joins a Zigbee network. The network addresses are introduced to comply with the relatively small storage of nodes and the Zigbee packet length limit, and are intended to be only unique and used for communication within the network. Zigbee nodes can be categorized into three types, including controllers, routers, and end devices.

A controller (also called coordinator) manages the Zigbee network and provides permission to all the rest nodes intended to join the network. Every centralized Zigbee network has one controller with a 16-bit Personal Area Network identifier (PAN ID) and a 64-bit Extended PAN ID (EPID). Both identifiers can uniquely define the network. The controller maintains the routing table with MAC address and its corresponding network address for every node. The controller is the only node to update network information (as network key described in Section 2.2). A router acts as an intermediate node between the controller and end devices. A router maintains a routing table and may include end devices' functionalities. We do not use router in the paper although the attacks are also applicable on routers. End devices can communicate with their parent nodes (controller or router), but cannot relay data. End devices support functionalities to interact with physical world and collect data.

Zigbee protocol stack. Zigbee stack architecture consists of five layers as shown in Figure 1. From bottom to top, there are Physical layer, Media Access Control (MAC) layer, Network (NWK) layer, Application Support Sublayer (APS) and Zigbee Cluster Library (ZCL). APS and ZCL are components belonging to the Application layer. Physical layer sets the physical channel (5Mb for each one) of the Zigbee network, which is one of 16 channels identified in ISM band in 802.15.4. MAC and Network layers are responsible for data transmission and routing. The Zigbee node addresses (MAC and network addresses) and the network identifiers (PAN ID and EPID) are included and expressed as cleartext format in MAC and Network layers. APS provides control services for other components of the Application layer. ZCL defines cluster functionality developed by the Connectivity Standards Alliance, where each cluster contains

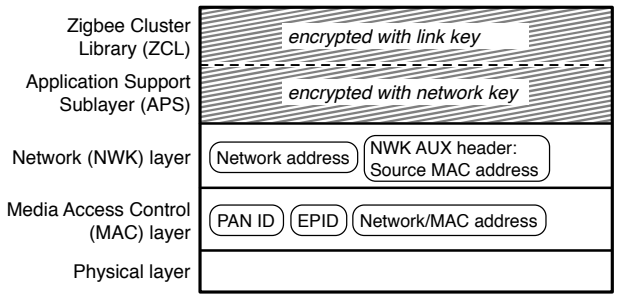

Figure 1: Zigbee protocol stack. The strip pattern indicates encrypted information. APS and ZCL belong to the Application layer. MAC and Network layers contain the cleartext information of MAC and network addresses, PAN ID, and EPID.

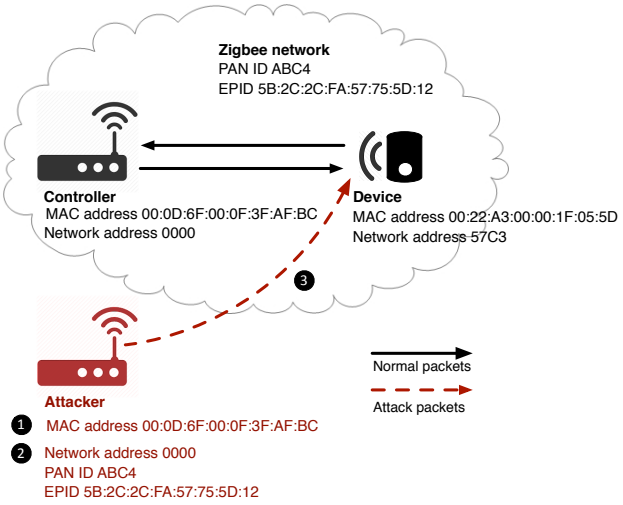

Figure 2: Threat model with new attack scenarios. Zigbee network is a closed network and does not allow unauthorized devices to communicate, but the basic network information (such as device MAC addresses and PAN ID) is unencrypted and can be easily learned through wireless sniffing The attack packets are generated and sent by an outside impersonator, which pretends to use the controller's MAC address (❶) and fills packets with the controller's network address and the target Zigbee network information (e.g., PAN ID) (❷). The target device will process the attack packets (❸) and result in an abnormal status.

a group of commands with attributes. The clusters in turn specify different application profiles, such as Light Link 1.1, Home Automation 1.3, and Health Care 1.0. A Zigbee network can choose to apply encryption mechanisms to protect Application-layer information (see Section 2.2).

2.2 Zigbee Security

Security keys. The Zigbee protocol supports symmetric encryption for secure communications. The encryption algorithm is the Advanced Encryption Standard (AES). The security keys for a Zigbee network include a network key and link keys. A 128-bit network key is distributed during the commissioning process and used to encrypt Network layer payload (i.e., Application layer information). There are 128-bit factory-programmed link keys, which are used to encrypt initial network key transmission or optionally encrypt the payload of APS (i.e., ZCL information). The encrypted parts in the Zigbee protocol stack are shown as the stripe pattern in Figure 1. Though the IEEE 802.15.4 MAC layer has encryption option, due to high power consumption of crypto operations and the challenge of key distribution, MAC layer encryption is not used in practice.

Data encryption and authentication. Zigbee uses AES encryption and the CCM* mode of operation to provide authenticity and confidentiality of data transmissions. The AUX header of the corresponding layer contains fields of `security level`, `frame counter`, and `source address` (e.g., source MAC address at the Network layer), which construct the nonce used in the AES encryption to prevent replay attacks. A Message Integrity Code (MIC) is calculated and appended after the encrypted payload for integrity protection (32-bit long with the default `security level` specified). Directly brute forcing 32-bit MIC is infeasible in terms of time consumption. With a packet transmission rate of 200 packets per second, the brute force of the 32-bit MIC is estimated to take more than 248 days.

3 THREAT MODEL WITH NEW ATTACK SCENARIOS

There are two main factors that make it challenging to compromise Zigbee networks and communications: (1) Zigbee networks have the closed network nature, which are equipped with a dedicated commissioning process to add new devices into the networks. The commissioning process typically requires users' actions to enable the controller to accept joining requests (e.g., pushing a button on the controller). The controller and the new device will exchange EPID, PAN ID, MAC addresses and establish encryption keys during the commissioning phase. Except commissioning, the Zigbee network is closed and the controller will not process joining requests. Packets from unauthorized devices are denied; (2) The Zigbee protocol uses encryption with the AES algorithm and authentication with the CCM* mode to protect the Application layer (i.e., payload of the Network layer). Without knowing the correct security keys (exchanged during the commissioning period), adversaries can not infiltrate Zigbee systems. Consequently, existing attack scenarios focus on exploiting the commissioning phase [1, 51, 52, 61] or side channels with security keys exposed [26, 32, 57, 60, 81].

However, we find Zigbee attacks that can bypass these two constraints. Figure 2 shows our threat model with example attack scenarios. The simplified Zigbee network has two nodes: one Zigbee controller and one device communicate and transfer data. Each node has a 64-bit MAC address and a 16-bit network address (which is assigned during the commissioning process). The Zigbee network is identified by the 16-bit PAN ID and the 64-bit EPID, which are specified by the controller. The attack device in Figure 2 is not in the target Zigbee network (i.e., not authorized) and does not know the

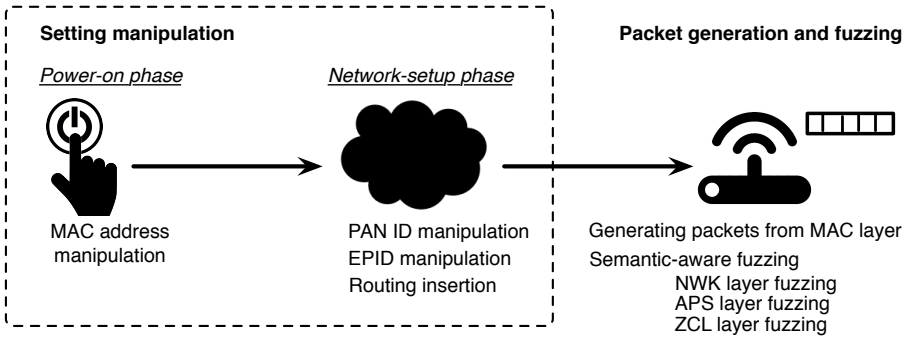

Figure 3: Overview of our analysis framework.

network key. The only requirement is that adversaries sniff publicly available Zigbee network information in the packets over the radio link, including devices' MAC addresses and network addresses, and network PAN ID and EPID. As manifested in Figure 1, this information is unencrypted in the headers of MAC and Network layers.

The attack device will impersonate as a node that is already in the target Zigbee network. Since the controller has the most capabilities in the Zigbee network, we focus on attacks impersonating the controller. Note that the following attack steps can be launched at arbitrary time during the closed normal operations of Zigbee networks (instead of during the commissioning phase).

Attack step ❶. As a prerequisite to communicate on the MAC layer of IEEE 802.15.4, the attack device needs to overwrite its manufacturer-produced physical address and pretend to be the controller in the Zigbee network. The adversary can easily obtain the controller's MAC address by sniffing Zigbee packets, since the MAC addresses are contained as plain text in the headers of MAC and Network layers (as shown in Figure 1). In the example of Figure 2, the attack device sets the MAC address to `00:0D:6F:00:0F:3F:AF:BC`, the same as the controller's address.

Attack step ❷. The attack device further imitates the network identifiers. The controller's network address (`0000`) and the network PAN ID (`ABC4`) can be extracted by eavesdropping on regular Zigbee packets. To obtain the network EPID, the adversary can broadcast a beacon request. Then the controller will send back the EPID (`5B:2C:2C:FA:57:75:5D:12`) in a beacon response (meanwhile announcing the network is closed, not accepting joining requests). Next, the adversary selects a target device (or multiple devices) as the attack target. Similarly, the target device's MAC and network addresses (`00:22:A3:00:00:1F:05:5D` and `57C3` in the example of Figure 2) can be easily obtained from packet sniffing. With the setup of the matching network information, the attack device can fake as the controller to send packets to the target device.

Attack step ❸. The attack device constructs packets and injects them into the Zigbee network. The goal of the adversary is to cause the target device to process the forged packets and end up in dysfunctional statuses. Though regular Zigbee communications use encryption on the Network layer payload, packets crafted with specific control fields and commands can induce vulnerabilities. Section 7 shows five detailed attacks that we have identified. These attacks can force Zigbee devices to disconnect from the network or leak the encryption keys.

The attacks rely on minimal conditions. The adversary is not within the target Zigbee network, does not need to know the encryption keys, and does not rely on the commissioning phase. All required information can be easily obtained through wireless traffic sniffing (e.g., to obtain MAC addresses and PAN ID) or active probing (e.g., to obtain EPID). The threat model is compatible with the capabilities of real-world unprivileged adversaries.

4 OVERVIEW OF OUR ANALYSIS APPROACH

We develop a framework to systematically analyze the Zigbee protocol and identify weaknesses that adversaries can exploit to launch the aforementioned attacks. The general idea is to simulate an attack device, enumerate potentially problematic control fields and commands to generate packets for evaluation, and trace defective statuses in the Zigbee systems.

Development platform. We choose to build the framework based on the Texas Instruments CC2538 evaluation module, which provides programmable functionality for Zigbee development. The CC2538 module supports Zigbee 3.0, the latest version of the Zigbee standard. The module implements the Zigbee protocol as Z-stack (in C). More experiment setting details are described in Section 7.

Our framework development provides flexible analysis for Zigbee networks and the design includes two parts: setting manipulation and packet fuzzing. Figure 3 demonstrates the overview of our framework design.

Flexible setting manipulation (Section 5). To support flexible configurations and packet transmission, we manipulate the framework module in two phases. *Power-on phase* (Section 5.1): When the framework module is powered on, we modify the loading procedure and overwrite the MAC address of the framework module. We can set arbitrary MAC address for the framework module to interact with a Zigbee network. *Network-setup phase* (Section 5.2): When the radio antenna is activated, we manipulate the network information in the framework module. We modify the network formation procedure to assign the PAN ID and EPID that we select. In the routing table, we directly insert the destination node information for experiment.

Our framework is flexible to configure device and network settings, and other researchers can adapt it to conduct a wide variety of Zigbee IoT experiments, e.g., testing management during application development. In this paper, we use our framework to simulate the attacker's capability (i.e., outside of the target network and not experiencing the commissioning phase) and generate packets for fuzzing. We configure the framework module with the MAC address of the controller node, assign the PAN ID and EPID as the target network, and insert routing paths to the target devices. More details of the framework development are described in Section 5.

Semantic-aware packet fuzzing (Section 6). The goal is to identify packets that can cause Zigbee systems to fall into anomalous states (such as disconnection). We generate packets based on bit manipulations from the MAC layer. The low-level operations enable fine-granularity control over the packet components. To increase analysis speed, we develop *semantic-aware* fuzzing, which leads to packets more likely to be processed by the destination node and explore meaningful corner cases. We fuzz various fields and commands to find packets that result in attack vectors. Since regular Zigbee networks use AES-CCM* protection (described in Section 2.2), we also devote to bypassing encryption mechanisms. We elaborate our semantic-aware fuzzing in Section 6.

Constructing and fuzzing packets from the MAC layer. Next we describe our design choice of assembling packets on the MAC layer. On each layer (e.g., MAC, Network, or APS layer) there are independent functions to generate datagram. As shown in Figure 1, either APS layer (the payload of Network layer) or ZCL layer (the payload of APS layer) can be encrypted. For example, to handle the encrypted APS layer, we need to manipulate the Network layer. However, the Network layer functions provide limited capability to modify its own content. To solve the problem, we instead use the MAC layer functions to edit the MAC payload directly, which provides complete bit-level manipulation capabilities for the Network layer and other upper layers. Constructing packets from the MAC layer allows fine-granularity control over packet details to find vulnerabilities.

5 DESIGN OF FLEXIBLE ANALYSIS FRAMEWORK

In this section, we introduce the details of our analysis framework. First, we explain how to modify the MAC address at the power-on phase. Then we illustrate configurations of network information to allow the framework module to send forged packets to the target Zigbee network.

5.1 Manipulation for Power-on Phase

As the first phase of the setup, we need the capability of arbitrarily manipulating the MAC address of the framework module. The original MAC address is specified by the manufacturer and is automatically loaded every time when the device is booted up. We add the code in the implementation to replace the MAC address with a different value, after the step that the MAC address is fetched from one of the sources (including non-volatile memory, flash memory, or random generation). The new MAC address that we specify will be loaded in the framework module at booting time to use for the communications.

5.2 Manipulation for Network-setup Phase

A more challenging task is to establish network connections from the framework module (playing as an adversary from outside of the network) to the target network's devices. The Zigbee network is a closed network. During normal operations, the network does not accept association requests, and authorized nodes have their roles in the network (such as controller, router, or end device). However, a new device is not recognized by the network and can not play any role.

The general idea of our approach is that instead of attempting to join the target Zigbee network directly, we make the framework module (as a controller) create a separate network with the identical network identifiers as the target network, and straightly insert rules to route to the target devices (which are the devices in the target network). Therefore, when the framework module sends packets to its "own" device, it actually transmits to the device in the target network.

We exploit the Zigbee network formation procedure to enforce network-setup manipulation. Network formation needs three steps from a controller: (1) enabling the radio antenna, (2) setting PAN ID and EPID, and (3) opening to accept association requests. The framework module is configured in (2) generating the target network PAN ID and EPID and (3) adding routing path of the target device without commissioning. Note that so far the module hasn't sent packets and affected the target network. To avoid the original controller from interfering with the network formation, the setup is conducted at a place outside of the transmission range of the target Zigbee network (after the needed information is sniffed as in Section 3). Below we describe manipulation details.

PAN ID manipulation. The 16-bit PAN ID is contained in every Zigbee packet (in the MAC layer header) to identify the associated network. We aim to modify the PAN ID of the new Zigbee network that the framework module creates, and make it the same as the PAN ID of the target network. In the implementation, the PAN ID is specified in a configuration file. The PAN ID of the target network can be easily obtained through wireless traffic sniffing, since it is transmitted unencrypted. If we put the same value in the configuration file, the framework module will use it as the PAN ID when constructing a new network.

EPID manipulation. The 64-bit EPID is a long version of the unique network identifier (counterpart of the 16-bit PAN ID). Zigbee mainly uses PAN ID in regular communications, while EPID is used in commissioning or rejoining process (in beacon packets to uniquely identify a Zigbee network). As described in Section 3, an adversary can acquire the EPIDs of the nearby Zigbee networks by simply broadcasting a beacon request. During network formation, we assign the EPID to the newly created network by setting the specific value.

Routing insertion. Though the framework module has generated a new network with the exact identifiers as the target network, this network does not contain any other node yet. Our goal is to induce the framework module to believe that a target device is associated with this new network and stand-by for communications. At the

```
1  static void zclGenericApp_HandleKeys(byte shift,
2                                        byte keys )
3  {
4    ...
5    if (keys & HAL_KEY_SW_1)//if pushing button 1
6    {
7      uint16 DestShortAddr=0x57C3;
8      uint8 DestExtAddr[Z_EXTADDR_LEN]={
9                                0x00,0x22,0xA3,0x00,
10                               0x00,0x1F,0x05,0x5D};
11     Dest = AssocAddNew(DestShortAddr,
12                        DestExtAddr,
13                        CHILD_RFD_RX_IDLE);
14   }
15   ...
16 }
```

Listing 1: Example code that we use to add routing information about a node (destination) with network address `0x57C3` and MAC address `00:22:A3:00:00:1F:05:5D`. Function `zclGenericApp_HandleKeys()` handles events triggered by pushing buttons. The code will be executed if we push button 1.

end of network formation, the framework module will be in the open mode for a couple of hundred seconds to accept new nodes to join. However, a device has no incentive to actively send association requests (since it is in a stable connection status with the original network).

We find an alternative solution through the function `AssocAddNew(A,B,C)`. The function maps the network address (`A`) and the MAC address (`B`) of a new node with relationship between the two nodes (`C`). The relationship type in our framework module is a parent (controller). After the module powers up, we push button 1 in the function `zclGenericApp_HandleKeys()` (on line 5 in Listing 1) to execute the routing insertion. The framework module maps the network address `0x57C3` and the MAC address `00:22:A3:00:00:1F:05:5D` of the target device (in Figure 2) in the routing table as an associated node without commissioning. The module does not have the knowledge of the network key of the target network, which corresponds to our threat model presented in Section 3.

6 SEMANTIC-AWARE ZIGBEE PACKET FUZZING

This section describes how we use the framework and generate candidate packets to find vulnerabilities in Zigbee. First, we systematically explore the semantics of Zigbee packets to analyze encryption and varied packet structures. Second, we fuzz fields and commands to identify packets that can cause Zigbee networks to be in abnormal states.

6.1 Semantically Exploring Zigbee Packet Structure

A straw-man approach is to put random contents in the generated packets and blindly test whether to cause Zigbee networks to malfunction. However, such an approach has poor efficiency, since many generated packets have broken formats and will be rejected directly by the target node. For example, even the shortest Network layer packets have 32 bits[1] as the exploration space of random fuzzing. The maximum packet transmission rate of the antenna that we use is less than 200 packets per second. The brute force will take more than 248 days to examine the NWK packets with the shortest lengths. Our goal is to explore Zigbee packets not only on the NWK layer, but also on the APS and ZCL layers.

To accelerate analysis speed, we instead generate protocol-compliant packets and dissect detailed fields in the packet. There are two challenges that we address: (1) Zigbee uses encryption for data transmission, and (2) packets have varied lengths and formats according to different header values.

Managing encryption and authentication. As described in Section 2.2, Zigbee uses 128-bit AES-CCM* encryption and generates a 32-bit MIC, which can prevent replay attacks or plaintext brute force attacks (i.e., enumerating cyphertext to hit a meaningful plaintext). In our threat model, the adversary does not know the network key of the target Zigbee network. Without the correct key to apply, the packets cannot be accepted and read properly by the other node.

We examine unencrypted fields in the Zigbee packets (e.g., MAC or network headers) and find ways to transmit plaintext that can cause the target node to process. To have the payload of a layer encrypted, there are three security-related fields on that layer: `security enabled bit` (in the frame control field of the layer header), security AUX header (used to construct a nonce), and the MIC (appended after the payload for integrity checking). Figure 4 demonstrates the field locations on the Network layer. Among these parts, the `security enabled bit` plays a decisive role. If the `security enabled bit` is set to 0, the layer payload will not deploy any security mechanism. Accordingly, the layer will not include the AUX header (due to no data encryption) or MIC. The setting bypasses the security mechanisms, and the forged unencrypted messages can be received and affect the target device.

If the `security enabled bit` is set to 1, the payload will be encrypted along with the MIC appended after the payload. The packets that we generate without the knowledge of the network key a priori are not valid packets, but they may cause negative impacts if the systems have implementation flaws. The Zigbee 128-bit AES-CCM* encryption uses the "counter mode", which allows generating encryption output with arbitrary length (the output can be shorter than 128 bits). In the "counter mode", the 128-bit AES encrypts an incremental nonce (not directly encrypts the plaintext), and then XOR with the plaintext. The ciphertext with the same length as the original plaintext is sufficient for decipher, e.g., one byte of data will be encrypted and transmitted as one-byte ciphertext. In Section 6.2, we develop strategies to fuzz and explore the encrypted NWK payload and the MIC.

Analyzing varied packet formats. In addition to encryption, Zigbee has varied packet formats, which makes existing protocol fuzzing approaches implausible [27, 68, 73]. First, the header values cause different header lengths and consequently change the packet structures. Zigbee uses such dynamic strategy to make the packets as short as possible. For example, the `security enabled bit` mentioned above will add the AUX header (as shown in Figure 4),

[1]Two bytes of frame control field in the header, one byte of NWK command, and one byte of command attribute. The header also contains two bytes of source network address and two bytes of destination network address, which we do not count in the exploration space.

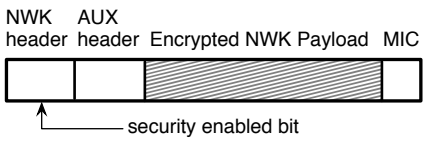

Figure 4: Packet structure on the Network layer (security-related information).

the two-bit `frame type` on the NWK layer determines whether the payload corresponds to a NWK command or the upper APS layer, and the setting of the `IEEE address included` field will add the eight-byte MAC addresses (in addition to the short network addresses) in the header. Therefore, the Zigbee packet structure varies dramatically based on the header values. Second, in the payload, commands and attribute parameters are correlated and require different lengths. For example, `attribute IDs` have different data types to achieve various functionalities. We need to recover the specific commands and the associated data structures to make meaningful choices when mutating the data.

To address the challenges, we analyze the Zigbee packet structure by the following steps. We actively enumerate each individual field by sending packets and examining the format changes from the captured Zigbee traffic. There are two new steps that we design for Zigbee analysis: one is to decide the header fields to find the fuzzing locations, and the other is to retrieve the fuzzing ranges of commands and parameters (for the NWK, APS, and ZCL layers). First, every layer header has a frame control field (two bytes in the NWK layer, one byte in the APS layer and one byte in the ZCL layer), which decides the other header fields and whether the packet contains upper layers. We enumerate the frame control values bit by bit to find the corresponding structure changes and later construct protocol-compliant lower layers when we fuzz the upper layers. Second, our fuzzing focuses on the `cluster ID`, `command ID`, and `attribute ID` in different upper layers, because they decide the packet functions. These fields are correlated and across different layers. For example, different `command IDs` require different lengths of `attribute IDs`, and the `cluster ID` in the APS header determines the ZCL layer `command ID` (at a different layer). We test the maximum and minimum values of clusters, commands, and attributes to find the mapping ranges and meaningful values for fuzzing. Our fuzzing details per layer are elaborated in Section 6.2.

6.2 Fuzzing Details and Anomaly Locating

We fuzz the fields of the packets ranging from the Network (NWK) layer, to the APS, to the ZCL layer. We consider the semantics and relationship across different layers. To set the packet details, we construct each layer's header and payload (based on the MAC layer operations) and leverage the structure of Zigbee commands and attributes to ensure the generated packets having protocol-compliant formats.

Network (NWK) layer fuzzing. We first configure the Network layer header and payload. In particular, the frame control byte in the Network header has the `security enabled bit` that can be set to 0 or 1 as mentioned in Section 6.1. The network header can use three types of address settings (source and destination network address, source and destination network address and source MAC address, source and destination network address and source and destination MAC address) and two security bit settings (0 and 1). The maximum length of the Zigbee packet should be less than 128 bytes as specified in the standard [28]. Network layer packets include `command ID` in this layer. Each network command packet has its own attribute to be fuzzed. There are 13 valid network commands. Each command has one attribute (one byte). On the Network layer, if security enabled bit set as 0, there are 3,328 ($= 13 \times 256$) combinations in total.

If the security enabled bit set as 1, the network payload is encrypted and the MIC is added for integrity check. On the bit level, we can put random data into the packets to explore. However, it is infeasible to brute force the entire space. As described in Section 6.1, the "counter mode" of Zigbee encryption preserves the content lengths, so we will leverage the length structure of Zigbee commands to prioritize cases that are likely to lead to meaningful results. All defined Network commands are one byte, and we add at most another one byte as attributes. Since we do not have the network key, we set random MIC values (to fulfill the MIC lengths) and fuzz the payload only with the lengths of possible commands. To examine the flaws of bypassing the integrity check, we fuzz encryption payload lengths of 8 and 16 bits. There are 65,792 combinations in this category (the sum of 2^8 and 2^{16}). The NWK header has 3 unencrypted bits that can be fuzzed for different packet settings. The total fuzzing number is 526,336 ($= 65792 \times 2^3$) that we explore under the security enabled bit set as 1.

#	*Event*	*Implied anomaly*
①	No response more than 10 minutes	Normal communication being interrupted
②	Beacon request	Disconnection and attempts of reconnection
③	Beacon requests for 10 minutes or longer	Continual failed reconnection attempts
4	Unencrypted packets	Information leakage
5	No MIC for encrypted packets	Possible brute force on ciphertext
6	Network key transmitted in plaintext	Key leakage
⑦	Network key encrypted with a known key	Key leakage
8	High RSSI (Received Signal Strength Indicator) in packets	Increase of battery power consumption
⑨	Response packets delayed more than 3 seconds to receive	Communication lagging
10	Sleep mode packet	Enforcement of device sleeping
11	Relay list in the network header changed	Routing path change

Table 1: Potential anomaly events that we define. The indexes in circles indicate the anomalies that we eventually find in our fuzzing experiments.

Equipment	*Role*
Vendor-A alarm system	Target
Vendor-B integration system	Target
Vendor-C light system	Target
Vendor-D lock system	Target
Vendor-E leak detection system	Target
Vendor-F thermostat system	Target
Vendor-G electronics plug system	Target
Vendor-H air monitor system	Target
Vendor-I door sensor system	Target
Vendor-J prototype system	Target
Programmable module CC2538	Packet generator
USB dongle CC2531	Sniffer

Table 2: Equipment used in our experiments.

APS layer fuzzing. For upper layers, we continue to leverage our framework module to enumerate APS and ZCL fields on a bit level and use packet sniffing tools to identify meaningful values. We introduce the fields to fuzz in the generated packets that have potential to trigger abnormal states. In the APS header, the `profile ID`, such as Home Automation (`0x0104`), indicates a specific application. Each profile corresponds to a group of clusters used in the packet as a container for commands (defined in ZCL). The cluster is specified by the `cluster ID` in the APS header, such as Power Configuration (`0x0001`), For the Home Automation profile, there are 113 valid clusters. The source and destination `endpoint` fields are for the purpose of multiplexing, and we assign the default value 1. There exists difference between the APS layer packets fuzzing parameters and ZCL layer packets. There is command ID in APS layer packets. So the fuzzing parameters in the APS layer are command ID and its related attributes. There are 16 valid APS commands. Since APS commands are mostly related to security information or the routing address of the network, we do not fuzz the attributes in APS commands. We just fix the address parameters with the target device and the controller addresses, and set all security information to 0 (we have no knowledge of security information). Besides, there is no cluster ID, profile ID and endpoints in APS header in APS layer packets, so we do not need to set or fuzz these parameters. On the APS layer, there are just 16 combinations.

ZCL layer fuzzing. The ZCL header has `command ID` that causes actions or generates responses on the recipient device. There are a total of 22 `command IDs`. Example commands include write, read, or configure. The ZCL payload contains attributes and data types. An `attribute ID`, e.g., Hardware Version (`0x0003`), represents the data for the command. There are 21 defined attribute values. Each attribute will be correlated with a data type, e.g., `signed 16-bit integer`. There are a total of 56 data types defined on ZCL layer. For a fixed `profile ID`, we will generate candidate packets by enumerating different values through `cluster ID`, `command ID`, `attribute ID`, and data type, which leads to 2,923,536 combinations ($= 113 \times 22 \times 21 \times 56$).

Locating attack packets. Combining NWK, APS, and ZCL layers, we have 3,453,216 combinations to fuzz. To efficiently locate particular attack packets that can impact the Zigbee systems, we develop a binary search approach. First, we divide all packets into smaller groups (e.g., 120,000 per group) for easier experiment setup. Second, for packets in each group, we send them repeatedly to examine whether negative impact occurs on the evaluation system (such as disconnection). If we observe anomalies, it indicates that there are potential attack packets. We divide the sequence into half, and test the first half sequence and the second half sequence separately. Through the iteration process, we can narrow down the sequence containing attack packets. When evaluating a sequence of candidate packets, we send the same sequence repeatedly, so there are enough opportunities for a specific packet to affect the target system. Third, when the sequence size reduces to around 10, we change to evaluate individual packet one by one. For each candidate packet, we analyze the minimum number it may cause the attack by sending fixed number of packets.

Identifying anomalies. We monitor the traffic of the target Zigbee network to examine whether the fuzzing causes anomalous states. We define a list of possible anomaly events in Table 1. The first column shows the index numbers, and the circled indexes indicate the anomalies that we eventually find in the experiments (in Section 7.2, we specify the observed anomaly events by their indexes). The second column describes the anomaly events that we expect to observe. The last column further explains the attack results. For example, if we observe the beacon request in Zigbee traffic (anomaly ②), it indicates that the attack causes the device to disconnect from the network and attempt rejoining (with the beacon request). We acknowledge that our anomaly definition is not able to catch all potential vulnerabilities. However, with our fuzzing framework, vendors and developers can extend the defined anomaly events regarding their products to find more specific vulnerabilities.

7 EXPERIMENTS AND FINDINGS

This section describes the experiments we conducted in real Zigbee systems and the results of examining the transmission packets. First, we introduce the experiment settings that we use for analysis. Then we detail the five attacks that we find by using fuzzing tests and generating protocol-compliant packets.

7.1 Experiment Setup

In our experiments, we used 10 types of equipment to find potential vulnerabilities. To avoid targeted attacks, we anonymize vendor-specific information in the paper. Table 2 lists the Zigbee devices and systems used in our experiments. These systems use various chipsets and software. We programmed the module CC2538 [39] to generate and send Zigbee packets. We also used USB dongle CC2531 [38] to sniff Zigbee packets and analyze the packets in Ubiqua protocol analyzer [69]. In the proof-of-concept attacks, we set the transmission rate as 100ms (i.e., 10 packets per second). We tested physical distances to launch attacks. With enhanced antennae, attacks are successful up to a distance around 60 meters.

7.2 Identified Attacks and Analysis

We identify 39 parameter settings that lead to five attack categories. Our fuzzing takes 57–120 hours depending on the number of found vulnerabilities. Table 3 shows the summary of the attacks. Each row represents one identified attack. The first column indicates the attack category. The second column shows the corresponding layer (either on NWK, APS, or ZCL). The rest 10 columns represent

Attack category	*Layer*	Vendor-A *alarm system*	Vendor-B *integration system*	Vendor-C *light system*	Vendor-D *lock system*	Vendor-E *leak detection system*	Vendor-F *thermostat system*	Vendor-G *electronics plug system*	Vendor-H *air monitor system*	Vendor-I *door sensor system*	Vendor-J *prototype system*
Communication interruption	ZCL		●		●	●		●	●	●	
	ZCL		●		●	●		●	●	●	
Disconnection	APS		●	●	●			●	●	●	●
	APS		●	●			●	●	●	●	
	APS	●	●	●	●		●	●	●	●	●
	APS	●			●	●	●	●	●	●	●
	APS	●	●	●	●			●	●	●	●
	APS	●	●			●	●	●	●		●
	APS	●		●				●	●	●	
	APS	●	●	●	●	●	●	●	●	●	●
	APS	●				●		●	●	●	●
	APS	●		●	●	●	●	●	●	●	
	NWK	●				●		●	●	●	●
	NWK	●			●	●	●	●		●	●
	NWK		●	●	●			●	●	●	●
	NWK		●	●	●	●		●	●	●	●
	NWK		●	●	●			●	●	●	●
	NWK	●	●		●		●	●	●	●	
	NWK	●	●	●	●	●	●	●	●	●	
	NWK	●	●		●	●		●	●	●	
	NWK	●			●			●	●	●	●
	NWK	●	●	●	●		●	●	●	●	●
Key leakage	ZCL	●	●								
	ZCL	●	●								
	ZCL	●	●								
	ZCL	●	●								
	ZCL	●	●								
	ZCL	●	●								
	ZCL	●	●								
	ZCL	●	●								
	ZCL	●	●								
Improper integrity check	NWK*	●		●		●	●	●	●	●	●
	NWK*	●	●		●		●	●	●	●	●
	NWK*	●	●	●	●	●	●	●	●		●
	NWK*	●		●	●	●		●	●	●	
	NWK*	●	●	●	●	●	●	●	●	●	
	NWK*	●	●	●	●	●	●	●	●	●	●
Truncated packet	NH*	●		●	●	●	●	●	●	●	●
	NH*			●	●	●	●	●	●	●	●

Table 3: Identified attacks on 10 Zigbee systems. Each row represents one identified attack. The ● labels indicate the Zigbee systems are affected. The * signs in Improper integrity check and Truncated packet indicate the packets have security enabled bit set as 1. NH in Truncated packet indicates network header. The detailed analysis is at each attack description in Section 7.2.

whether an attack is applicable to each Zigbee system respectively. We use our Zigbee semantic-aware fuzzing method developed in Section 6.2 to identify vulnerabilities across different packet layers. Next, we will describe the details of each attack category and analyze the root causes.

Attack 1: Communication interruption. The first attack can suppress communication from Zigbee devices (observing anomaly ① from Table 1). As shown in Table 3, our experiments find two ZCL `cluster IDs`, which cause such problems on six Zigbee systems. The attack can be triggered within 25 seconds (with settings and transmission rate as described in Section 7.1). The target device will stop sending any response to the controller (while the controller still believes the device is connected in the network). The communication interruption continues multiple minutes (sometimes over one hour). Consequently, the device can not fulfill the intended functions. Figure 5 shows examples of regular and interrupted responses (no response for around one hour based on the timestamps).

Cause analysis of Attack 1 (ZCL layer). This attack exploits the vulnerabilities of negative effects with combination of clusters, attributes and commands, which is the cluster functionality (as in Section 2.1) located in the ZCL layer. There are two main reasons to trigger this attack. First, the device is affected by the unencrypted packets. In the devices, there is no definition of such unencrypted packet processing. However, our experiment shows that the unencrypted packets affect the target device. Second, the device processes the packets that have clusters not defined in the application. Each application defines specific clusters. The application should only process the packets with the defined clusters. But we find undefined `cluster IDs` affect the behavior of Zigbee devices.

Attack 2: Disconnection. The second attack category is able to cause the target devices to lose connection with their networks (observing anomaly ② and ③ from Table 1). Such attacks are applicable on all 10 Zigbee systems. As shown in Table 3, we find vulnerable commands on NWK and APS layers. An attack can be accomplished in 75 seconds (with settings and transmission rate as described in Section 7.1). The device disconnects and attempts to rejoin. On the Zigbee systems, the rejoin procedure does not succeed or repeats several times, and the system loses connection during this period.

Cause analysis of Attack 2 (NWK and APS layers). The attack causes negative effects to the target device by commands in the NWK and APS layers, which are responsible for the routing service and control service (as in Section 2.1). The root cause is that the vulnerable systems accept unencrypted packets and undefined commands. In addition, the device cannot rejoin the network due to the `leave` packet sent by the Zigbee controller. During the rejoin procedure, the device sends rejoin request to the controller, and then the controller replies with rejoin response to the device. Next the device should send a `device announcement` to broadcast the device information. In some systems, we observe that the `device announcement` packet is not sent, and the controller sends a `leave` packet to indicate the device to rejoin the network again, which results in rejoin failures. Figure 6 shows the comparison between regular and failure rejoin packets. Some other systems can finish sending the `device announcement` to the target device, but

Timestamp	Packet Information	APS Counter
18:20:56.029696	Door Lock: Default Response	1
18:21:05.197289	Door Lock: Default Response	2
18:21:15.382417	Door Lock: Default Response	3
18:21:25.553770	Door Lock: Default Response	4

Timestamp	Packet Information	APS Counter
19:40:57.768680	Door Lock: Default Response	4
⋮		
20:44:03.837264	Door Lock: Default Response	5

Figure 5: Attack 1 comparison between regular and interrupted responses. The left example shows the regular responses (timestamp interval is about 10 seconds), and the right example show the interrupted case (timestamp interval is one hour).

	Layer	Packet Information	MAC Src.	MAC Dst.
1	NWK	Rejoin Request	0x57C3	0x0000
2	NWK	Rejoin Response	0x0000	0x57C3
3	ZDP	Device Announce	0x57C3	0x0000

	Layer	Packet Information	MAC Src.	MAC Dst.
1	NWK	Rejoin Request	0x57C3	0x0000
2	NWK	Rejoin Response	0x0000	0x57C3
3	NWK	Leave	0x0000	0x57C3

Figure 6: Attack 2 comparison between the regular and failure rejoin process. The top example shows the regular case, and the bottom example shows the failure case.

there are frequent multiple disconnections causing communication problems. Some systems cannot rejoin and need manual reset.

Attack 3: Key leakage. This attack will cause vulnerable systems to leak security information (network key) to the adversary (observing anomaly ⑦ from Table 1). The network key is being used by all the devices in the network. Therefore, with the network key leaked, an attacker can send arbitrary commands to all devices in the target network. We find ZCL `cluster IDs` that can cause key leakage on the Zigbee systems. The attack packets will cause the device to send rejoin requests. During the rejoin process, the controller will resend the network key which is only encrypted by the default link key (known to the public). With the publicly known link key and the frame counter in the packet (plaintext), we are able to decrypt and retrieve the network key of the network. Figure 7 shows an example packet that leaks the network key. In particular, the APS `command ID` is `0x05` for "Transport Key". If the network key is leaked from one vulnerable device, an attacker can use it to control all other devices (e.g., to unlock an intact Zigbee door lock) in the same Zigbee network.

Cause analysis of Attack 3 (APS layer). The attack exploits the vulnerabilities of security information transmission controlled by the APS layer. There are two causes that lead to the key leakage. First, the unencrypted rejoin packets sent by the device make the controller resend the network key. During the rejoin process, the device and the controller exchange rejoin request, rejoin response and device information broadcast with encrypted packets. However, the target device sends unencrypted rejoin request to the controller. The controller needs to inform the security information to the device. Second, many Zigbee devices send the network key with a publicly known link key. Adversaries can get the information encrypted with the link key. Typically the network key is only exchanged in the commissioning phase. However, our attack can induce the unsafe network key transmission at regular operation time.

Attack 4: Improper integrity check. This attack causes the target device in an abnormal state with manipulated integrity code packets. The attack is applicable in all the 10 Zigbee systems. As shown in Figure 8, we set the security enabled bit as 1 and manipulate the MIC. As shown in Table 3, we find the problem in NWK command packets. In all the 10 systems, disconnection is observed with identified command packets (observing anomaly ② from Table 1, with security enabled bit set as 1). The Zigbee systems fall into repeated rejoin processes, failing to fulfill normal functions.

Cause analysis of Attack 4 (NWK layer). The attack exploits improper integrity check in the NWK layer. There are two main reasons of the attack. The first reason is the improper integrity check of the incoming packets. In standard Zigbee protocol [8], the incoming packet should check the payload and MIC. But in the experiment, the device is affected by invalid MIC packets. The second reason is that Network layer payload length is small, at most two bytes. As described in Section 6.1 and 6.2, due to Zigbee "counter mode" encryption, fuzzing encrypted payload on the Network layer is feasible to reach defined commands that affect the device.

Attack 5: Truncated packet. This attack causes response lagging on the target device which delays data transmission. The attack is able to affect nine of the 10 Zigbee systems (observing anomaly ⑨ from Table 1). The transmitting packets in this attack also set the security enabled bit as 1. As the example in Figure 9, different from normal packets, there is no payload and MIC in truncated packets. The packet only consists of network header and network AUX header. As shown in Table 3, we find two settings of such truncated packets can cause problems during the experiments.

Cause analysis of Attack 5 (NWK layer). The truncated packets include the NWK layer but incomplete payload. The attack explores the potential problems of packet processing in the NWK and APS layers. The main reason is that the target device processes the truncated packets and interferes with the normal communication with the controller. To process the incoming packet, the device matches the address in the network header and the addresses stored in the device, and does the decryption. The truncated packet with correct network header can consume the device's resources in address matching and decryption. The consumption of resources will cause lagging in the normal communication of the target devices.

Efficiency comparison with blind fuzzing. We examine the efficiency of our semantic-aware fuzzing by comparing with blind fuzzing in practice. Since it is implausible to brute force fuzzing the entire Zigbee header as described in Section 6.1, we consider a specific blind fuzzing limited on the frame control fields in different layer headers. Different values in frame control fields will affect the Zigbee packet structure (see Section 6.1). For an identified attack, we keep the same payload, fuzz the frame control fields, and calculate how many packets can cause the successful attacks. One device is used as the comparison target. We first check the NWK layer, where the frame control field has two bytes. We choose the NWK command `09` in Attack 2 for testing, and the ratio between all blind fuzzing packets and the ones that can successfully trigger the attacks is

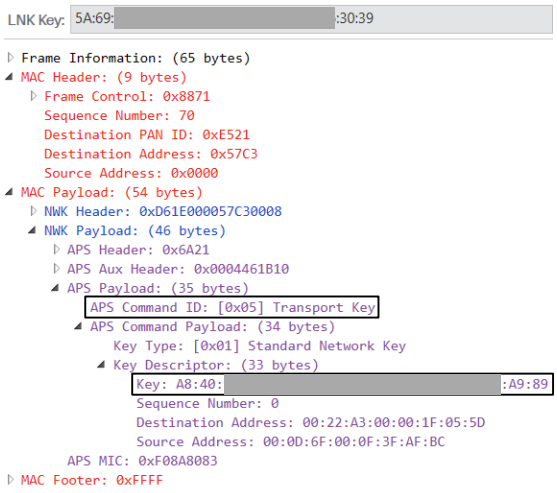

Figure 7: Attack 3 example packet that leaks the network key (with the black boxes). Given the publicly known link key, it can retrieve the network key.

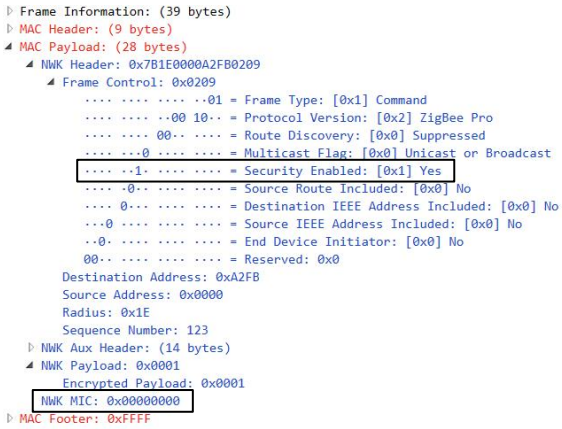

Figure 8: Attack 4 example packet with the security bit as 1 and invalid MIC (highlighted with the black boxes).

139 : 1. The observation demonstrates that the tested blind fuzzing takes 139 times of packets to find the vulnerability that our fuzzing approach identified. We further consider both NWK and APS layers, where the APS layer adds its own one-byte frame control field. We examine the APS command `0D` in Attack 2, and the ratio increases to 2962 : 1. Considering the potential variations of the other header fields and commands, our semantic-aware approach for Zigbee is more efficient to generate candidates that produce meaningful results.

Evaluation over multiple trials. We experimented on the fuzzing over multiple trials and compared performance with the standard protocol fuzzer Sulley [25]. We also experimented on blind fuzzing and it found no vulnerability in the experiments, as the efficiency of blind fuzzing is too low. Sulley is not originally designed for Zigbee, and we adapted Sulley to apply on Zigbee packets. We experimented on a fuzzing subspace of the NWK layer and used the first 5,000 test cases for comparison. One target device was selected to demonstrate the results. Figure 10 shows the results and comparison over three trials. The x-axis represents the trials, and the y-axis indicates the number of identified attacks. We observe that our semantic-aware fuzzing is more effective in terms of identified vulnerability numbers. In addition, different trials have similar results, indicating the performance is stable and converges over multiple trials.

8 DISCUSSION

Responsible disclosure. We reported our identified Zigbee vulnerabilities to the Connectivity Standards Alliance and the 10 Zigbee product vendors in June and July 2022. The Connectivity Standards Alliance responded to our request quickly and aimed to investigate vulnerability causes. For the vendors, we reported details to the corresponding product security teams. Eight vendors have responded to our requests and discussed vulnerability details (two vendors have not responded), and four vendors have acknowledged our findings. The attacks can be launched from outside of the closed Zigbee networks. Two vendors have designated high severity for the preliminary rating of our identified security vulnerabilities, and one vendor committed to fix the issues through firmware updates. The feedback from the alliance and the vendors validates the significance of our findings. Our developed approaches and findings facilitate effectively scrutinizing and improving Zigbee security.

Mitigation discussions. We suggest countermeasures based on the packet layers (NWK, APS and ZCL layers) as follows.

NWK layer. The network (NWK) layer provides network maintenance (as in Section 2.1). There is a command in the NWK layer (network status, command ID `03`) to check the address conflict and verify addresses [8]. To mitigate the Disconnection attack (Attack 2 in 7.2), both the controller and the device can use the network status command to check the connection status of the network. The NWK layer encrypts the payload and generates the integrity code for authentication (as in Section 2.2). The available authentication code lengths are 32, 64 and 128 bits [8]. To mitigate the Improper integrity check attack (Attack 4 in 7.2), we can redefine the authentication code with more flexible lengths (e.g., 22, 24, 26 bits, etc.). The authentication process load can be lower and the protection level remains the same (22 bits need 5-6 days to catch the correct match by random fuzzing). The NWK layer also processes the incoming packets and transports the NWK payload to the upper layer (as in Figure 1). There is no standard minimum NWK packet length (including headers, payload and integrity code) in the specification [8]. To mitigate the Truncated packets attack (Attack 5 in 7.2), we can standardize the minimum NWK packet length threshold. Truncated packets, as incomplete packets, can be discarded without network layer check.

APS layer. The APS layer provides control services (as in Section 2.1). The APS command 'verify key' (command ID `0F`) provides the source address (MAC), key type and key, which can be utilized as the verification message. To mitigate the Disconnection attack (Attack 2 in 7.2), the controller can check the device's information periodically. The APS layer also provides APS encryption (as in Figure 1). To mitigate the Key leakage attack (Attack 3 in 7.2), we propose to add a separate encryption key and put the key management

```
Frame Information: (47 bytes)
MAC Header: (9 bytes)
MAC Payload: (36 bytes)
  NWK Header: (16 bytes)
    Frame Control: 0x1209
      Destination Address: 0xFFFC
      Source Address: 0x0000
      Radius: 0x01
      Sequence Number: 65
      Source IEEE Address: 00:15:5F:00:41:25:DF:4A
  NWK Aux Header: (14 bytes)
    Network Security Control: 0x28
      NWK Frame Counter: 63014
      Source Address: 00:15:5F:00:41:25:DF:4A
      NWK Key Sequence Number: 0
  NWK Payload: 0x6008
      NWK Command ID: [0x08] Link Status
    Link Status Options: 0x60
  NWK MIC: 0xC5B5D3FD
MAC Footer: 0x0E9B
- - - - - - - - - - - - - - - - - - -
Frame Information: (41 bytes)
MAC Header: (9 bytes)
MAC Payload: (30 bytes)
  NWK Header: (16 bytes)
    Frame Control: 0x1209
      Destination Address: 0x61F3
      Source Address: 0x0000
      Radius: 0x01
      Sequence Number: 111
      Source IEEE Address: 00:15:5F:00:41:25:DF:4A
  NWK Aux Header: (14 bytes)
    Network Security Control: 0x28
      NWK Frame Counter: 4294967295
      Source Address: 00:15:5F:00:41:25:DF:4A
      NWK Key Sequence Number: 0
MAC Footer: 0x0F8E
```

Figure 9: Attack 5 Comparison between regular and truncated packets. The top example shows a regular packet, and the bottom example shows a truncated case (the truncation comparison is highlighted with the black box).

overhead on the controller side (which has more computational capability).

ZCL layer. The ZCL layer provides data services (cluster functionality as in Section 2.1). The data services include in the clusters, commands and attributes. To mitigate the Communication Interruption attack (Attack 3 in 7.2), Zigbee can standard the packet processing with undefined clusters, commands or attributes to avoid unpredictable results.

Limitation. Though our developed framework is effective to discover Zigbee vulnerabilities, our approach has overhead and scalability limitations due to analyzing physical hardware and devices. The experiments use a real development module to execute code, and the packet transmission rate is restricted by the antenna capability. A future research direction is to develop emulation to reduce the experiment overhead. Software emulation of Zigbee stack and packets needs less dependency on hardware, and a combination approach with both software emulation and hardware testing has the potential to improve performance and address the limitations.

9 RELATED WORK

Zigbee security. Previous studies focused on Zigbee security in limited attack scenarios. One main type is to attack Zigbee commissioning phase, including touchlink vulnerabilities [52], unauthorized commissioning [1, 45, 61], human action leakage [51],

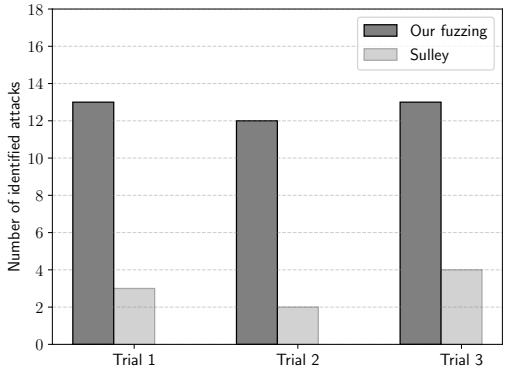

Figure 10: Comparison over multiple trials.

association request attack [58], and insecure rejoining [64]. However, the commissioning phase has a short attack time window, which requires human interactions to enable. Another type is to learn security keys of the Zigbee system, including breaking into Zigbee encryption [60], command functions in Killerbee [57, 71, 77], local flash memory [32], key revocation and key sniffing [26, 81]. Different from existing work, the attacks in our study do not know security keys a priori. The third type is attack launched in the target Zigbee network, including spoofing device fingerprinting [66], spoofing system processing [62], and attacking router buffer [56]. Others studied attacks targeting lower layers, such as physical layer attacks [13, 17, 70, 72], MAC layer jamming attacks [5], and power consumption [6, 16]. In contrast, our work reveals Zigbee attacks with unprivileged conditions, i.e., not limited to the commissioning phase and not knowing security keys a priori, and systematically examines upper layer information.

IoT security. With the thriving of IoT devices, research on IoT security has drawn more attention. Vulnerabilities in connection phase exploit authentication [30, 67, 74], application connection [9] and commissioning in 802.11 and LTE [15, 19, 42]. Goodspeed et al. [33] crafted packets-in-packets in the physical layer to inject radio packets; On the other hand, we developed methods to manipulate higher layer fields for Zigbee. Compromised IoT devices can be exploited to launch physical-world attacks [36, 65]. Krivtsova et al. [44] used broadcast storm attack to congest the transmission channel. Antonakakis et al. [11] studied that Mirai botnet infected IoT devices' ecosystem. In contrast, our work focuses on analyzing the security of packet communication of Zigbee, widely used by modern IoT devices.

Protocol fuzzing. Existing Zigbee fuzzing analysis was merely on the MAC layer [48, 80]. Other traditional fuzzing [10, 23, 43, 46, 53, 59] targeted specific protocols and required normal communication to perform analysis, not applicable to our analysis scenarios where the device is outside of the target Zigbee network. Miller et al. [50] introduced classic fuzzing, and our work leveraged similar insight and developed the semantic-aware method for Zigbee. Existing semantic fuzzing considered simple cases, such as payload length fuzzing (long payload, zero or no payload) [14, 27, 47, 68, 73], simple character manipulation [3], or special values enumeration [40]. On the other hand, our Zigbee semantic-aware fuzzing is to analyze

packet structure and parameter relationships in upper layers, to explore Zigbee vulnerabilities more efficiently.

10 CONCLUSION

In this paper we systematically analyze attacks targeting Zigbee. We find new attacks where adversaries can launch attacks from outside of the target network and do not need any encryption keys. We further develop an analysis framework that provides flexible modification of Zigbee network information and fine-granularity control of the packet details from the MAC layer. To achieve efficient analysis, we design semantic-aware fuzzing to explore packets that are more likely to produce meaningful results. We identified five attacks that can cause malfunctional states (including disconnection and security information leakage) and built proof of concepts on 10 industry Zigbee products and systems. Our work highlights previously unknown Zigbee security problems and provides mitigation suggestions. In the future, the Zigbee protocol will be widely used in 5G cellular network as Massive IoT [76]. The vulnerabilities of one Zigbee device can affect other IoT devices in the network.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments to improve the paper. This work was supported by the Office of Naval Research under ONR award number N00014-20-1-2738. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] 2020. CVE-2019-15914. https://github.com/chengcheng227/CVE-POC/blob/master/CVE-2019-15914_1.md. (2020).
[2] 9to5Mac. 2018. HomeKit Devices Getting More Affordable as Lenovo Announces Smart Home Essentials Line. https://9to5mac.com/2018/08/31/cheap-homekit-bulbs-switches-camera/. (2018).
[3] Humberto J. Abdelnur, Radu State, and Olivier Festor. 2007. KiF: A Stateful SIP Fuzzer. In *1st International Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm)*.
[4] Raafat Aburukba, A. R. Al-Ali, Nourhan Kandil, and Diala AbuDamis. 2016. Configurable ZigBee-based Control System for People with Multiple Disabilities in Smart Homes. In *International Conference on Industrial Informatics and Computer Systems(CIICS)*.
[5] Dimitrios-Georgios Akestoridis, Madhumitha Harishankar, Michael Weber, and Patrick Tague. 2020. Zigator: Analyzing the Security of Zigbee-Enabled Smart Homes. In *13th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*.
[6] Dimitrios-Georgios Akestoridis and Patrick Tague. 2021. HiveGuard: A Network Security Monitoring Architecture for Zigbee Networks. In *2021 IEEE Conference on Communications and Network Security (CNS)*.
[7] Ahmad Alagil, Meshari Alotaibi, and Yao Liu. 2016. Randomized Positioning DSSS for Anti-Jamming Wireless Communications. In *International Conference on Computing, Networking and Communications (ICNC)*.
[8] Connectivity Standards Alliance. 2017. *Zigbee Specification.* ZigBee Document 05-3474-22.
[9] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *40th IEEE Symposium on Security and Privacy (S&P)*.
[10] Pedram Amini and Aaron Portnoy. 2007. Sulley: Fuzzing Framework. http://www.fuzzing.org/wp-content/SulleyManual.pdf. (2007).
[11] Manos Antonakakis, Tim April, Michael Bailey, Matt Berhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Daimian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *26th USENIX Security Symposium (USENIX Security)*.
[12] Apple. 2022. HomeKit - Apple Developer. https://developer.apple.com/homekit/. (2022).
[13] Anshuman Biswas, Abdulaziz Alkhalid, Thomas Kunz, and Chung-Horng Lung. 2012. A Lightweight Defence Against the Packet in Packet Attack in ZigBee Networks. In *2012 IFIP Wireless Days (WD)*.
[14] Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. 2019. Geneva: Evolving Censorship Evasion Strategies. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
[15] Laurent Butti and Julien Tinnès. 2007. Discovering and Exploiting 802.11 Wireless Driver Vulnerabilities. In *Information and Communications Technology Security Symposium (SSTIC)*.
[16] Xianghui Cao, Devu Manikantan Shila, Yu Cheng, Zequ Yang, Yang Zhou, and Jiming Chen. 2016. Ghost-in-ZigBee: Energy Depletion Attack on ZigBee-Based Wireless Networks. *IEEE Internet of Things Journal* 3, 5 (2016), 816–829.
[17] Romain Cayre, Florent Galtier, Guillaume Auriol, Vincent Nicomette, Mohamed Kaâniche, and Géraldine Marconato. 2021. WazaBee: Attacking Zigbee Networks by Diverting Bluetooth Low Energy Chips. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.
[18] Jiming Chen, Qing Yu, Bo Chai, Youxian Sun, Yanfei Fan, and Xuemin Shen. 2015. Dynamic Channel Assignment for Wireless Sensor Networks: A Regret Matching Based Approach. In *IEEE Transactions on Parallel and Distributed Systems (TPDS)*.
[19] Weiteng Chen and Zhiyun Qian. 2018. Off-Path TCP Exploit: How Wireless Routers Can Jeopardize Your Secrets. In *27th USENIX Security Symposium (USENIX Security)*.
[20] Tao Chi and Haowei Yan. 2017. A Dynamic Channel Assignment for Coexistence of ZigBee/WiFi. In *2017 International Conference on Information Technology and Intelligent Manufacturing (ITIM)*.
[21] Connectivity Standards Alliance 2022. Connectivity Standards Alliance Products. https://csa-iot.org/csa-iot_products/. (2022).
[22] Cristiano Andre da Costa, Cristian F. Pasluosta, Bjorn Eskofier, Denise Bandeira da Silva, and Rodrigo da Rosa Righi. 2018. Internet of Health Things: Toward intelligent vital signs monitoring in hospital wards. *Artificial Intelligence In Medicine* (2018), 61–69.
[23] Joeri de Ruiter and Erik Poll. 2015. Protocol State Fuzzing of TLS Implementation. In *24th USENIX Security Symposium (USENIX Security)*.
[24] George Demiris and Brian K. Hensel. 2008. Technologies for an Aging Society: A Systematic Review of 'Smart Home' Applications. *IMIA Yearbook of Medical Informatics 2008* (2008), 33–40.
[25] Ganesh Devarajan. 2007. Unraveling SCADA Protocols: Using Sulley Fuzzer. In *DEF CON 15 Hacking Conference*.
[26] Gianluca Dini and Marco Tiloca. 2010. Considerations on Security in Zigbee Networks. In *2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*.
[27] Karel Domin, Eduard Marin, and Iraklis Symeonidis. 2016. Security Analysis of the Drone Communication Protocol: Fuzzing the MAVLink Protocol. https://orbilu.uni.lu/bitstream/10993/37613/1/article-2667.pdf. (2016).
[28] ElectronicDesign. 2004. ZigBee Special Report: The ZigBee Buzz Is Growing. https://www.electronicdesign.com/energy/zigbee-special-report-zigbee-buzz-growing. (2004).
[29] Altaf Engineer, Esther M. Sternberg, and Bijan Najafi. 2018. Designing Interiors to Mitigate Physical and Cognitive Deficits Related to Aging and to Promote Longevity in Older Adults: A Review. *Gerontology* (2018), 612–622.
[30] Earlence Fernandes, Amir Rahmati, Jaeyeon Jung, and Atul Prakash. 2018. Decentralized Action Integrity for Trigger-Action IoT Platforms. In *Network and Distributed Systems Security Symposium (NDSS)*.
[31] Arthur Gatouillat, Youakim Badr, Bertrand Massot, and Ervin Sejdic. 2018. Internet of Medical Things: A Review of Recent Contributions Dealing With Cyber-Physical Systems in Medicine. *Internet of Things Journal* (2018), 3810–3822.
[32] Travis Goodspeed. 2009. Extracting Keys from Second Generation Zigbee Chips. In *Black Hat USA*. https://www.blackhat.com/presentations/bh-usa-09/GOODSPEED/BHUSA09-Goodspeed-ZigbeeChips-PAPER.pdf
[33] Travis Goodspeed, Sergey Bratus, Ricky Melgares, Rebecca Shapiro, and Ryan Speers. 2011. Packets in Packets: Orson Welles' In-Band Signaling Attacks for Modern Radios. In *Proc. 5th USENIX Workshop on Offensive Technologies (WOOT)*. San Francisco, CA.
[34] Tom's Guide. 2022. Best Smart Home Hubs. https://www.tomsguide.com/us/best-smart-home-hubs,review-3200.html. (2022).
[35] Jan Haase, Mahmoud Alahmad, Hiroaki Nishi, Joern Ploennigs, and Kim Fung Tsang. 2016. The IOT Mediated Built Environment: A Brief Survey. In *IEEE 14th International Conference on Industrial Informatics (INDIN)*.
[36] Bing Huang, Alvaro A. Cardenas, and Ross Baldick. 2019. Not Everything is Dark and Gloomy: Power Grid Protections Against IoT Demand Attacks. In *Proc. 28th USENIX Security Symposium (USENIX Security)*. Santa Clara, CA.
[37] InformationWeek. 2020. IoT Revenue Projected to Reach $3 Trillion by 2025. https://www.informationweek.com/it-life/iot-market-could-top-3-trillion-by-2025-report-finds. (2020).
[38] Texas Instruments. 2022. CC2531 USB Evaluation Module Kit. https://www.ti.com/tool/CC2531EMK. (2022).
[39] Texas Instruments. 2022. CC2538 Development Kit. https://www.ti.com/tool/CC2538DK. (2022).

[40] Samuel Jero, Xiangyu Bu, Cristina Nita Rotaru, Hamed Okhravi, Richard Skowyra, and Sonia Fahmy. 2017. BEADS: Automated Attack Discovery in OpenFlow-Based SDN Systems. In *Research in Attacks, Intrusions, and Defenses (RAID)*.

[41] Won Min Kang, Seo Yeon Moon, and Jong Hyuk Park. 2017. An enhanced security framework for home appliances in smart home. *Human-centric Computing and Information Sciences* (2017), 7(6).

[42] Hongil Kim, Jiho Lee, Eunkyu Lee, and Yongdae Kim. 2019. Touching the Untouchables: Dynamic Security Analysis of the LTE Control Plane. In *40th IEEE Symposium on Security and Privacy (S&P)*.

[43] Takahisa Kitagawa, Miyuki Hanaoka, and Kenji Kono. 2010. AspFuzz: A State-aware Protocol Fuzzer based on Application-layer Protocols. In *The IEEE symposium on Computers and Communications (ISCC)*.

[44] Irina Krivtsova, Ilya Lebedev, Mikhail Sukhoparov, Nurzhan Bazhayev, Igor Zikratov, Aleksandr Ometov, Sergey Andreev, Pavel Masek, Radek Fujidiak, and Jiri Hosek. 2016. Implementing a Broadcast Storm Attack on a Mission-Critical Wireless Sensor Network. In *Wired/Wireless Internet Communications (WWIC)*.

[45] Gunhee Lee, Jaesung Lim, Dong kyoo Kim, SungHyun Yang, and MyungHyun Yoon. 2008. An Approach to Mitigating Sybil Attack in Wireless Networks Using ZigBee. In *10th International Conference on Advanced Communication Technology (ICACT)*.

[46] Seungsoo Lee, Changhoon Yoon, Chanhee Lee, Seungwon Shin, Vinod Yegneswaran, and Phillip Porras. 2017. DELTA: A Security Assessment Framework for Software-Defined Networks. In *Network and Distributed Systems Security Symposium (NDSS)*.

[47] Antti Levomäki, Olli-Pekka Niemi, and Christian Jalio. 2017. Automatic Discovery of Evasion Vulnerabilities Using Targeted Protocol Fuzzing. In *Black Hat 2017*. https://www.blackhat.com/docs/eu-17/materials/eu-17-Levomaki-Automatic-Discovery-Of-Evasion-Vulnerabilities-Using-Targeted-Protocol-Fuzzing-wp.pdf

[48] Hui Li, Weishi Zhang, Weifu Zhou, and Bo Su. 2014. A Novel Vulnerability Detection Method for ZigBee MAC Layer. In *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing (DASC)*.

[49] Spencer Michaels, Kemal Akkaya, and A. Selcuk Uluagac. 2016. Inducing Data Loss in Zigbee Networks via Join/Association Handshake Spoofing. In *IEEE Conference on Communications and Network Security (CNS)*.

[50] Barton P. Miller, Lars Fredriksen, and Bryan So. 1990. An Empirical Study of the Reliability of UNIX Utilities. *Communications of the ACM* (1990), 32–44.

[51] MIT. 2017. Security Analysis of Zigbee. https://courses.csail.mit.edu/6.857/2017/project/17.pdf. (2017).

[52] Philipp Morgner, Stephan Maejat, Zinaida Benenson, Christian Muller, and Frederik Armknecht. 2017. Insecure to the Touch: Attacking ZigBee 3.0 via Touchlink Commissioning. In *10th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*.

[53] Mozilla. 2017. Peach Security Fuzzing. https://wiki.mozilla.org/Security/Fuzzing/Peach. (2017).

[54] Aristides Mpitziopoulos, Damianos Gavalas, Charalampos Konstantopoulos, and Grammati Pantziou. 2009. A Survey on Jamming Attacks and Countermeasures in WSNs. In *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, Vol. 11. 42–56.

[55] Yahoo! News. 2016. Why IoT, Big Data & Smart Farming Are the Future of Agriculture. https://uk.news.yahoo.com/why-iot-big-data-smart-192155797.html. (2016).

[56] Satoshi Okada, Daisuke Miyamoto, Yuji Sekiya, and Hiroshi Nakamura. 2021. New LDoS Attack in Zigbee Network and Its Possible Countermeasures. In *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*.

[57] Olayemi Olawumi, Keijo Haataja, Mikko Asikainen, and Niko Vidgren Pekka Toivanen. 2014. Three Practical Attacks Against ZigBee Security: Attack Scenario Definitions, Practical Experiments, Countermeasures, and Lessons Learned. In *14th International Conference on Hybrid Intelligent System (HIS)*.

[58] Taifeng Pan. 2021. ZigBee Wireless Network Attack and Detection. In *Advances in Artificial Intelligence and Security*, Xingming Sun, Xiaorui Zhang, Zhihua Xia, and Elisa Bertino (Eds.). Springer International Publishing, Cham, 391–403.

[59] Van Thuan Pham, Marcel Böhme, and Abhik Roychoudhury. 2020. AFLNET: A Greybox Fuzzer for Network Protocols. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*.

[60] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. 2017. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. In *Proc. IEEE Symposium on Security and Privacy (S&P)*.

[61] Kudelski Security. 2017. ZigBee Security: Basics. https://research.kudelskisecurity.com/2017/11/21/zigbee-security-basics-part-3/. (2017).

[62] Narmeen Shafqat, Daniel J. Dubois, David R. Choffnes, Aaron Schulman, Dinesh Bharadia, and Aanjhan Ranganathan. 2021. ZLeaks: Passive Inference Attacks on Zigbee based Smart Homes. *CoRR* abs/2107.10830 (2021). arXiv:2107.10830

[63] Haykin Simon and Michael Moher. 2009. *Communication systems*. Wiley.

[64] Smartthings. 2016. Disable ZigBee Insecure Rejoin. https://community.smartthings.com/t/disable-zigbee-insecure-rejoin/40809. (2016).

[65] Saleh Soltan and Prateek Mittal. 2018. BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid. In *27th USENIX Security Symposium (USENIX Security)*.

[66] Grace Hanusha Talakala and Jyotsna Bapat. 2021. Detecting Spoofing Attacks in Zigbee using Device Fingerprinting. In *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*.

[67] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, XianZheng Guo, and Patrick Tague. 2017. SmartAuth: User-Centered Authorization for the Internet of Things. In *26th USENIX Security Symposium (USENIX Security)*.

[68] Petar Tsankov, Mohammad Torabi Dashti, and David Basin. 2012. SECFUZZ: Fuzz-testing Security Protocols. In *7th International Workshop on Automation of Software Test (AST)*.

[69] Ubilogix. 2022. Ubiqua Protocol Analyzer. https://www.ubilogix.com/ubiqua/. (2022).

[70] Ivan Vaccari, Enrico Cambiaso, and Maurizio Aiello. 2017. Remotely Exploiting AT Command Attacks on ZigBee Networks. *Security and Communication Networks* (2017), 9.

[71] Niko Vidgren, Keijo Haataja, Jose Luis Patino-Andres, Juan Jose Ramirez-Sanchis, and Pekka Toivanen. 2013. Security Threats in Zigbee-Enabled Systems: Vunerability Evaluation, Practical Experiments, Countermeasures, and Lessons Learned. In *2013 46th Hawaii International Conference on System Sciences (HICSS)*.

[72] K. Vivek Raj, P. Dinesha, and S. I. Arpitha Shankar. 2022. Security Considerations in the Design of IEEE 802.15.4 Transceiver: A Review. In *Cyber Security and Digital Forensics*, Kavita Khanna, Vania Vieira Estrela, and Joel José Puga Coelho Rodrigues (Eds.). Springer Singapore, Singapore, 213–229.

[73] Artemios G. Voyiatzis, Konstantinos Katsigiannis, and Stavros Koubias. 2015. A Modbus/TCP Fuzzer for Testing Internetworked Industrial Systems. In *IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*.

[74] Weicheng Wang, Fabrizio Cicala, Syed Rafiul Hussain, Elisa Bertino, and Ninghui Li. 2020. Analyzing the Attack Landscape of Zigbee-Enabled IoT Systems and Reinstating Users' Privacy. In *13th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*.

[75] WIOMAX. 2016. Key Applications of the Smart IoT to Transform Transportation. http://www.wiomax.com/what-can-the-smart-iot-transform-transportation-and-smart-cities/. (2016).

[76] Network World. 2020. What 5G Promises for IoT. https://www.networkworld.com/article/3584385/what-5g-brings-to-iot-today-and-tomorrow.html. (2020).

[77] Joshua Wright. 2009. KillerBee: Practical ZigBee Exploitation Framework. http://www.willhackforsushi.com/presentations/toorcon11-wright.pdf. (2009).

[78] Xie Xiao-Feng and Wang Zun-Jing. 2017. Integrated in-Vehicle Decision Support System for Driving at Signalized Intersections: A Prototype of Smart IoT in Transportation. In *Transportation Research Board (TRB) Annual Meeting*.

[79] Chen Yang, Weiming Shen, and Xianbin Wang. 2018. The Internet of Things in Manufacturing: Key Issues and Potential Applications. *IEEE Systems, Man, and Cybernetics Magazine* (2018), 6–15.

[80] Jingling Zhao, Shilei Chen, Shurui Liang, Baojiang Cui, and Xiaolong Song. 2013. RFSM-Fuzzing a Smart Fuzzing Algorithm Based on Regression FSM. In *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*.

[81] Tobias Zillner. 2015. ZigBee Exploited: The Good the Bad and the Ugly. In *Black Hat USA*. https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly-wp.pdf