Evaluating the Security Vulnerabilities of ZigBee Motion Detectors in Wireless Sensor Networks: A Study on Jamming, Interception, and Behavioral Manipulation

Diogo Marques, Henk Netten, Muhammad Mansour

October 27, 2024

Abstract

The increasing adoption of ZigBee-based Internet of Things (IoT) devices has revolutionized automation and connectivity across residential, commercial, and industrial settings. Despite their advantages, such as low power consumption and scalability, the integration of these devices into critical systems, including home security networks, exposes them to a variety of cyber threats. This study focuses on the security evaluation of ZigBee motion detectors, assessing their vulnerabilities when deployed in high-security applications. Specifically, it investigates the impact of attacks such as jamming, interception, and behavioral manipulation on the integrity and reliability of these devices. Through controlled traffic analysis and experimentation, the research identifies critical weaknesses within the ZigBee protocol, offering insights into how these vulnerabilities can be exploited in real-world scenarios. The study's findings emphasize the importance of robust defense strategies, including improved encryption practices and resilient network configurations, to mitigate the risks associated with these increasingly ubiquitous IoT systems. By enhancing the understanding of ZigBee security challenges, this research contributes to developing more secure IoT deployments, particularly for environments where the stakes are high.

Contents

1 Introduction

The proliferation of Internet of Things (IoT) devices, especially those employing ZigBee-based protocols for sensors and controllers, has transformed modern living by enhancing automation and efficiency across smart homes, healthcare, and industrial settings ([1]; [2]). ZigBee's popularity stems from its design as a low-cost, low-power, and highly interoperable communication protocol, ideal for connecting various devices within IoT networks. However, this very design prioritization has introduced significant security risks, as the protocol often lacks robust encryption mechanisms and strong access controls ([3]). This vulnerability becomes particularly concerning in high-security environments where ZigBee devices are deployed for critical functions like motion detection and surveillance.

Existing research highlights the susceptibility of ZigBee networks to a variety of attacks, including replay, man-in-the-middle (MITM), and device spoofing attacks, which exploit weaknesses in the IEEE 802.15.4 standard upon which ZigBee is built ([2]). These vulnerabilities are exacerbated in environments where ZigBee devices communicate via mesh networks, often leading to compromised node communications and unauthorized access ([3]). Therefore, this study's primary research question is: How can ZigBee-based motion detectors be safely utilized in high-security applications, such as break-in detection? To investigate this, the following sub-questions are posed:

- What are the vulnerabilities in the ZigBee-based IoT network?
- How do we defend against attacks that use these vulnerabilities?

This study involves sniffing ZigBee traffic within a controlled laboratory setting to identify and evaluate these vulnerabilities. Adhering to ethical standards, the research ensured that no unauthorized ZigBee systems within range were intruded upon; only packets with keys exclusive to our testing environment were decrypted. This precaution aligns with best practices to minimize the risk of unauthorized data access and interference according to OS3 principles.

The identified vulnerabilities will be disclosed in accordance with OS3 guidelines, ensuring a responsible approach that balances the need for transparency with the risks associated with publicizing potential security flaws ([1]). This research adds to the growing body of literature addressing IoT security, emphasizing the necessity for enhanced protective measures in critical systems operating within an increasingly interconnected digital landscape.

2 Literature Review / Related Work

Discuss prior research on ZigBee network security, its encryption mechanisms, vulnerabilities like jamming and replay attacks, and key management issues. Identify research gaps that your study aims to address.

1. **Research on the Security of ZigBee Wireless Sensor Network**: Research by Tang (2022) analyzes various aspects of ZigBee network technology, including its security structure and encryption algorithms. It demonstrates the implementation of symmetric keys and the security architecture of the protocol stack. Additionally, it investigates the security of the ZigBee on different layers. It provides a deeper insight into the structural and security aspects of ZigBee networks. [4]

2. **ZigBee Security Vulnerabilities: Exploration and Evaluation**: This work discusses common vulnerabilities such as jamming attacks and replay attacks in ZigBee networks. It explores how attackers can exploit ZigBee's relatively weak encryption and key management protocols, which are crucial considerations when evaluating whether ZigBee motion detectors are secure enough for break-in detection.[5]

3. **MIT's Security Analysis of ZigBee Protocol**: This paper provides a comprehensive analysis of ZigBee's security, covering encryption mechanisms, key management, and various attack vectors, including jamming and eavesdropping. It gives valuable context for understanding how an attacker might intercept or manipulate the data sent between a ZigBee motion detector and a controller. [6]

4. **Advanced Analysis of ZigBee-based IoT System Security**: Another relevant study offers practical insights into attacks like man-in-the-middle, jamming, and replay attacks, showing how easily ZigBee networks can be compromised without proper countermeasures. The findings align with your aim to evaluate the motion sensor's reliability in a security system. [7]

5. **ZigBee Jamming**: This journal describes a low-tech way of disabling a ZigBee sensor through jamming. They used SDR to perform multiple types of jamming attacks. It also goes into why wireless devices are generally not suitable for physical security applications. [8]

3 Methodology / Approach and Methods

3.1 Network Setup and Configuration

For our research subject we chose the Sonoff human presence sensor (snzb-06p). We chose this Device for its affordability. At the time of writing it costs only €10.18 on AliExpress.com making it accessible for low end security systems. To setup the presence sensor with a network we used a Philips Hue Bridge and a TP-Link Wireless Router TL-WR841N. We used a mobile phone with the Tuya Smart app to pair the sensor to the bridge. We used a HackRF One to sniff the ZigBee traffic and perform attacks on the system. Finally, we also used a ZigBee button (Sonoff SNZB-01P). The button was more convenient for triggering an observation because the presence sensor would always give a positive signal when we were in the room. We also used it to test adding a device to the network. The full setup is shown in Figure 1.

To control the SDR and analyze the ZigBee traffic we used a laptop running Linux and GNU Radio 3.10. GNU Radio is a flow graph based tool for controlling SDR devices.

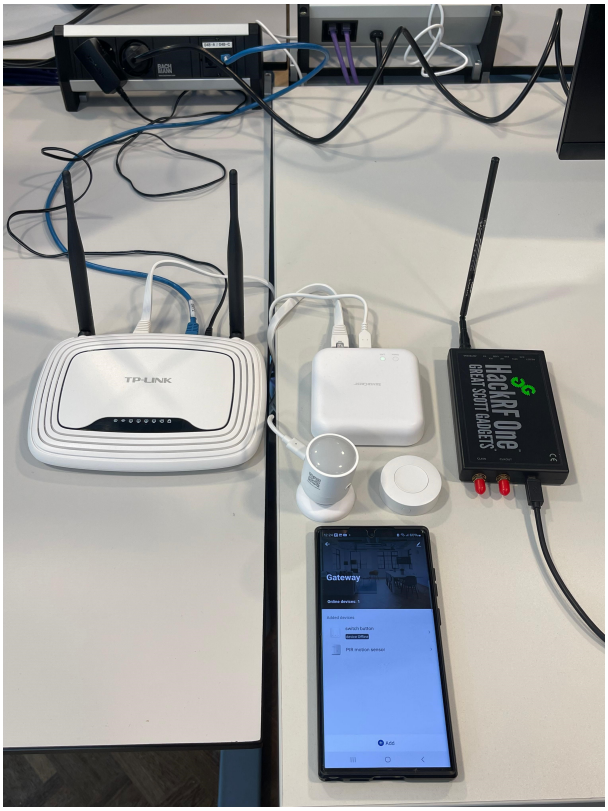

Figure 1: Testing environment including wifi router, ZigBee bridge, presence sensor, ZigBee button, software defined radio and mobile phone

The flow graph we used contained blocks from the grc-ieee802154 GitHub project [9]. We also used the RFJamming-FMRadio-SDR GitHub project to generate a jamming signal [10]. To analyze the traffic we captured with the HackRF we used Wireshark.

3.2 Packet Capture and Traffic Analysis

To setup the sniffing environment we first turned on the HackRF to start listening. Then we turned on the router, bridge and presence sensor. Then we set the bridge to pairing mode through the app and we set the sensor to pairing mode by pressing its button. In the mean time the HackRF was capturing the traffic and stored it to a pcap file to later be analyzed through Wireshark. Figure 2 shows how the different devices are connected.

To capture what happens when the sensor goes from detecting no presence to detecting a presence we would leave the room, wait until the app indicates the sensor senses no presence and then enter the room again. This would usually take between 1-5 minutes.

3.3 Denial of Service and Jamming

We tested two methods to disable the system using jamming. The first is to jam the communication between the sensor and the bridge so reports of any detected presence

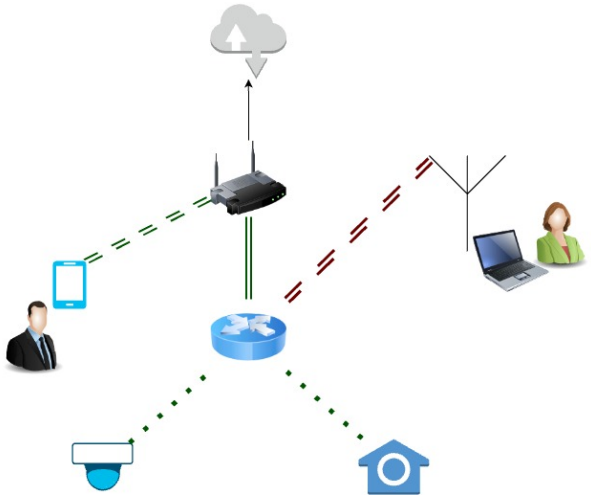

Figure 2: Testing environment for sniffing traffic between the sensor and the bridge.

are lost. The second is to jam the detection by the sensor. The sensor uses 5.8GHz pulses to detect human presence. We attempted to jam this frequency in the hopes this disables the sensor.

For jamming the communication between the sensor and the bridge we varied the distance between the sensor and the bridge and between the HackRF and the bridge. When the sensor is farther from the bridge the signal received by the bridge is weaker which makes the jamming less effective. Similarly, when the jammer is closer to the bridge the jamming signal received by the bridge is stronger making the jamming more effective. In a real security system it is likely the attacker does not have direct access to the bridge so it makes sense to test how effective an attack still is when the attackers jammer is further away. In a real security system the sensor will likely also be in different rooms from the bridge. In our first experiment we put all three devices within 50 cm of each other. Then we moved the sensor to the next room with the jammer still next to the bridge. Finally, we moved the jammer to the other room with the sensor.

For jamming the detection by the sensor we positioned the HackRF directly in front of the sensor and set it to jam the 5.8 GHz frequency with Gaussian noise. We then left the room to see if the sensor would notice we left. After 5 minutes we would re-enter to see if the sensor noticed us coming back.

The jamming signal was Gaussian noise centered around 2.475 GHz with a 20 MHz bandwidth and 15 dBm amplitude. The 2.475 GHz frequency corresponds with ZigBee channel 25. We found this channel by iterating over all ZigBee channels, from 11 to 26, and listening for traffic using the HackRF in receiving mode. We use a bandwith of 20 MHz because this is the bandwidth of ZigBee channels. We set the amplitude to 15 dBm because this is the maximum amplitude the HackRF supports for this frequency range [11].

4 Results / Analysis

4.1 Testing the normal operation of the ZigBee network

In this project, a Zigbee presence sensor was connected to a ZigBee coordinator, which was subsequently linked to an Android application via a WiFi router connected to the Internet. Additionally, a ZigBee sniffer (HackRF) was utilized for monitoring purposes. The initial scenario involved testing the system by placing the Zigbee devices in pairing mode and attempting to visualize their connections within the applications. Despite using applications recommended by the coordinator and sensor companies, both failed to detect the devices. However, the use of a third-party application resolved this issue, enabling the successful addition of the sensor and coordinator. The application now displays the sensor's status, indicating whether it is online (connected to the coordinator) or offline (not connected). When online, the sensor has two states: "On," signifying the detection of presence within the space (remaining in this state for 2 to 3 minutes even after the space is vacated), and "Off," indicating no presence detected.

4.2 Intercepting ZigBee packets

In the subsequent scenario, we utilized a Zigbee sniffer (HackRF) to intercept the Zigbee traffic between the sensor and the coordinator. GNU Radio was employed as the processing tool to handle the received packets, followed by Wireshark to present the captured data. The flow graph (Figure 3) illustrates the modules used in GNU Radio.

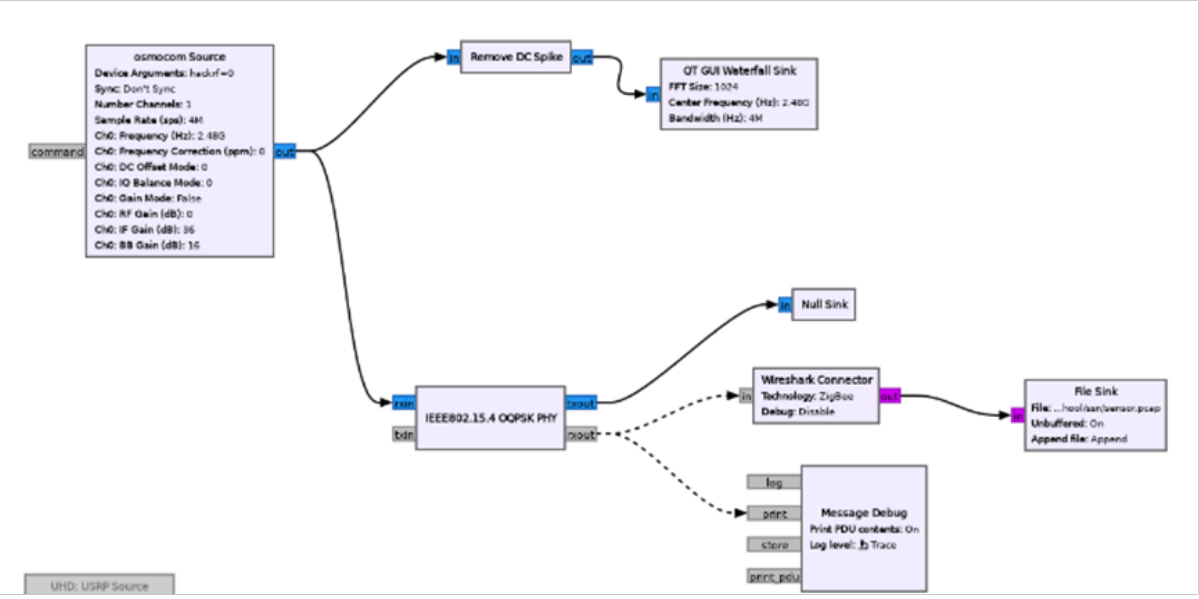

Figure 3: inception blocks flow diagram

We used the Osmocom Source module to obtain samples from the sniffer. The IEEE 802-15-4 OQPSK PHY block was utilized to demodulate the Zigbee messages, which use the OQPSK (Offset Quadrature Phase Shift Keying) modulation technique. For debugging

purposes, we added a PDU Message Debug block to display packets on the terminal. Additionally, we incorporated the Wireshark Connector and File Sink to generate a pcap file containing all captured packets.[9]

To enhance the signal quality, we used the Remove DC Spike block to eliminate the DC component from the captured signal. Moreover, the QT GUI Waterfall Sink was employed to visualize the captured data over time and frequency. The configuration parameters are detailed in Table 1 below:

Parameter	**Value**	**Description**
Device Argument	hackrf	Device to capture the signal
Central Frequency	2.48 GHz	Frequency range used by our Zigbee devices
Channel Bandwidth	5 MHz	Difference between Zigbee channels
Sample Rate	4 M	Number of samples taken per second
Channels	11 to 26	Manually changed from 11 to 26 until data is captured

Table 1: Configuration Parameters

Zigbee packets are encrypted using AES-128, ensuring that packet contents cannot be read without the appropriate key. Initially, all Zigbee devices are preconfigured with a default global trust center link key (5A 69 67 42 65 65 41 6C 6C 69 61 6E 63 65 30 39), which is used when the sensor begins communication with the coordinator. The devices use this default key until they establish a new link key, which is sent by the coordinator through a Network Key message. Subsequent communications are encrypted using this new link key. A significant aspect of this process is the need to intercept messages during the initial handshake period. During this phase, the default global trust center link key is used to decrypt the Network Key message. Once this message is decrypted, the link key is obtained and can be configured in Wireshark to decrypt subsequent messages. Notably, the same link key is used whenever the sensor reconnects to the same coordinator, meaning it only needs to be acquired once for continuous use. The intercepted link key is depicted in Figure 4 below, illustrating the critical role it plays in decrypting ongoing communications within the Zigbee network.

4.3 Obtaining the link key by pairing another ZigBee device

However, if you attempt to intercept the messages between already paired ZigBee devices, you will not be able to obtain the link key. To address this issue, we tried pairing another ZigBee device with the coordinator, forcing it to initiate the handshake again. This allowed us to investigate if the same link key would be used, which could then be used to decrypt the first device's messages. The second device we attempted to pair was a switch for controlling a lamp. Unfortunately, this approach was not successful because the user must manually enter the app on their mobile phone and accept the pairing to add a new device to the network. This requirement provides an additional layer of security, preventing unauthorized access to the link key and ensuring the integrity of the communication within the ZigBee network.

Furthermore, attempting to connect the coordinator or sensor to another mobile phone

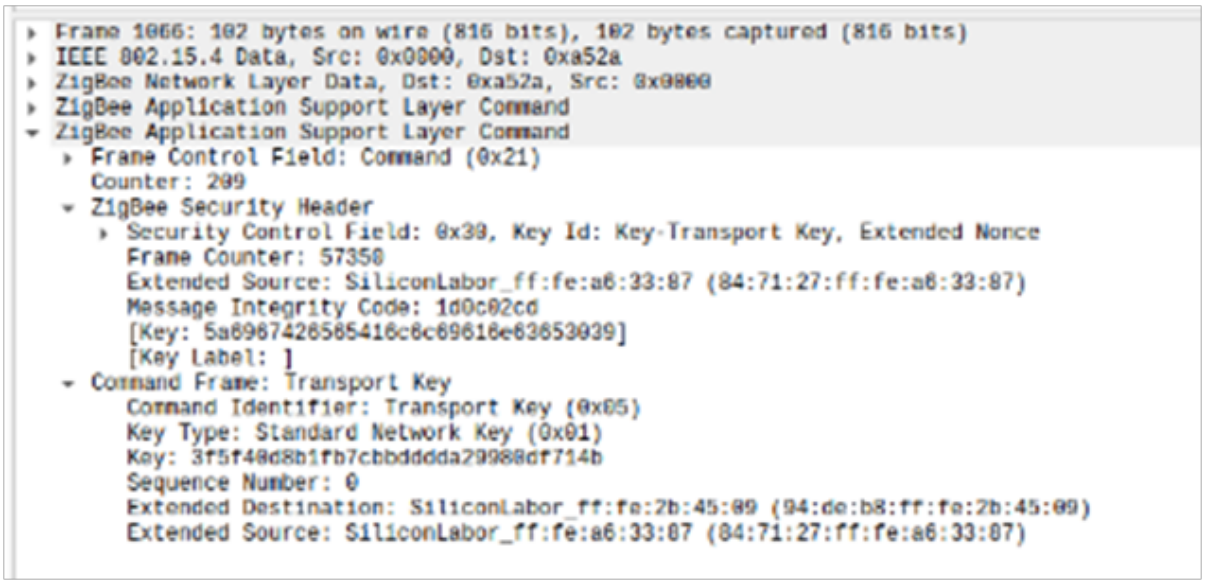

Figure 4: Link Key

requires placing the device into pairing mode by pressing the reset button for approximately five seconds, necessitating physical access to the ZigBee device. This action triggers the sensor to record the attempt, which is then communicated to the coordinator. Consequently, this attempt will appear in the application's log history, even if the device is later paired with another mobile phone. This requirement prevents unauthorized remote pairing attempts and ensures that only individuals with physical access to the device can initiate the pairing process.

4.4 Conducting a Denial of Service (DoS) attack by jamming the coordinator

In the following scenarios, a Denial of Service (DoS) attack was simulated using various methods. Specifically, we focused on disrupting communication between the coordinator and the sensor. The software tools employed in this experiment were jamRF and RFJamming-FMRadio-SDR, along with a HackRF hardware to generate a jamming signal at 2.475 GHz. During a prior interception experiment, we established that the channel utilized was channel 25. The sensor was positioned 5 meters from the coordinator. We conducted tests using the HackRF at various distances between the devices. In all instances, the sensor failed to communicate with the coordinator, rendering it offline in the application. Additionally, when we moved in front of the sensor, this movement was not detected by the coordinator, even after the jammer was deactivated, meaning that the sensor stop trying to send the detecting message after several attempts. Figure 5 illustrates the flow diagram of the employed jammer. The noise source block can be configured to generate either Gaussian noise or impulse noise.[10]

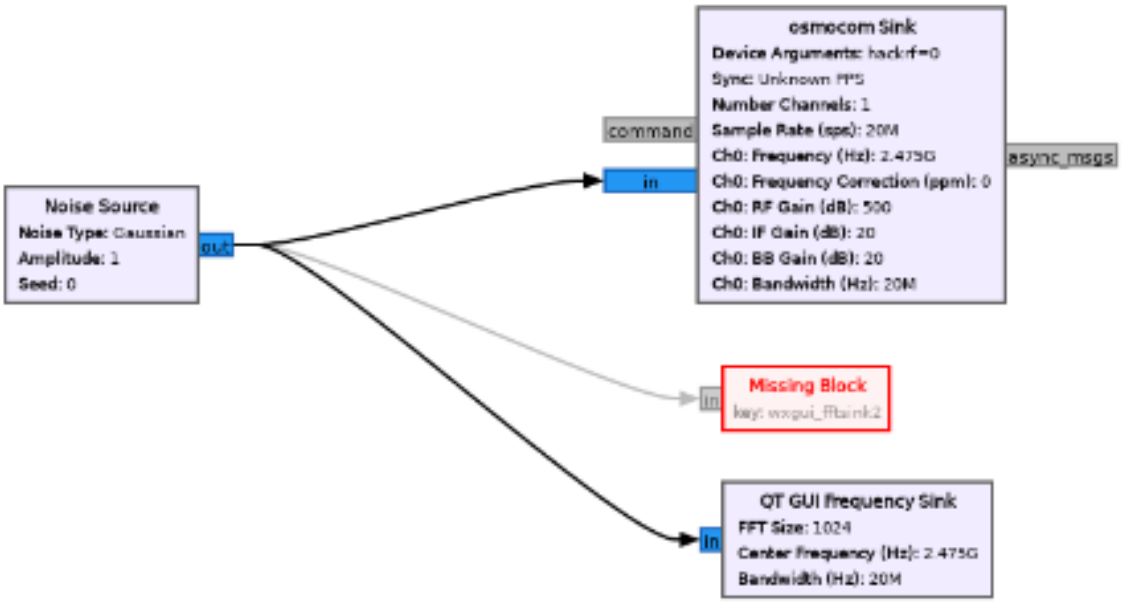

Figure 5: Jammer flow diagram

4.5 Conducting a DoS attack by jamming the sensor

In the subsequent step, we attempted to disrupt the sensor detection process. The presence sensor emits high-frequency (5.8 GHz) Millimeter-Waves, which reflect off objects and individuals. This sensor is highly sensitive, capable of detecting minimal movements such as breathing or a heartbeat. To jam the sensor, we utilized the HackRF, deploying the same jamming block diagram, to emit noise signals at 5.8 GHz (noting that the HackRF supports signals up to 6 GHz). Both Gaussian and impulse noise types were tested. Despite these efforts, the sensor successfully detected occupancy and communicated this to the coordinator, even when the jammer was placed directly beside the sensor and between the sensor and the approaching individual. This suggests two possibilities: the signal power level used was insufficient despite employing the maximum supported RF gain (500), or the sensor was actively scanning the radio spectrum and operating on a different frequency channel within the 5.8 GHz range than the one being jammed. No related information was found in the sensor's documentation.

We investigated also whether the sensor would remain in its initial state after the first detection due to the influence of the jamming signal, potentially making it incapable of future detections. This scenario could be done by an attacker who places a jamming device and subsequently returns without being detected. However, our findings demonstrated that the sensor consistently detected and reported each subsequent detection, indicating that under our experimental conditions, the sensor maintained a level of security against jamming that could compromise its functionality.

5 Discussion

This project includes sniffing the local ZigBee traffic. During our experiments extra care was taken to not intrude on any ZigBee systems within sniffing range. Only packets with keys associated with our testing environment were decrypted. Furthermore, we have found no traffic from devices not in our testing environment.

The vulnerabilities we found will be disclosed according to the OS3 disclosure policy if said policy will deem it necessary to do so. [12]

6 Conclusion

Our research investigates the secure deployment of ZigBee-based presence sensors for high-security applications, such as break-in detection. Specifically, the study focuses on identifying and mitigating attacks like jamming, interception, and message manipulation. To address this, the primary research question posed is: How can ZigBee-based presence sensors be safely utilized in high-security environments? This central question is further divided into two sub-questions:

What are the specific vulnerabilities present within ZigBee-based IoT networks? How can we effectively defend against attacks exploiting these vulnerabilities? In exploring these questions, we uncovered several critical vulnerabilities, particularly relating to the susceptibility of ZigBee networks to jamming and interception attacks. Notably, our findings highlight that these networks are vulnerable to jamming tactics targeting the coordinator, disrupting signal transmission, as well as data interception, especially during the handshake process.

To mitigate these risks, we propose specific defense mechanisms. These include employing AES encryption while avoiding the use of the default global trust center link key, which is publicly accessible and thus compromises security. Additionally, integrating frequency hopping techniques proves effective in minimizing jamming risks, as it allows the coordinator to switch to more secure communication channels dynamically.

In conclusion, our study demonstrates that with the implementation of these enhanced security measures and by linking sensor detections to additional devices, such as alarm systems, ZigBee-based presence sensors can be safely deployed for high-security applications.

References

[1] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based iot deployments," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1362–1380. [Online]. Available: https://ieeexplore.ieee.org/document/8835392

[2] J. Granjal, E. Monteiro, and J. Sá Silva, "Security for the internet of things: A survey of existing protocols and open research issues," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7005393

[3] A. Zohourian, S. Dadkhah, E. C. P. Neto, H. Mahdikhani, P. K. Danso, H. Molyneaux, and A. A. Ghorbani, "Iot zigbee device security: A comprehensive review," *Internet of Things*, vol. 22, p. 100791, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660523001142

[4] J. Tang, "Research on the security of zigbee wireless sensor network," in *Proceedings of the International Conference on Intelligent Systems, Communications, and Computer Networks (ISCCN 2022)*, vol. 12332, SPIE. Chengdu, China: SPIE, 2022, p. 123321P, yunnan Police College (China). [Online]. Available: https://www-spiedigitallibrary-org.proxy.uba.uva.nl/conference-proceedings-of-spie/12332/2653014/Research-on-the-security-of-ZigBee-wireless-sensor-network/10.1117/12.2653014.full#_=_

[5] S. Khanji, F. Iqbal, and P. Hung, "Zigbee security vulnerabilities: Exploration and evaluating," in *2019 10th international conference on information and communication systems (ICICS)*. IEEE, 2019, pp. 52–57. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8809115

[6] X. Fan, F. Susan, W. J. Long, and S. Li, "Security analysis of zigbee," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:37828092

[7] M. Kumar, V. Yadav, and S. P. Yadav, "Advance comprehensive analysis for zigbee network-based iot system security," *Discover Computing*, vol. 27, no. 1, p. 22, Jul 2024. [Online]. Available: https://doi.org/10.1007/s10791-024-09456-3

[8] J. T. Jackson, "Zigbee jamming," in *Journal of Physical Security*. Right Brain Sekurity, 2022, vol. Volume 15, pp. 1–13. [Online]. Available: https://rbsekurity.com/the-journal-of-physical-security.html

[9] akestoridis, "grc-ieee802154," Available at https://github.com/akestoridis/grc-ieee802154 (maint-3.10).

[10] timkim0713, "Rfjamming-fmradio-sdr," Available at https://github.com/timkim0713/RFJamming-FMRadio-SDR/tree/42b5a0698075963a2cd31f76076721c51711efd0/.

[11] (2024, Oct) Hackrf faq. [Online]. Available: https://hackrf.readthedocs.io/en/latest/faq.html#what-is-the-transmit-power-of-hackrf

[12] J. van der Ham, "Ethics procedures for the master sne," Available at https://www.os3.nl/_media/info/ecos3-procedure.pdf (2014/05/22).

A Appendix

Include supplementary material such as data sets, additional figures, or technical details. Ensure that the main text references these appendices where appropriate.