Discovering and Exploiting IoT Device Hidden Attributes: A New Vulnerability in Smart Homes

Xuening Xu
Stevens Institute of Technology
Department of Electrical and Computer Engineering
Hoboken, NJ, United States
xxu64@stevens.edu

Chenglong Fu
The University of North Carolina at Charlotte
Department of Software and Information Systems
Charlotte, NC, United States
chenglong.fu@charlotte.edu

Xiaojiang Du
Stevens Institute of Technology
Department of Electrical and Computer Engineering
Hoboken, NJ, United States
xdu16@stevens.edu

Bo Luo
The University of Kansas
EECS/I2S
Lawrence, KS, United States
bluo@ku.edu

Abstract

With the growing popularity and pervasive adoption of smart home Internet of Things (IoT) platforms, IoT security and privacy issues are gaining more attention. In this work, we reveal a new vulnerability inherent in most smart home IoT automation platforms and systems but previously unnoticed by the security community: the *hidden attributes*, i.e., attributes that are configurable by knowledgeable attackers through IoT APIs to effectively change device behaviors, but these attributes are not manageable or observable by users. An IoT device with compromised hidden attributes may behave differently from user expectations and cause severe security and safety consequences (e.g., burglary or fire). We present the root causes of the vulnerability and develop an approach to systematically discover hidden attributes. We evaluate a total of 31 commodity IoT devices of various types from 16 manufacturers and identify hidden attributes in all of them. Furthermore, we select several IoT devices with security and safety-critical hidden attributes and demonstrate the end-to-end hidden attribute attack on two popular IoT platforms: Samsung SmartThings and Amazon Alexa. In addition, we develop a tool that can automatically patch edge drivers and fix the hidden attribute issue. The source code of the auto-patching tool can be found in the Anonymous GitHub.

CCS Concepts

• **Security and privacy** → **Systems security**; • **Computer systems organization** → *Embedded and cyber-physical systems.*

Keywords

IoT, Security, Smart Home, Hidden Attribute

ACM Reference Format:
Xuening Xu, Chenglong Fu, Xiaojiang Du, and Bo Luo. 2025. Discovering and Exploiting IoT Device Hidden Attributes: A New Vulnerability in Smart Homes. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3744847

CCS '25, Taipei, Taiwan

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3744847

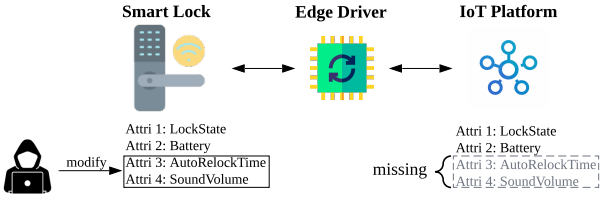

Figure 1: Mapping from device-supported attributes to available attributes on the IoT platform via edge drivers.

1 Introduction

A typical smart home Internet of Things (IoT) automation platform includes a cloud server, IoT hubs, and smartphone apps. To improve interoperability with a wider range of devices, major platforms like Samsung SmartThings and Amazon Alexa support third-party devices with diverse functionalities. On these platforms, devices from other vendors are known as third-party IoT devices. To ensure a seamless user experience and simplify app development, each platform provides predefined functions that specify the content and format of messages exchanged between third-party devices and the platform's hubs and cloud. Integrating a device model into a platform requires manufacturers to map the device's functionalities to those required by the platform, usually through middleware known as an "edge driver" or "device driver". This architecture allows platforms to offload device adaptation to manufacturers, simplifying the interface for users and apps by hiding intricate low-level details of IoT devices.

While edge drivers enable access to IoT device functions, they may not fully map advanced features. For instance, as shown in Figure 1, a smart lock natively supports four attributes, but only two of them are available on the IoT platform after the mapping via the edge driver. An attacker can modify the two unmapped attributes without being noticed by the IoT platform, as demonstrated in Section 6.3. Since users and automation apps are decoupled from the actual devices, they cannot access or view these configurable

attributes via any interfaces on the IoT platform. We refer to these non-observable attributes as the *hidden attributes*. Notably, many of these hidden attributes can be configured through API access on a local network, potentially exposing them to malicious attacks. The inadequate coverage of edge drivers leads to inconsistent mappings, making such exploits unnoticed by users. Continuing with the example in Figure 1, if an attacker changes the "AutoRelockTime" without the user's knowledge, the door could remain unlocked for an extended period, increasing the risk of burglary. The details are provided in Section 6.3.

In this paper, we conduct a systematic study of the hidden attributes on 31 commercial IoT devices with different features and communication protocols. We detail the processes of discovering and exploiting hidden attributes and present an automatic patching approach as a countermeasure. Our contributions are as follows:

- We are the first to discover a new vulnerability – *hidden attributes* – ubiquitous in IoT devices across various wireless technologies. We reveal the root causes and develop an effective approach for discovering hidden attributes.
- We present a systematic study and exploration of hidden attributes. We conduct experiments on 31 commercial IoT devices (using Zigbee, Z-Wave, or Wi-Fi) from 16 manufacturers and identify 119 hidden attributes.
- We introduce the *hidden attribute attack*, a new attack that exploits the hidden attribute vulnerability. We implement it on real IoT devices. Our experiments show that the attack can cause serious safety and security issues, which we have responsibly disclosed to the manufacturers.
- We design an approach to automatically patch edge drivers and fix the hidden attribute vulnerability. We implement this as an auto-patching tool on the Samsung SmartThings platform and make it available to the public. The tool is easy to use for general users and can effectively fix the hidden attribute problem.

The security impact of this work lies in the discovery of a previously unexplored vulnerability: hidden attributes that lack visibility and auditing across smart home platforms. These attributes can be stealthily manipulated by adversaries, making them especially suitable for persistent, undetected attacks, such as establishing long-term backdoors. By systematically exposing and demonstrating the severe vulnerability, we demonstrate how attackers can exploit them in real-world settings, while showing the urgent need for strong defensive measures across the entire IoT ecosystem. Our work belongs to proactive security research that is essential for identifying potential vulnerabilities before they are exploited by attackers, which not only sheds light on an undiscovered threat but also provides insights for fortifying systems at multiple aspects, including device manufacturers, IoT platform owners, end users, driver developers, and protocol designers.

Ethical Considerations and Responsible Disclosure. The objective of this project is to investigate the security risks associated with hidden attributes of IoT devices and to stimulate further attention from both the research community and manufacturers, which can benefit millions of smart home users. We conducted all experiments in a lab environment with isolated networks and did not probe or attack any device outside the lab. In February 2023, we disclosed our findings of hidden attributes and the associated potential security risks to the Connectivity Standards Alliance (CSA) and the manufacturers of all the tested devices. As of the submission of this manuscript, we have received responses from CSA, Samsung, and Amazon. They all confirmed the validity of the reported vulnerabilities and acknowledged our findings. CSA has notified its members to update existing devices that may be impacted. **Amazon awarded us a bounty of $2,500 for our disclosure.** We have not disclosed the hidden attribute attack to the public. Among the contributions mentioned above, only the auto-patching approach and tool rely on the artifacts, and we provide the artifacts of the auto-patching tool in the Anonymous GitHub[1]. Note that our contributions in discovering and studying the hidden attributes do not rely on any particular artifacts. The existence of hidden attributes is a fact that we have uncovered and studied.

The rest of the paper is organized as follows: we introduce the background in Section 2, followed by the system and threat models in Section 3. We present the hidden attribute discovery approach in Section 4, the details of the hidden attribute attack in Section 5, and the attack experiments with real-world IoT devices in Section 6. In Section 7, we introduce our auto-patching tool developed to mitigate the hidden attribute attack on the SmartThings platform. In Section 8, we present the lessons learned from this work, highlighting the root causes, systemic challenges, and strategic directions for strengthening IoT security. We review the related works in Section 9 and conclude the paper in Section 10.

2 Background

2.1 Smart Home IoT Architecture

In a typical smart home, there are IoT devices, IoT apps, and services being used. They can be roughly classified into two categories: home automation platforms and IoT devices.

Home Automation Platforms. A home automation platform consists of an integrated suite of hardware, software, and cloud services. For instance, the Samsung SmartThings platform encompasses smart home hubs, cloud servers, and mobile companion apps. These platforms serve as control centers to integrate a diverse array of IoT devices from third-party manufacturers, offering users versatile interfaces to control and manage their smart homes. Additionally, they enable the creation of automation rules and routines, empowering IoT devices to operate autonomously based on user-defined logic.

IoT Devices. Smart home IoT devices feature a wide variety of sensors and actuators to interact with surrounding environments. They use various wireless communication technologies, such as Wi-Fi, Zigbee, or Z-Wave, to exchange sensing and actuating messages with IoT servers. Each device typically possesses a unique set of functionalities, represented by distinct attributes. Device manufacturers develop specialized APIs to facilitate access to these attributes via wireless communications. However, differences in wireless protocols and message formats across devices complicate direct control through smartphone apps and hinder inter-device communication. To bridge this gap, IoT devices from different manufacturers are often integrated into an IoT home automation platform (e.g., Amazon Alexa) via smart IoT hubs.

[1] https://anonymous.4open.science/r/SmartThings-Edge-Driver-Auto-Patching-49CE

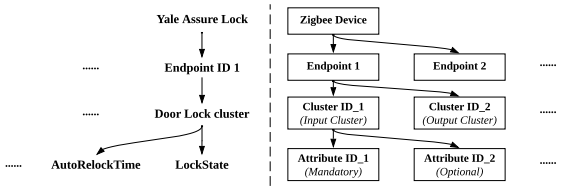

Figure 2: Cluster hierarchy and the structure of a Zigbee node

2.2 Edge Drivers

Edge drivers are software programs running on smart home hubs to support third-party IoT devices. As the middleware bridging IoT devices and home automation platforms, edge drivers are responsible for mapping different types of IoT device messages into unified formats that a home automation platform can understand. A typical edge driver is designed to integrate one or several specific types of IoT devices that share common functionalities (e.g., measuring temperature). They can be developed either by home automation platforms or by IoT device manufacturers. For platforms (e.g., SmartThings) where device manufacturers are required to provide their edge drivers, users can replace manufacturers' edge drivers with custom ones for devices. In this paper, *we primarily use Zigbee networks to demonstrate the hidden attribute attack.*

2.3 The Problem Scale

More and more smart home users are using smart home automation platforms, enabling the automation of IoT devices from different vendors through user-customized rules [38]. Leading platforms such as Amazon Alexa, Samsung SmartThings, and Google Home have over 100 million users globally [41, 44]. These users could be impacted by the vulnerability disclosed in this paper. Moreover, a user study by Chi et al. [32] reports that 61 out of 85 Amazon Mechanical Turk respondents use at least one device on automation platforms (not the vendor platform), potentially subject to the hidden attribute attack.

2.4 Zigbee Network and Device Hierarchy

A typical Zigbee network has only one Zigbee Coordinator (ZC), which regulates the commissioning and joining of other devices in the network. The Zigbee network created by the ZC is uniquely identified by a 16-bit Personal Area Network Identifier (PAN ID) and a 64-bit Extended PAN ID (EPID) [9]. Each node has a 64-bit unique MAC address assigned by its manufacturer. When a Zigbee device joins a Zigbee network, it receives a 16-bit network address, which is used as the source or destination address for Zigbee communications. The address 0x0000 is reserved for the ZC, and all other devices receive a randomly generated address [2].

Figure 2 illustrates the cluster hierarchy of a Zigbee node, using the Yale Assure Lock as an example. A Zigbee device has one or more endpoints, each containing several clusters. The Zigbee Cluster Library (ZCL) [27] defines the attributes for each cluster. The components are explained below.

Endpoint: Endpoints are logical extensions that support multiple applications. They are addressed with integers from 0 to 255. Endpoint 0 is used for the configuration of the Zigbee device, and 255 is used for broadcasting to all endpoints.

Cluster: A set of functions defined in the ZCL [27] to build Zigbee applications, identified by a 2-byte cluster ID. There can be multiple clusters on each endpoint, which are either Input (server) or Output (client). Servers store and manage attributes, while clients manipulate them. Commands like Read/Write Attributes are sent by clients and received by servers, with responses sent back from servers to clients. In this work, we focus on servers (i.e., Input Clusters) that are related to configurations and can be modified to change the behavior of devices. Moreover, manufacturers are free to add manufacturer-specific clusters to a standard device endpoint.

Attribute: A cluster attribute is a variable with a defined 2-byte attribute ID, data type, and access type (e.g., Read, Write, Read/Write, Reportable). Each attribute ID is associated with an attribute name defined in the ZCL. If an attribute supports "Read" access, its value can be read by a Read Attributes command. If it supports "Write", its value can be modified by a Write Attributes command. Each cluster may include multiple attributes, such as device states or data items. Some mandatory attributes must be implemented by manufacturers to fulfill the desired functionalities if a device includes the cluster. Other attributes are optional, and it is the manufacturer's choice to implement them. Manufacturers can also add manufacturer-specific attributes to a standard cluster.

3 The System and Threat Models

3.1 The System Model

In our study, we focus on a typical smart home setup, as illustrated in Figure 3, which includes IoT devices, a smart home hub, a home automation cloud server, and a smartphone IoT app. The platform operator (e.g., Samsung) can directly control the hub and the smartphone app. Note that third-party IoT devices are produced by various third-party manufacturers. The cloud server is responsible for exchanging event messages and commands with IoT devices via the smart home hub. Additionally, it hosts home automation rules that are triggered by event messages. When a homeowner uses the smartphone app to view or control an IoT device, the view/control command is sent from the app to the cloud server and then to the hub. An IoT hub is equipped with Wi-Fi and Zigbee/Z-Wave modules, which can be used by edge drivers to communicate with a wide range of IoT devices. Most smart home users rely on the manufacturer's official drivers that are pre-loaded on the hub. Smartphones typically lack Zigbee and Z-Wave communication capabilities, preventing apps from directly accessing IoT devices. Consequently, most Zigbee/Z-Wave devices are designed to operate with third-party smart hubs and usually do not come with a manufacturer-provided smartphone app. Thus, users do not have a mechanism to view or access hidden attributes of the Zigbee/Z-Wave devices.

3.2 The Threat Model

The Attack Target. The target of the attack could be any residential or organizational IoT automation network and the devices within it. The vulnerabilities identified in this study are inherent in the design of home automation platforms. Therefore, all enrolled devices that do not carefully handle their attributes and APIs are susceptible to the attack. The attack is not limited to devices from specific

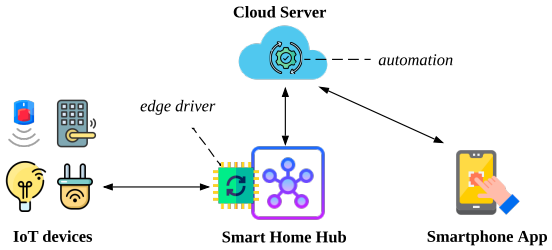

Figure 3: Overview of IoT devices in smart homes.

manufacturers. Instead, general Zigbee, Z-Wave, and Wi-Fi devices could all be vulnerable to the attack.

The Attack Objective. The attacker attempts to interfere with the operation of the target IoT devices, particularly to disrupt safety-critical operations. Another objective is to remain stealthy, i.e., changes to device attributes are expected to be unnoticeable to the victim. For instance, an effective attack is to significantly delay the auto-lock function of a smart lock, while the owner does not notice any change or alert in the home hubs or smartphone apps. More specifically, the attacker focuses on hidden attributes, whose threat stems from their inherent stealth. Regular attribute modifications typically trigger visible notifications or alerts, which can be detected by users or security monitoring systems. In contrast, hidden attributes can be altered silently, allowing attackers to manipulate device behavior without raising suspicion. Because these attributes are unknown to users, unauthorized changes are unlikely to be noticed and cannot be easily reverted even after detection, unless resetting the device. This makes hidden attributes particularly attractive for attackers aiming to establish persistent, undetected control over IoT devices, such as through long-term backdoors.

The Attackers. We assume the attacker has control of a malicious device on the target home automation platform. This is achievable by an *external adversary* or a *malicious insider* of the network. In this paper, ***we implement the external attack***, which does not rely on such a strong assumption.

As an external adversary, they can plant an attack device into the network. To do so, the attacker may employ a known attack, e.g., [35]. In Section 3.3, we will present an approach that places a malicious device into a Zigbee network from outside of the Zigbee network without knowing network credentials beforehand. In this case, the attacker needs to be within the wireless signal range of the target network. This is easily achievable since the practical range of Zigbee networks is up to 325 feet indoors and 980 feet outdoors [31, 58], e.g., the attacker may park a car near the target household and launch the attack from the car. Because legacy Zigbee devices only support the default link key [42], Zigbee coordinators across all Zigbee networks accept the default link key, known as "ZigbeeAlliance09" [23]. External attacks rely on the weak authentication mechanisms, which are widely adopted by existing Zigbee hubs and devices. The attacker needs a window of opportunity when the victim sets the hub into pairing mode.

The attacker does not interfere with users' normal interactions with the target device, i.e., the device operates normally until the attacker explicitly interrupts a function. The attacker does not compromise the codebase of the victim devices or any other component of the target home automation network. The attacker does not attempt to access or compromise the companion apps, users' smartphones, or the cloud servers – all these components are considered benign and secure.

3.3 Attack Precursors

In an *external* attack, the attacker deploys an attack device, which is a regular Zigbee node, such as a light switch, with the "customized" function to conduct the hidden attribute attack. The attack preparation involves the following steps:

1) Approaching. The adversary gets into the range of the target Zigbee network (i.e., a maximum range of 325 feet indoors and more than 980 feet outdoors [31, 58]), which is feasible.

2) Reconnaissance. Note that the IEEE 802.15.4 MAC layer encryption is not adopted in practice due to the high power consumption of cryptographic operations and the challenges of key distribution [57]. Many IoT devices are battery-powered, and employing MAC layer encryption significantly shortens their battery life. This is the main reason why most IoT device manufacturers do not adopt MAC layer encryption. Furthermore, it does not address this issue for the billions of existing (legacy) IoT devices by making it mandatory for IoT manufacturers to use MAC layer encryption. This leaves the network layer header exposed as plaintext, and the attacker can sniff Zigbee packets to obtain the plaintext information (e.g., network addresses) in the network layer header. Note that the MAC layer header is in plaintext even if MAC layer encryption is used. It is important to emphasize that the network layer payload and the upper layers are still encrypted by the network key.

3) Disconnecting. The attacker can utilize existing attacks [25, 57] to force a smart home hub into pairing mode. For instance, the attacker could carry out the Disconnection attack proposed in [57] by bit-level manipulations, which causes a Zigbee device to be disconnected from the network. This necessitates a manual reset and a commissioning phase in order to reconnect the Zigbee device to the network. Note that the Disconnection attack can generate protocol-compliant packets through bit-level manipulations on the encrypted network payload, and can be launched outside the Zigbee network without needing to know the Zigbee network key. We have *implemented the Disconnection attack* proposed in [57].

4) Joining. The attack device automatically joins the target Zigbee network without knowing the network key during the commissioning phase of the coordinator, which is an opportunity window created by the Disconnection attack. Note that once the attack device has been placed near the target Zigbee network, it can keep scanning the environment to detect the re-pairing opportunity window to join. After that, the attack device receives the network key from the coordinator. With the network key, the attack device can decrypt Zigbee packets to find devices in the clusters of interest. The attacker extracts the information of the target devices, e.g., network addresses, cluster IDs, and endpoint numbers, and utilizes them for the hidden attribute attack.

Once joining the network, the attack device could directly send commands to a target device. The target device will change its status and then generate an event (e.g., a feedback event) indicating the change. The user and the IoT system can detect the device status change and/or the feedback event and discover the above attack. On the other hand, the hidden attributes are not visible to users,

Table 1: Subset of Door Lock cluster attributes defined in the ZCL: R - Readable; W - Writable; *W - Optionally Writable; P - Reportable; M - Mandatory; O - Optional.

Cluster: Door Lock (0x0101)				
Attr. ID	**Attribute Name**	**Type**	**Access**	**M/O**
0x0000	LockState	enum8	RP	M
0x0001	LockType	enum8	R	M
0x0002	ActuatorEnabled	bool	R	M
0x0023	AutoRelockTime	uint32	R*WP	O
0x0024	SoundVolume	uint8	R*WP	O
0X0025	OperatingMode	enum8	R*WP	O
0x002B	EnablePrivacyModeButton	bool	RWP	O
0x0030	WrongCodeEntryLimit	uint8	R*WP	O
0x0031	UserCodeTemporaryDisableTime	uint8	R*WP	O

and their modifications cannot be easily detected, making them a unique and significant concern for smart homes.

4 Discovering Hidden Attributes

In this paper, we present a new vulnerability referred to as the ***hidden attribute vulnerability***, which can be exploited by an attacker to launch a hidden attribute attack. The attack overwrites the values of the hidden attributes on a target IoT device, causing it to operate in an undesired state. This could lead to serious security or safety risks, e.g., a motion sensor failing to detect or report the movements of intruders.

4.1 Definition of Hidden Attributes

Edge drivers are highly diverse and their coverage of attributes depends on the efforts and carefulness of developers. Many edge drivers only implement essential attributes of the corresponding IoT devices and omit others. The omitted attributes, although supported by devices with open APIs, are invisible and inaccessible to the home automation platform and homeowners. Below we formally define the *hidden attributes*.

The Hidden Attributes refer to attributes of IoT devices that satisfy all the following conditions: 1) they are implemented by IoT device manufacturers and can affect the devices' behaviors; 2) they can be accessed and modified by directly calling APIs on IoT devices; and 3) they are not supported in the edge drivers deployed in the smart home hubs. ***Note:*** General users cannot see the hidden attributes when they use the existing remote control interfaces, including but not limited to third-party apps, voice assistants, IoT hubs, and most companion apps provided by device manufacturers.

4.2 Discovering Zigbee Attributes

Here we aim to discover attributes that exist on IoT devices. The discovery process can be conducted in a controlled environment with software tools that automatically scan devices, where it is reasonable to assume that there is sufficient time and resources to discover attributes on Zigbee devices. Moreover, a custom Zigbee coordinator can be developed to facilitate the discovery process, as it allows for flexible configuration of the Zigbee network as needed, enabling a thorough exploration of device attributes. According to the ZCL, two commands are used to discover attributes supported in a Zigbee device: *"Discover Attributes"* and *"Read Attributes"*. For both commands, the "Manufacturer Specific" sub-field must be set to "True" in order to discover manufacturer-specific attributes. Otherwise, only ZCL-defined attributes will be discovered.

Discover Attributes Command: A Discover Attributes command takes a 16-bit "Start Attribute Identifier" as the starting number and another 8-bit "Maximum Attribute Identifiers" as the maximum number of attribute IDs to return. A list of supported attributes is included in the response to the Discover Attributes command.

Read Attributes Command: A Read Attributes command takes a list of attribute IDs as parameters. If the device supports the attribute, a status "SUCCESS" with a code 0x00 is returned in the Read Attributes Response command. Otherwise, a status "UNSUPPORTED_ATTRIBUTE" with a code of 0x86 is returned to indicate that the attribute does not exist on the device. By traversing all possible attribute IDs (i.e., 0 to 65535), the supported attributes within each cluster can be obtained.

After obtaining device attributes, we need to know the value range that each attribute accepts, and then we can write with valid values. Note that in the *Discover/Read Attributes Response* commands, every supported attribute is returned along with the data type (e.g., 8-bit Enumeration, Boolean). Although the data type indicates the value range to some extent, not every value in that range is guaranteed to be accepted by the particular Zigbee device, as the manufacturer's implementation may add additional constraints on the valid values due to various considerations. For example, the data type of the attribute *OperatingMode* is an 8-bit enumeration as shown in Table 1, which has 256 possible values. However, the Zigbee Yale Assure Lock only accepts 0, 1, 2, and 3 as valid values since it only supports 4 different operating modes.

We can try all possible values for the specific data type and check if values are accepted by the target Zigbee device. Specifically, we send *Write Attributes* commands to the target device in an attempt to set new values for the attributes. In each *Write Attributes* command, we need to include the new value and the data type of the attribute. Three possible status codes can be received in the *Write Attributes Response* command: 1) SUCCESS: attribute is updated successfully; 2) INVALID_VALUE: the supplied value is invalid and thus not accepted by the target device; 3) READ_ONLY: attribute is read-only. We record all values with a returned status of SUCCESS as valid values for a writable attribute, and these values can be used later for the exploitation of the hidden attributes.

4.3 Discovering Z-Wave and Wi-Fi Attributes

Z-Wave. Exploring the attributes of Z-Wave devices involves the utilization of open-source libraries, such as Z-Wave JS [22], to discover Z-Wave device attributes. Z-Wave JS acts as a versatile tool that can be employed to create a server-like environment for Z-Wave network management. Note that running the library requires expertise in IoT development and dedicated wireless dongles. Most general users would not be able to do this and instead use smartphone apps to control IoT devices. This tool facilitates the interaction with Z-Wave devices, enabling not only device management but also effective discovery of their attributes.

Wi-Fi. Wi-Fi devices have considerable flexibility in attribute definition. On a Wi-Fi device, its manufacturer can freely customize the names and contents of attributes and the format of messages. To enhance interoperability with various smart home hubs, many

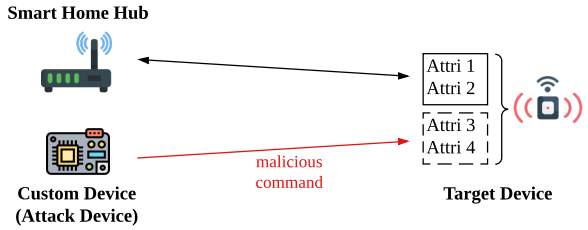

Figure 4: Overview of the hidden attribute attack. The attributes within the dotted box are hidden attributes.

Wi-Fi IoT devices implement the Simple Service Discovery Protocol (SSDP), thereby achieving a plug-and-play capability. When employing SSDP, these devices respond to discovery requests from smart home hubs with a URL pointing to an XML document that contains a detailed list of supported attributes. A simple program or some existing tools [1, 4] can be used to automatically detect SSDP-supported Wi-Fi devices in the vicinity.

4.4 Manual Discovery of Hidden Attributes

Now that we have identified all supported attributes on IoT devices, the next step is to determine which attributes are hidden. This involves identifying observable attributes on home automation platforms, with the remaining classified as hidden. This can be done either manually or through an automated approach.

Using Zigbee as an example, we focus on attributes with "Write" access, as these can be modified by an attacker, potentially compromising smart home security. We use the Door Lock cluster to illustrate the presence of hidden attributes. Table 1 lists the mandatory (M) and optional (O) attributes of the Zigbee Door Lock cluster. Mandatory attributes include lock state, lock type, and actuator status, while optional attributes enhance functionality if implemented by manufacturers. For instance, when a Yale Assure Lock is connected to a SmartThings hub, only two attributes, *LockState* and *BatteryPercentageRemaining*, are visible in the app. However, the lock supports additional attributes like *AutoRelockTime* and *SoundVolume*, which are hidden from the user. Changes to these hidden attributes, which are not reflected in the app, could lead to significant safety and security risks, as detailed in Section 6.

4.5 Automated Discovery of Hidden Attributes

For platforms with open-source edge drivers or APIs, we develop automated methods to retrieve observable attributes. We implement two automated methods on the SmartThings platform: 1) using the SmartThings API [52], and 2) analyzing edge drivers [53].

SmartThings API manages IoT devices on the SmartThings platform and accesses device attributes via HTTP GET requests, returning a list of connected devices with their supported attributes in JSON format. This method can be used when IoT devices are already connected to the platform.

Edge Drivers contain a list of supported attributes for different IoT devices. By analyzing edge drivers, we can extract attributes supported by each device model. This method can be used when IoT devices are unavailable or not connected to the platform.

We implement both methods to automatically obtain observable attributes on the SmartThings platform. The two automated methods obtain the same results for the same device models, confirming that the unobservable attributes are hidden.

5 Exploiting Hidden Attributes

The hidden attribute vulnerability exists in many IoT devices, including Zigbee, Z-Wave, and Wi-Fi devices. This vulnerability can be exploited by an attacker to launch a hidden attribute attack, which can overwrite the values of hidden attributes on IoT devices and cause security issues in a smart home. In this work, we mainly use Zigbee devices to demonstrate the hidden attribute attack.

5.1 Overview of the Hidden Attribute Attack

In this work, we present a new attack, referred to as the ***Hidden Attribute Attack*** (illustrated in Figure 4): The attacker overwrites the hidden attribute values on a target IoT device so that it operates in an undesired state, which could cause serious security or safety risks, e.g., a motion sensor fails to detect or report an intruder's movements. The attack is expected to be stealthy – users would not receive any notification or alert through smartphone apps or other channels.

The hidden attribute attack can be classified into two categories based on the target device type: *Attacks on Actuator Attributes* and *Attacks on Sensor Attributes*.

Attacks on Actuator Attributes. This attack modifies actuators' attributes. Since an actuator is the endpoint of executions, changes to attribute values usually affect the device's configurations and result in different behaviors. Such intentional misconfigurations could directly lead to safety and security issues. For example, according to the ZCL Specification, the *AutoRelockTime* attribute is "the number of seconds to wait after unlocking a lock before it automatically locks again". Maliciously changing the value of this attribute on a smart lock to a large value causes the lock not to be automatically locked promptly after the user leaves home, leaving a time window for burglars or intruders to enter the home.

Attacks on Sensor Attributes. Zigbee and Z-Wave sensors detect changes in a physical environment. They usually report those changes to a smart home hub and then to the cloud server to support home automation functions. Changes in the sensor attribute values may affect the execution of automation rules. Many automation rules are related to the security and safety of a smart home. If they are not properly executed, home security and safety may be compromised. For instance, a simple rule, "When the motion sensor detects motion while the user is not home, sound the alarm" aims to detect unusual movements (e.g., a burglary) when the user is not at home. By maliciously changing the sensitivity of motion detection to a minimum value, the motion sensor may fail to detect the motion when a burglar breaks into the home. As a result, the alarm would not sound, and the user may lose valuable items.

5.2 The End-to-End Hidden Attribute Attack

In this subsection, we present the end-to-end hidden attribute attack, which has been implemented by our team using real Zigbee devices and networks. An attacker can place a custom Zigbee device near the victim's home as the attack device and connect it to the

home's Zigbee network. A custom Zigbee device is a general Zigbee device with the customized function to send Write Attributes commands. The Zigbee communication range can be up to 325 feet indoors. Note that except during the commissioning phase (the pairing mode), a Zigbee network is closed and the Zigbee coordinator does not accept any joining requests from new Zigbee devices. To make a Zigbee coordinator enter pairing mode, the attacker can utilize existing methods (e.g., the Disconnection attack in [57], the selective jamming and spoofing attacks in [25]), which require the user to reset the devices (otherwise, the devices cannot be used). In our work, we have implemented the Disconnection attack in [57]. Thanks to the source code shared by the authors of [57], our implementation can successfully disconnect a Zigbee device from a Zigbee network. After a Zigbee device is disconnected, the user has to make the smart home hub enter pairing mode in order to reconnect the disconnected device.

The attacker does not have to stay near the victim's house to wait for the time window of re-pairing. Instead, the attacker can leave the attack device there. The attack device continuously scans nearby Zigbee networks and waits until a time window occurs (i.e., when the user re-pairs a disconnected device). This attack can be considered a persistent threat and does not have a tight time requirement. In the worst case, if the user does not re-pair a disconnected device for a long time and the attack device drains its battery, the attacker can come back later and replace/recharge the battery multiple times until the attack is successful. Additionally, the attacker may attach a power bank or a solar charger power bank to the attack device, which could extend the battery life to months or even years. To sum up, there is no tight time requirement for launching the attack. Besides a targeted attack that waits for the re-pairing time window, the attacker can conduct untargeted attacks by scanning nearby Zigbee networks and joining the one that happens to be in pairing mode and launching attacks.

During the pairing phase, the attack device can be automatically (i.e., without any manual operations) connected to the Zigbee network and then obtain the network key from the Zigbee coordinator, which is used to decrypt Zigbee messages. Because Zigbee devices could go offline from time to time even without attacks, a user would not think that this is an attack. Then the attack device sniffs Zigbee packets and finds events from a device of interest to obtain the network address and the endpoint number, which are used to send *Write Attributes* commands to the target device. The above attack can be done within a few minutes after the attack device joins the Zigbee network. Furthermore, the attack device can disguise itself as a commonly used device, such as a light switch or a smart plug, so that it would not be detected by the user.

6 Evaluation

We conduct a comprehensive evaluation of the hidden attribute vulnerability and attack. This work primarily focuses on Zigbee devices, which are presented in Section 6.1 through Section 6.3. We also evaluate the hidden attribute issue of Z-Wave and Wi-Fi devices in Section 6.4. Similar to the limitations in other works, such as *Protected or Porous* [45], we also require physical testing targets (i.e., various IoT devices in our case) for evaluation, which affects scalability. Nevertheless, we make our best efforts to select

Table 2: *Writable* or *observable* attributes from scanning on devices: Yale Assure Lock, Kwikset Smart Lock, Frient Smart Siren, and Philips Hue Motion Sensor. Observable attributes are noted in the table and others are writable. Entries with "-" symbol mean the attribute is not supported. The unit of time-related value is seconds. "mfr" means "manufacturer".

	Attr. ID	Attribute Name	Type	Value Range	
				Yale	Kwikset
Lock	**Cluster: Door Lock (0x0101)**				
	0x0000	LockState *(Observable)*	enum8	*ReadOnly*	*ReadOnly*
	0x0021	Language	string	es,en,fr	-
	0x0022	LEDSettings	uint8	-	0, 2
	0x0023	AutoRelockTime	uint32	0 - 2^{32}	6 - 2^{8}
	0x0024	SoundVolume	uint8	0, 1, 2	0, 2
	0x0025	OperatingMode	enum8	0, 1, 2, 3	*ReadOnly*
	0x0028	EnableLocalProgramming	bool	-	0, 1
	0x0029	EnableOneTouchLocking	bool	0 - 2^{8}	-
	0x002A	EnableInsideStatusLED	bool	0 - 2^{8}	-
	0x002B	EnablePrivacyModeButton	bool	0 - 2^{8}	-
	0x0030	WrongCodeEntryLimit	uint8	3 - 10	1 - 10
	0x0031	UserCodeTemporaryDisableTime	uint8	10 - 180	1 - 2^{8}
	0x0032	SendPINOverTheAir	bool	0 - 2^{8}	0 - 2^{8}
	0x0033	RequirePINforRFOperation	bool	-	0 - 2^{8}
	0x0040	AlarmMask	map16	0 - 2^{16}	0 - 2^{16}
	0x0041	KeypadOperationEventMask	map16	0 - 2^{16}	0 - 2^{16}
	0x0042	RFOperationEventMask	map16	0 - 2^{16}	0 - 2^{16}
	0x0043	ManualOperationEventMask	map16	0 - 2^{16}	0 - 2^{16}
	0x0044	RFIDOperationEventMask	map16	0 - 2^{16}	-
	0x0045	KeypadProgrammingEventMask	map16	0 - 2^{16}	0 - 2^{16}
	0x0046	RFProgrammingEventMask	map16	0 - 2^{16}	0 - 2^{16}
	0x0047	RFIDProgrammingEventMask	map16	0 - 2^{16}	-
	Cluster: Power Configuration (0x0001)				
	0x0021	BatteryPercentageRemaining *(Observable)*	uint8	*ReadOnly*	*ReadOnly*
Siren				**Frient Smart Siren**	
	Cluster: IAS WD (0x0502)				
	0x0000	MaxDuration	uint16	0 - 2^{16}	
Motion Sensor				**Hue Motion Sensor**	
	Cluster: Occupancy Sensing (0x0406)				
	0x0000	Occupancy *(Observable)*	map8	*ReadOnly*	
	0x0010	PIROccupiedToUnoccupiedDelay	uint16	0 - 2^{16}	
	0x0030	Motion Sensitivity (mfr-specific)	uint8	0, 1, 2	
	Cluster: Power Configuration (0x0001)				
	0x0021	BatteryPercentageRemaining *(Observable)*	uint8	*ReadOnly*	
	Cluster: Temperature Measurement (0x0402)				
	0x0000	MeasuredValue *(Observable)*	int16	*ReadOnly*	

representative devices that can effectively demonstrate our findings. We test twenty-six popular commercial off-the-shelf Zigbee devices from twelve different manufacturers on two popular IoT platforms: Samsung SmartThings and Amazon Alexa. In total, we identify eighty-eight hidden attributes on the Zigbee devices tested in this work. We obtain the above results based on experiments conducted independently by two authors and then confirmed by a third author. Due to the page limit, we only present and discuss the results of some devices as listed in Table 2. After identifying hidden attributes, we further validate and demonstrate the end-to-end hidden attribute attack on several security-critical devices. The selection of the 26 Zigbee devices is guided by the following considerations: 1) Popular Brands: Devices are chosen from well-known and trusted brands to ensure coverage and reliability. 2) Comprehensive Coverage: A wide range of device categories are included, such as locks, motion sensors, and water sensors, to ensure diverse functionalities and applications. Our experiments show that all these devices are vulnerable to the hidden attribute attack, regardless of whether they are using the default link key or not.

6.1 Results of Attribute Exploration

To systematically identify attributes supported by each Zigbee device, we develop an automated Zigbee attribute scanner using a laptop equipped with a SONOFF Zigbee 3.0 USB Dongle [17]. On this laptop, we leverage the Zigbee2MQTT [40] library to establish a Zigbee network coordinator, responsible for managing and interfacing with the Zigbee devices being tested. Building upon this library, we develop an application module that can autonomously scan and record the attributes of the connected devices. In this work, we focus on the primary Zigbee clusters on the Zigbee devices (e.g., the Door Lock cluster on smart locks and the Occupancy Sensing cluster on motion sensors), and non-primary clusters such as Power Configuration will not be discussed due to space limit.

As described in Section 4.2, *Discover Attributes* and *Read Attributes* commands are used to identify supported attributes. However, we find that some Zigbee devices return only a limited number of attribute IDs (e.g., 10), even when the "Maximum Attribute Identifiers" is set to 255, likely due to a hardcoded constraint in the device's firmware. To address this, we use the *Read Attributes* command to identify all supported attributes. We develop a customized *Read Attribute Function* that scans all possible 16-bit attribute IDs within the cluster (0 to 65535), recording those with a "SUCCESS" status. This process takes less than two hours at a rate of 10 packets per second and can be completed by attackers in preparation at their own places as mentioned in Section 4.2. Additionally, we create a function to write attribute values, filtering out READ_ONLY attributes and determining valid value ranges for writable attributes.

Although some Zigbee device manufacturers have official documents [24] that describe the attributes supported by their devices, most end users do not search for or read such technical documents. Additionally, the attribute names listed in the documents often differ from those defined in the ZCL. Furthermore, some attributes, such as *sendPINOverTheAir*, discovered in our work are not listed in the manufacturers' documents.

There are many writable and hidden attributes found in the devices tested in this work. For instance, we discover a total of 29 attributes for the Zigbee Yale Assure Lock under the Door Lock cluster, of which 18 are writable. For the Zigbee Kwikset 914 SmartCode Smart Lock, 29 attributes are discovered under the Door Lock cluster, 14 of which are writable. Among all the discovered attributes for the two locks, only one attribute (i.e., LockState) is observable, and the rest are hidden. Table 2 lists all the writable attributes within the Door Lock cluster for the two locks. Note that the values in this table can be "successfully" written to devices (the Write Attribute commands yield "SUCCESS", as mentioned in Section 4.2), but may not necessarily take effect due to some implicit constraints set by manufacturers. Among the over twenty attributes in the Door Lock cluster, only one attribute (*LockState*) is observable to users. All the writable attributes are hidden from the end users, and users would not even notice when these attributes are maliciously changed. This hidden attribute problem creates serious vulnerabilities that attackers can exploit to launch attacks, which could affect the security and safety of smart homes and homeowners.

Table 2 also lists the writable attribute in the IAS WD cluster supported on the Frient Smart Siren. It does not have any observable attributes on smartphone apps. In addition, Table 2 lists the writable or observable attributes supported on the Philips Hue Motion Sensor. The three observable attributes are noted (with the word "Observable" in blue) in the table, and the other two attributes are writable. We discuss the observable attributes in Section 6.2.

The Yale Assure Lock (*YRD226-HA2-619*) [20] supports three languages: Spanish (es), English (en), and French (fr). The lock's language changes when a new language code is written to the *Language* attribute. The *AutoRelockTime* attribute, with a uint32 data type, accepts values between 0 and 4,294,967,295 (i.e., 2^{32}, the maximum of a 32-bit number). However, when set to a very large value, the lock defaults to relocking three minutes after the last unlock. Reading the *AutoRelockTime* attribute reveals that the lock limits this value to a maximum of 180 seconds when a value greater than 180 is provided. The *SoundVolume* attribute adjusts the chime volume, where 0 mutes it, and the maximum value provides the loudest chime. The *OperatingMode* attribute supports several modes: 1) Normal Mode: enable all interfaces (keypad, RF, RFID); 2) Vacation Mode: only enable RF interaction and disable keypad; 3) Privacy Mode: only the thumb turn works, disabling all external interactions; 4) No RF Lock or Unlock: disable RF interaction.

WrongCodeEntryLimit and *UserCodeTemporaryDisableTime* define the number of incorrect passcode attempts allowed before lockout and the lockout duration, respectively. *SendPINOverTheAir* controls whether the PIN code in door lock messages is masked. *RequirePINforRFOperation* determines if a PIN is needed for RF lock operations. *EnableOneTouchLocking*, *EnableInsideStatusLED*, and *EnablePrivacyModeButton* are boolean attributes. Although they accept values from 0 to 255, 0 means "False" and any positive value means "True". If *EnablePrivacyModeButton* is set to "True", the lock can switch to Privacy Mode with a single press of the privacy button. The rest of event mask attributes accept values from 0 to 65535 and can be configured by setting the corresponding bit in the mask.

Most of the writable attributes on the Kwikset Smart Lock (*914TRL ZB 3.0 L03*) [6] are similar to those of the Yale Assure Lock. In addition, the Kwikset Smart Lock supports the *LEDSettings* to indicate if the lock uses LEDs for signalization. Unlike the Yale Assure Lock, which sets the value to 180 for any value greater than 180, the *AutoRelockTime* attribute only accepts numbers from 6 to 255. The *OperatingMode* attribute is Readable and Optional Writable for locks (see Table 1), but for the Kwikset Smart Lock, it is read-only. The local programming features include adding and deleting user codes, schedules and so on, which can be enabled by setting the *EnableLocalProgramming* attribute to "True".

The Frient Zigbee Smart Siren (*SIRZB-110*) [5] is used to emit loud alarm noise during emergencies. It can be flexibly integrated into automation rules and triggered by various sensor events. We identify one writable attribute: *MaxDuration* within the IAS WD cluster. It specifies the maximum time that the siren will sound continuously and accepts a value ranging from 0 to 65535. The siren can be completely muted by setting this attribute to 0.

The Philips Hue Motion Sensor [10] has two writable attributes. *PIROccupiedToUnoccupiedDelay* belongs to the Occupancy Sensing cluster. It defines the shortest time for the sensor to change back to the unoccupied status after detecting motion, and it has a configuration range from 0 to 65534 seconds (more than 18 hours). The other writable attribute with the ID 0x0030 is a manufacturer-specific attribute and defines the sensitivity level of the motion sensor [7].

Table 3: Consequences of the Actuator Attribute Attack on hidden attributes. "Affected Devices" only includes tested devices.

Attr. ID	Attribute Name	Original	Original Behavior	New	New Behavior	Affected Devices
0x0023	AutoRelockTime	15 seconds	The door is locked again 15 seconds after being unlocked.	180 seconds	The door is locked again 180 seconds after being unlocked.	Yale, Kwikset
0x0029	EnableOneTouchLocking	True	The lock can be locked with one touch on the touchpad.	False	The lock cannot be locked with one touch on the touchpad.	Yale
0x0030	WrongCodeEntryLimit	3	The lock enters a lockout state after 3 incorrect code attempts.	10	The lock enters a lockout state after 10 incorrect code attempts.	Yale, Kwikset
0x0031	UserCodeTemporaryDisableTime	60 seconds	The lock shuts down for 60 seconds after reaching the wrong code limit.	10s for Yale 1s for Kwikset	The lock shuts down for 10 / 1 second(s) after reaching the wrong code limit.	Yale, Kwikset
0x0032	SendPINOverTheAir	False	The PIN field is masked in any door lock cluster message payload.	True	The PIN field is not masked in any door lock cluster message payload.	Yale, Kwikset
0x0000	MaxDuration	900 seconds	The siren sounds for 900 seconds before it stops.	0 seconds	The siren does not sound at all.	Frient Siren

It can take values of 1, 2, or 3. The detection distance of the motion sensor increases with higher values.

6.2 Observable Attributes in Apps

After identifying the attributes supported by the devices, we check their availability on two popular IoT platforms: Samsung SmartThings hub V3 and Amazon Alexa Echo 4th Gen. We connect the devices to the hubs and review all user interfaces, including apps, automation rules, notifications, and voice commands, to determine if the attributes in Table 2 are observable and configurable.

Table 2 shows that the two smart locks have two observable attributes: *LockState* and *BatteryPercentageRemaining*, both visible on the SmartThings app, while only *LockState* is shown on the Alexa app. For the Frient Smart Siren, the attribute listed in Table 2 is hidden on the SmartThings app, and the siren is unsupported on Alexa via direct connection to the Echo 4th Gen. Even when delegated through other platforms like SmartThings, the *MaxDuration* attribute remains hidden on Alexa. Additionally, Table 2 shows that the Philips Hue Motion Sensor has three observable attributes on the SmartThings app, but only *Occupancy* is displayed on Alexa. The results show that IoT platforms and apps provide users access to only a portion of the attributes while hiding the rest.

6.3 Consequences of Hidden Attribute Attack

We evaluate and show the effectiveness of the hidden attribute attack. For the actuator attribute attack, we compare the target device behaviors before and after the attack. For the sensor attribute attack, we evaluate how the modified attribute affects automation rules. We select seven Zigbee devices with different capabilities (i.e., lock, siren, motion, vibration, presence, and temperature) to illustrate the consequences of the actuator attribute attack and the sensor attribute attack in Section 6.3.1 and Section 6.3.2, respectively.

6.3.1 Consequences of the Actuator Attribute Attack. We use two popular Zigbee smart locks (Yale Assure Lock and Kwikset Smart Lock) and one Zigbee smart siren (Frient Smart Siren) to demonstrate the consequences of the actuator attribute attack. For the attributes of the two locks and the siren in Table 2, we focus on safety and security-related attributes to determine if the modified attributes could cause safety or security issues. The behaviors of the smart locks and the smart siren before and after the hidden attribute attack are listed in Table 3, where the first five rows are attributes of the Door Lock cluster, and the last row is an attribute of the IAS WD cluster. Users can configure some hidden attributes (e.g., *AutoRelockTime*, *EnableOneTouchLocking*, and *SoundVolume*) directly on a lock by physically interacting with the lock but still cannot see or configure them on smartphone apps.

Prolonging Auto Relock Time: A user sets the *AutoRelockTime* to 15 seconds to ensure the door locks automatically when the user leaves home. However, if an attacker extends this delay, the door could remain unlocked for up to 180 seconds, providing a burglar with the chance to enter the home.

Failure of One Touch Locking: One-touch Locking allows users to lock doors by tapping a key or an area of the lock. However, this feature can be disabled by stealthily setting the *EnableOneTouchLocking* attribute to "False". Victims accustomed to this feature may tap the button as usual, unaware that the door has not locked, increasing the risk of burglary. This attack can be amplified by attacking the *SoundVolume* attribute. Typically, the lock plays an acknowledgment chime after a one-touch lock to indicate success. An attacker could silence this chime by setting the *SoundVolume* attribute to 0 via a hidden attribute attack. As a result, users might assume the absence of the chime is normal and fail to realize the door is not actually locked.

PIN Code Leakage: The *SendPINOverAir* is set to "False" by default to mask the PIN code in the Zigbee payload. However, the hidden attribute attack can easily change it to "True", which makes the PIN code displayed as plaintext in the Zigbee payload. After testing this attribute on two smart locks, we find that neither smart lock includes the PIN code in the general "lock"/"unlock" events or commands. However, the Yale Assure Lock includes the PIN code in the payload of the code registration event. By setting the attribute to "True", the attacker can easily obtain the PIN code from the sniffed Zigbee packet.

Smart Siren Silencing: The *MaxDuration* attribute on a smart siren usually has a default value (e.g., 900 seconds) that makes the siren sound long enough to alert the user. However, an attacker can silence the siren by setting *MaxDuration* to 0. As a result, the siren will not sound, and the user will not be notified of any emergency.

6.3.2 Consequences of the Sensor Attribute Attack. We selected one device from each of four popular sensor types: motion, vibration, presence, and temperature. These devices are used to demonstrate the severe consequences of sensor attribute attacks and how modified sensor attributes can disrupt automation rules.

Faking Motion Active: The *PIROccupiedToUnoccupiedDelay* attribute can be set to an excessively large value (e.g., 65,534 seconds), causing the motion sensor to remain active for an extended period after detecting an initial motion event. During this time, the sensor will not report any subsequent motion events. Table 4 lists several

Table 4: Consequences of the Sensor Attribute Attack on hidden attributes and parameters.

Device Name	Cluster → Attribute	Automation Rule	Attack Method	Consequences
Philips Hue Motion Sensor	Occupancy Sensing → PIROccupied-ToUnoccupiedDelay	When presence sensor turns inactive, if motion sensor is inactive, lock the door and arm security system.	Prolong "PIROccupiedToUnoccupiedDelay" attribute so that motion sensor stays active.	Door is not locked and security system is not armed when user leaves home.
Philips Hue Motion Sensor		When detecting motion, if presence sensor is inactive, sound alarm.	Prolong "PIROccupiedToUnoccupiedDelay" attribute so that motion sensor stays active.	Burglary not detected since the motion sensor is stuck at active.
Philips Hue Motion Sensor	Occupancy Sensing → Motion Sensitivity (0x0030) (manufacturer-specific)	When detecting motion, if presence sensor is inactive, sound alarm.	Lower the sensitivity of motion sensor.	Burglary not detected due to the low sensitivity level.
SmartThings Multipurpose Sensor	Acceleration (0xfc02) (manufacturer-specific) → Vibration (0x0010) (manufacturer-specific)	When detecting vibration on the safe (the sensor is attached inside the safe), if presence sensor is not present, sound alarm and call 911.	Set "Maximum Reporting Interval" to 0xffff, so that the attribute will not be reported.	The movement of the safe by a burglar will not be reported. The alarm will not sound, and 911 will not be called, which causes a huge loss.
SmartThings Presence Sensor	Binary Input → PresentValue	When presence sensor becomes inactive, turn off humidifier, outlets, and all light bulbs.	Set "Minimum Reporting Interval" to the value larger than 100 seconds, or set "Maximum Reporting Interval" to 0xffff.	Mistakenly turn off devices when user is home, causing inconvenience and safety issues, especially at night.
SmartThings Motion Sensor (w/ Temp)	Temperature Measurement → MeasuredValue	When temperature is below 18°C, turn on the heater. When temperature is above 22°C, turn off the heater.	Set "Maximum Reporting Interval" to 0xffff, so that the attribute will not be reported.	Heater is not turned on or off, causing fire or failing to keep the room (e.g., baby room) at desired temperature.

rules affected by this hidden attribute attack, which can lead to serious security and safety issues. For example, in the first automation rule, when the presence sensor becomes inactive, the conditions are not met due to the prolonged duration of the motion active event, preventing the execution of security-critical actions (e.g., the door remains unlocked, and the security system is not armed). In the second rule, no trigger event occurs because the motion sensor stays active, ignoring all subsequent motion events. Consequently, the rule is not executed, leaving the home vulnerable to burglary.

Modifying Motion Sensitivity: The "Motion Sensitivity" in Table 2 is a manufacturer-specific attribute in the Occupancy Sensing cluster on the Philips Hue Motion Sensor, which affects detection results. Specifically, it has three different sensitivity levels (i.e., Low, Medium, and High) with a default value of "High". The detection distance at a normal walking pace varies based on the sensitivity level: 1) Low: 80 centimeters; 2) High: 200 centimeters. By changing the sensitivity level from "High" to "Low", the detection distance is shortened by more than half, which allows a burglar to avoid being detected. As detailed in Table 4, an undetected motion event prevents the execution of the rule, and the alarm will not sound.

Tampering Reporting Interval: In addition to hidden attributes, Zigbee sensors have configurable parameters such as "Minimum Reporting Interval" and "Maximum Reporting Interval" that are neither visible nor accessible to users. Attackers can exploit these hidden 16-bit parameters to alter reporting behavior using "Configure Reporting" commands. The first parameter sets the minimum interval between consecutive reporting events. If set to 10, the attribute cannot be reported again within 10 seconds of the last event. The second one defines the maximum interval. If set to 60, the sensor must report the attribute to the Zigbee coordinator within 60 seconds, even if no changes occur. If the "Maximum Reporting Interval" is set to 0xffff, the sensor stops reporting that attribute, effectively losing the ability to sense it.

In Table 4, setting the "Maximum Reporting Interval" to 0xffff prevents the SmartThings Multipurpose Sensor (*GP-U999SJVLAAA*) [16] from reporting vibration, allowing a burglar to bypass automation and move a safe undetected.

The SmartThings Arrival Sensor (*F-ARR-US-2*) [11] reports the "PresentValue" every minute. If not received within 100 seconds, the coordinator marks the sensor as inactive. By setting the "Minimum Reporting Interval" to over 100 seconds or the "Maximum Reporting Interval" to 0xffff, the sensor fails to report in time, triggering the automation in Table 4 to turn off outlets (which may connect to important electronic devices) and lights, causing safety issues, especially for the elderly at night.

The SmartThings Motion Sensor (*GP-U999SJVLBAA*) [15] detects motion and measures temperature. Setting the "Maximum Reporting Interval" to 0xffff stops temperature reports, preventing temperature-related automation from triggering, which can lead to hazards in environments requiring stable temperatures, such as a baby room, as shown in Table 4.

6.4 Z-Wave/Wi-Fi Device Hidden Attributes

Hidden attributes are not exclusive to Zigbee devices and also exist on Z-Wave and Wi-Fi devices. Here, we explore the hidden attributes of several Z-Wave and Wi-Fi devices on the SmartThings platform for demonstration purposes.

Z-Wave Devices. We explore the hidden attributes of three Z-Wave devices: Zooz 4-in-1 Motion Sensor (*ZSE40*), Aeotec MultiSensor 6 (*ZW100-A*), and Aeotec Siren Gen5 (*ZW080-A17*). Table 5 lists the hidden attributes of these devices. Some hidden attributes define thresholds for reporting changes in measurements like temperature, humidity, brightness/luminance, battery level, and ultraviolet. These attributes could be exploited by attackers. For example, an attacker could set the threshold to a minimum value (e.g., 1% for humidity), causing the device to report every minor environmental change, which would quickly drain the battery. Alternatively, setting the threshold to a maximum value (e.g., 50% for humidity) could prevent the device from reporting any measurements, even if the environment changes significantly, leading to suppressed events and malfunctioning automation rules. The Zooz 4-in-1 Motion Sensor supports several levels of sensitivity and re-trigger intervals, which are not configurable on the SmartThings platform. These attributes can be exploited similarly to the Philips Hue Motion Sensor (Table 4). Additionally, the "Lock Configuration" attribute can be toggled between "Disabled" and "Enabled" to control whether the configuration can be modified. The "Reset to Factory Default Setting" offers three options: 1) Normal operation; 2) Reset all configuration parameters to default settings; and 3) Reset the product to factory settings and exclude it from the Z-Wave network.

Table 5: Z-Wave device hidden attributes on SmartThings.

	Attribute	Description
Zooz 4-in-1 Motion	Temperature Scale	Current temperature unit: Fahrenheit or Celsius
	Temperature Reporting Threshold	Temperature change to be reported in degrees
	Humidity Reporting Threshold	Humidity change to be reported in percentages
	Brightness Reporting Threshold	Brightness change to be reported in percentages
	Motion Re-trigger Interval	Time interval of motion being reported again after the initial trigger
	Motion Sensitivity	Different levels of motion sensitivity
	LED Indicator Mode	LED mode to indicate motion/temperature events
Aeotec MultiSensor 6	Low Battery Threshold	Threshold to indicate that battery level is low
	Temperature Threshold Unit	Temperature unit used for temperature threshold
	Temperature Change Threshold	Temperature change to be reported in degrees
	Humidity Change Threshold	Humidity change to be reported in percentages
	Luminance Change Threshold	Luminance change to be reported in lux
	Battery Level Threshold	Battery change to be reported in percentages
	Ultraviolet Change Threshold	Ultraviolet change to be reported
Aeotec Siren	Send Notifications to Associated Devices	Signal other associated devices within the association group when alarm sounds
	Partner ID	Manufacturer name
	Lock Configuration	Binary-valued attribute to indicate whether the configuration can be changed
	Reset to Factory Default Setting	Different operations during reset

Wi-Fi Devices. Both Wemo and Sonos have Wi-Fi devices supported on the SmartThings platform via edge drivers. We explore the hidden attributes of a Wemo Mini Smart Plug (*F7C063*) [19] and a Sonos One Gen2 Speaker (*ONEG2US1BLK*) [18]. Existing edge drivers for these two devices support only basic functions, but the devices themselves support additional attributes listed in Table 6. The hidden attributes of the Wemo plug mainly provide device information, such as MAC, serial number, firmware version, and port number, which are not available on the SmartThings platform. For the Sonos One Speaker, the hidden attributes are related to control commands: "lockVolumes" fixes the speaker volume, "unlockVolumes" allows volume changes, "timeseek" sets a specific time in the current track, "trackseek" switches to a target track by index, and "repeat" loops over one or all tracks.

6.5 Evaluation Summary

All attack consequences listed in Table 3 and Table 4 are successfully verified and are consistent with the physically observed results. From the experiments, ***we confirm that the hidden attributes are ubiquitous and exist on all the Zigbee, Z-Wave and Wi-Fi devices that we tested***. Many of these attributes are identified on safety-critical devices, such as locks and motion sensors, and can be exploited by attackers to incur severe risks. To understand platform-level differences in vulnerability, we conduct comprehensive experiments across various IoT devices, platforms, and communication protocols, with a focus on both hidden and observable attributes. Our findings reveal that Amazon Alexa is more susceptible to hidden attributes compared to the SmartThings platform. This is because: 1) The Amazon Alexa platform provides only basic support for IoT devices, lacking many key attributes. 2) The SmartThings platform open-sources its edge driver codes and hosts a community forum, encouraging developer participation and contributions. This leads to more frequent updates and improvements in integration through edge drivers.

Table 6: Wi-Fi device hidden attributes on SmartThings.

	Attribute	Description
Wemo Mini	GetNetworkList	Retrieve a list of nearby Wi-Fi networks
	GetHomeId	Retrieve the Home ID
	GetMacAddr	Retrieve MAC address, Serial Number, and Plugin UDN (Unique Device Name)
	ChangeFriendlyName	Change device name
	GetMetaInfo	Retrieve meta information of device: MAC address, serial number, device type, firmware version
	GetDeviceInformation	Retrieve device information: MAC address, port number, firmware version, friendly name
Sonos One	lockVolumes	Disable volume change
	unlockVolumes	Allow volume change
	timeseek	Go to a specific position in the current track
	trackseek	Go to a specific track in the current queue
	queue	Display all tracks in the queue
	clearqueue	Remove all tracks in the queue
	repeat	Repeat all/one/none tracks

6.6 Generality of the Hidden Attribute Problem

The hidden attribute problem exists in multiple IoT platforms and protocols. While our evaluation primarily focuses on two popular commercial platforms, SmartThings and Alexa, the same principles of discovering hidden device attributes and/or inadequate oversight in device drivers are broadly applicable to other IoT home automation frameworks such as Google Home and Apple HomeKit. These two platforms define standardized sets of attributes for each device type, i.e., Google Home *traits* [36] and Apple HomeKit *characteristics* [28], which may not include some attributes supported by IoT devices, making them hidden from the platforms. For example, we find that the attribute "clearqueue" of Sonos One is hidden on both Google Home and Apple HomeKit. While our experiments study Zigbee, Z-Wave, and Wi-Fi, the hidden attribute issue also applies to Bluetooth Low Energy (BLE). BLE devices expose functionality via the Generic Attribute Profile (GATT), where data is structured into services and characteristics. Although standardized profiles exist, vendors frequently implement proprietary services using custom UUIDs, which are often undocumented or inconsistently supported across platforms [33]. Even in emerging protocols such as Matter, which aim to improve interoperability through strict device models and standardized data schemas, vendors are still allowed to define optional or manufacturer-specific attributes. In essence, as long as there is a separation between the IoT device functionality and the automation platform, which is usually bridged by IoT device drivers, the possibility of hidden attributes exists. In summary, our findings indicate that the vulnerability widely exists in multiple IoT platforms, IoT devices, and communication protocols.

7 Countermeasures

The hidden attribute vulnerability is widespread in IoT devices and poses serious safety and security risks. Addressing this issue faces several challenges. First, it is unrealistic to expect all IoT manufacturers to invest in hardening edge drivers, as they often prioritize new features over security, leaving vulnerabilities unpatched for years [29, 37, 39]. Second, it is impractical for home automation platforms like SmartThings to resolve this alone, as obtaining comprehensive lists of supported attributes across thousands of devices

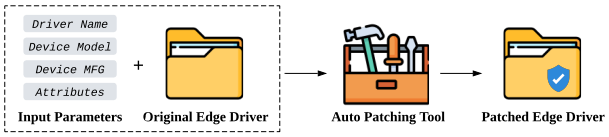

Figure 5: Automatic Patching for Edge Drivers.

requires extensive collaborations with device manufacturers. Even if home automation platforms try to solve the hidden attribute issue, they have to go through a similar procedure as we do for each device (i.e., obtaining all attributes and then patching edge drivers). As of the submission of this manuscript, SmartThings and other IoT platforms have not taken any action after we disclosed the vulnerability to them. Third, most smart home users lack the technical expertise to identify hidden attributes on their devices.

Automatic Patching for Edge Drivers. To address the hidden attribute issue, we propose an innovative countermeasure: automatic patching for IoT edge drivers. We develop an auto-patching tool that takes the input of existing vulnerable edge driver code along with the device attribute scanning results, as outlined in Section 4. The tool identifies hidden attributes not covered by the original driver and then automatically patches the driver with additional code and configuration files to display these attributes. The tool works autonomously and can be easily used to address hidden attribute issues. Besides making hidden attributes visible and configurable in smartphone apps, the tool can also monitor changes in hidden attributes and notify users when they are modified. In the following, we detail the design of the automatic patching approach, which has been implemented on the SmartThings platform.

Custom Capabilities and Presentations. In the first step, custom capability [13] and presentation [14] need to be created for each hidden attribute. Capabilities define IoT device features on the SmartThings platform, and presentations determine how these capabilities appear in the SmartThings app. Each custom capability is assigned a unique ID in the format `namespace.capabilityName`, where `namespace` is randomly assigned by the SmartThings platform and `capabilityName` is defined by the developer. Creating custom capabilities and presentations is a **one-time effort**, acting as preparation for our proposed module. Once created, these capabilities can be accessed by anyone via the capability IDs in edge drivers. Therefore, users of the auto-patching tool do not need to handle this technical aspect.

Auto-Patching Tool. The auto-patching tool contains Python scripts that are enclosed in a shell script. The overall size is less than 200 KB. As shown in Figure 5, the inputs used for the auto-patching tool include the original edge driver folder and four parameters: 1) *Driver Name*: The name of the edge driver to be patched; 2) *Device Model*: The device model the patched driver will be used for; 3) *Device MFG*: The device manufacturer; and 4) *Attributes*: The list of hidden attributes to be patched for the particular device. The output is the patched edge driver that is ready for use and can be directly installed onto the SmartThings hub.

To simplify obtaining the required parameters, we provide a list of *Driver Names* and corresponding *Attributes* supported by the auto-patching tool[2]. It is built based on the devices tested and hidden attributes discovered in this work. We have already created custom capabilities and presentations for the attributes listed, so users can use them directly. The *Device Model* and *Device MFG* can be easily found in the SmartThings Advanced Web App [8].

The auto-patching tool is easy to use and effective. With basic inputs (i.e., the original edge driver and parameters), users only need to run a one-line command to generate the patched edge driver, which can be directly installed on the home automation platform (e.g., SmartThings). The patched attributes then become visible and configurable on the platform. For example, to patch an edge driver named `zigbee-lock` for the Yale Door Lock (Model: YRD226 TSDB, MFG: Yale) with attributes (Language, AutoRelockTime), a user simply runs `./auto_patch.sh zigbee-lock "YRD226 TSDB" Yale Language:AutoRelockTime`, where attributes are separated by ":". Alternatively, specifying "ALL" for the attribute field patches all supported attributes associated with edge drivers. Once the patched driver is installed on a SmartThings hub, the patched attributes are displayed in the SmartThings app and can be controlled by the user.

Monitoring Patched Attributes. After installing the patched driver, previously hidden attributes become visible/accessible in the SmartThings app, appearing as capability names on the UI that can be easily mapped to the patched attributes by users. We further improve it by enabling automatic notification upon changes to safety-critical attributes. For this purpose, we define the "Automation View" [12] in the presentations during custom capability creation, making them configurable in automation routines. These attributes are then set as triggers in automation routines. For example, routine "Send a text message to the user if AutoRelockTime is equal to or above 30 seconds" is triggered when the value of AutoRelockTime exceeds 30 seconds, and the user will receive a text notification about this abnormal attribute setting.

Countermeasures from Different Parties. Different parties within a smart home IoT ecosystem could address the hidden attribute issue using the auto-patching tool. 1) *End Users:* End users own the physical devices and can scan them to identify attributes, then use the auto-patching tool to address any hidden attributes. However, this approach has a drawback as users may lack the technical expertise to run shell scripts. 2) *Platform Owners:* Platform owners maintain edge drivers and can gather feedback from forums and users about missing device attributes. They can then use the auto-patching tool to fix these gaps, update, and publish the revised edge drivers. 3) *Device Manufacturers:* Device manufacturers, with full knowledge of their devices' supported attributes, can either use the auto-patching tool to correct edge drivers or provide a complete attribute list to platform owners. In this way, the platform owners can use the auto-patching tool to patch missing attributes for devices from the particular device manufacturer. 4) *Edge Driver Developers:* Edge driver developers, who typically need the device on hand for testing, can first scan for all device attributes and then use the auto-patching tool to add any missing attributes to the edge driver. *5) Protocol Level:* Protocols can be enhanced to prevent hidden attribute attacks by adopting application-level encryption for sensitive clusters, avoiding global keys, and disabling insecure joins/rejoins. Moreover, IEEE 802.15.4 MAC layer encryption is crucial for securing the MAC layer header information, but it comes at the cost of significantly reducing the battery life of IoT devices.

[2]https://anonymous.4open.science/r/SmartThings-Edge-Driver-Auto-Patching-49CE

For other home automation platforms that do not allow users to access edge drivers' code (e.g., Amazon Alexa), smart home users or third parties cannot directly leverage our auto-patching tool to address the hidden attribute issue. However, the home automation platforms (e.g., Amazon Alexa) have access to the edge drivers' code, and they can use our auto-patching tool to identify and resolve the hidden attribute issue. Additionally, they can host some crowdsourcing platforms and encourage skilled users or third parties to scan their devices and share their results. This collaborative effort would enable platforms to compile comprehensive lists of supported attributes, facilitating the effective use of the auto-patching tool.

8 Lessons Learned

The hidden attribute attack can impact a significant number of users and devices. Through this work, we aim to stimulate awareness, promote security best practices, and thus further enhance IoT security. To further elaborate on the lessons learned from this research, here we discuss and answer two questions: *1) What are the factors (vulnerabilities and security issues) that collectively caused this dangerous yet stealthy attack? 2) What cybersecurity best practices should be adopted to prevent such issues in the future?*

8.1 Vulnerabilities and Security Issues

The hidden attribute attack is caused by several inherent system vulnerabilities across different aspects, as articulated below:

1) Insecure Join/Re-join [Protocol]. The Zigbee protocol has recommended secure join/re-join procedures, unfortunately, the secure join/re-join is not mandatory. To support legacy devices, many Zigbee networks still allow devices to join with weak/no authentication during commissioning using the well-known default link key (ZigbeeAlliance09 [23]), which makes it easy for a malicious device to join a Zigbee network and then obtain the network key, which is protected by the default link key during the key distribution.

2) Trust Assumption [Network and Protocol]. Devices in the same Zigbee network are designed to communicate with each other to enable home automation functions. A Zigbee network inherently assumes that all devices within the network are trustworthy, so there is no access control at the network layer of the protocol when one device accesses another in the same Zigbee network. A malicious device in the Zigbee network could eavesdrop on communications and access legitimate Zigbee devices.

3) Lack of Endpoint Authentication and Access Control [Endpoint]. The current Zigbee network lack sufficient authentication and fine-grained access control, leaving vendors unable to protect their devices and attributes from malicious activity. Vendors inherit this trust model and typically omit security measures such as authentication, access control, or intrusion detection in their network APIs. As a result, sensitive and security-critical attributes are left unprotected and can be modified by any device in the network. In contrast, Z-Wave [21] defines three security levels and enforces isolation between them. However, its coarse-grained model only blocks cross-level attacks and devices within the same level can still compromise each other due to implicit trust.

4) Loose Protocol [Protocol]. The Zigbee protocol lacks clear definitions for device behaviors and attributes, allowing vendors to implement custom, potentially hidden attributes outside the ZCL. Compounding this, Zigbee does not require home hubs to support all mandatory attributes, including security-critical ones. Since the hub often serves as the only interface for users to interact with devices, unsupported attributes become hidden and vulnerable.

5) Inherent Risks of Abstraction [Software Security]. Abstraction simplifies complexity but can obscure internal states, as seen in Heartbleed [34], where hidden implementation details lead to exploitation. In IoT ecosystems, edge drivers abstract device functionalities into platform-compatible interfaces, but incomplete mappings can hide critical attributes from the user. This risk is amplified by Zigbee's loosely defined protocol, which allows vendors to implement manufacturer-specific attributes. For example, security-critical attributes like *AutoRelockTime* or *MotionSensitivity* are often omitted from platform integrations, even though they are implemented and used internally by the devices.

6) Lack of User Alert [UI/UX]. When device attributes, including security-critical attributes, are modified, the current Zigbee system does not mandate a notification mechanism to alert the user of the change, which makes the attack stealthy. In fact, for the Zigbee devices we tested, none alert the user when a device attribute is modified by another channel, e.g., by another device in the network.

8.2 Mitigation and Defense

The above vulnerabilities and questionable designs in various aspects collectively made the hidden attribute attack feasible, stealthy, and applicable to many IoT devices regardless of their vendors. We argue that the security of all aspects should be enhanced.

Transparency and Disclosure. A secure solution starts with mandatory attribute disclosure during device certification, enforced via protocol extensions (e.g., ZCL) or compliance frameworks (e.g., Matter). Although Matter advances standardization and visibility, it still permits vendor-specific extensions that can hide attributes. Platforms should adopt transparency-by-design, logging all attributes even if not user-facing. However, challenges remain due to the fragmented IoT ecosystem, where manufacturers resist standardization, and economic incentives prioritize speed over security. Addressing these issues requires balancing between usability and transparency.

Security in Software Development. The reliance on obscurity as a security mechanism in software development has repeatedly failed in traditional contexts, as seen in exploits of proprietary software where attackers reverse-engineer hidden logic. In IoT, the assumption that invisible attributes are inherently secure creates a false sense of safety. A proactive defense involves adopting an assume-breach mindset, treating all attributes as potential attack surfaces. Automated auditing tools, such as the one we presented in Section 7, could be integrated into development pipelines to flag unmapped attributes during driver development. However, billions of legacy devices lacking updatable firmware remain vulnerable, and continuous auditing may strain resource-constrained devices. This highlights the need for security-by-default designs.

Principle of Least Privilege in Attribute Access. The principle of least privilege, a cornerstone of traditional security, is often neglected in IoT attribute management. Attribute-level access control by enforcing cryptographic consent or device authentication can protect safety-critical settings, while logging attribute changes enhances accountability. However, strict controls may hinder usability

and create interoperability issues across platforms. A balanced approach could involve adaptive frameworks with access tiers (e.g., "basic" vs. "critical") to align security needs with user convenience.

Protocol-level Defense Mechanisms. Encryption and authentication are inconsistently applied in IoT. Zigbee's use of a shared network key allows attackers to decrypt traffic after infiltration. Trust at the network level allows attackers to eavesdrop and manipulate attributes once inside. A stronger approach would use per-cluster encryption with unique keys for sensitive functions and secure commissioning with pre-provisioned keys. However, legacy compatibility and resource limits hinder adoption. Protocols should embed upgrade paths for cryptographic agility, while manufacturers phase out insecure legacy practices.

Security in All Design and Development Aspects. The lack of user alerts in IoT systems, where attribute changes go unnoticed, highlights the importance of user-centric security. Future platforms should enforce real-time notifications for modifications of critical attributes, akin to intrusion detection systems in enterprise networks. Cross-disciplinary collaboration is also critical: device manufacturers, protocol standards bodies, and platform owners should align incentives to prioritize transparency. Regulators could mandate attribute transparency reports, while academia develops open-source tools for community-driven audits. By fostering a culture of proactive scrutiny, the hidden vulnerabilities can be transformed from systemic weaknesses into managed risks.

9 Related Work

This section reviews the literature on vulnerability discovery and attacks in Zigbee networks. Several studies evaluate the security of the Zigbee protocol design. Zigator [25] develops a penetration testing tool for Zigbee networks, assessing interactions against passive and proactive attacks such as sniffing, injection, and jamming. Z3Sec [47] examines the security of the touchlink commissioning procedure, exposing an insecure design that enables device impersonation attacks. Snout [46] is a network mapping and penetration testing tool for automatic wireless exploitations, including device enumeration, vulnerability assessment, and packet sniffing/replay/spoofing. Wang et al. [56] analyze the key leakage problem caused by the insecure Zigbee network key distribution. Verejoin [55] highlights vulnerabilities in Zigbee's device rejoining procedure, which can result in connection hijacking and impact numerous devices. Tamarin [43] explores the formal verification of the Zigbee protocol stack. ZLeaks [51] introduces a tool to passively identify devices and events from encrypted Zigbee traffic based on device reporting patterns and intervals.

Some existing works examine vulnerabilities in the implementation of Zigbee devices. Z-Fuzzer [49] employs fuzzing to find new vulnerabilities caused by improper input validation. [3] provides a security testing tool for developers to evaluate the software implementation of their own devices. A Zigbee chain reaction is created in [50] by exploiting the vulnerabilities in the Zigbee Light Link protocol, potentially resulting in a city-wide blackout or a massive DDoS by exploiting the compromised devices. A recent work [57] demonstrates an attack without knowing Zigbee encryption keys and shows that Zigbee devices can be put into an abnormal status by attacking Zigbee packets with delicately crafted payloads. Remote Attention attack [54] provides the possibility of reconfiguring and disconnecting IoT sensors from the network.

Denial of Service (DoS) attacks can target Zigbee networks. A Low-Rate DoS (LDoS) attack [48] exploits Zigbee's routed transmission and throttles the packet rate to 0%. Energy depletion attacks (EDAs) are also applicable to Zigbee networks [26, 30]. In HiveGuard [26], an external attacker depletes the energy of battery-powered commercial Zigbee devices in less than 16 hours by interfering with the transmission of Data Requests to send crafted acknowledgments. Ghost-in-ZigBee [30] examines vulnerabilities in the IEEE 802.15.4 security suites that can be exploited in energy depletion attacks, leading to denial of service and replay attacks.

Unlike prior work on IoT protocol design and implementation security, our work identifies a new vulnerability caused by discrepancies between IoT devices and home automation platforms, an overlooked issue by both device manufacturers and platforms.

10 Conclusion

In this work, we revealed a new vulnerability – the hidden attribute issue – which is inherent in most home automation platforms and is ubiquitous in IoT devices with different wireless technologies (Zigbee, Z-Wave, and Wi-Fi). We systematically analyzed the root causes of this issue and developed an effective method for discovering hidden attributes in commercial smart home IoT devices. We conducted experiments on over 30 commercial smart home IoT devices of different types from 16 manufacturers and discovered a total of 119 hidden attributes. We presented a new attack – the hidden attribute attack – which exploits the vulnerability to modify hidden attribute values without being noticed by users, potentially leading to severe security and safety risks, such as burglary or fire. We responsibly disclosed the risks to several major IoT manufacturers and received acknowledgments. We hope this work raises awareness in the community to strengthen the security of widely used smart home IoT devices. In future work, we plan to extend our experiments to more Z-Wave and Wi-Fi devices.

Acknowledgments

This work was supported in part by NSF under grants CNS-2204785, CNS-2205868, 2409212, IIS-2014552, and DGE-1565570. Additional support was provided by the Ripple University Blockchain Research Initiative and the University of North Carolina System Research Opportunities Initiative.

References

[1] 2023. achingbrain-ssdp. https://github.com/achingbrain/ssdp. (Accessed on 01/24/2024).
[2] 2023. Addressing. https://www.digi.com/resources/documentation/Digidocs/90002002/Concepts/c_zb_addressing.htm. (Accessed on 02/03/2023).
[3] 2023. Beyond Security - ZigBee. https://www.beyondsecurity.com/dynamic-fuzzing-testing-zigbee?cn-reloaded=1. (Accessed on 02/06/2023).
[4] 2023. codingjoe-ssdp. https://github.com/codingjoe/ssdp. (Accessed on 01/24/2024).
[5] 2023. Frient Smart Siren. https://frient.com/products/smart-siren/. (Accessed on 10/15/2023).
[6] 2023. Kwikset 914 Smart Lock. https://www.amazon.com/Kwikset-SmartCode-Electronic-Featuring-Technology/dp/B08XNYSNT7?th=1. (Accessed on 10/15/2023).
[7] 2023. Manufacturer-specific attribute. https://github.com/Koenkk/zigbee-herdsman-converters/blob/master/devices/philips.js#L1919. (Accessed on

01/26/2023).
[8] 2023. New SmartThings Advanced Web App. https://my.smartthings.com/. (Accessed on 01/05/2024).
[9] 2023. PAN ID. https://www.digi.com/resources/documentation/Digidocs/90002002/Concepts/c_zb_pan_id.htm. (Accessed on 02/03/2023).
[10] 2023. Philips Hue Motion Sensor. https://www.philips-hue.com/en-us/p/hue-motion-sensor/046677570972?origin=71700000112071407&gclid=CjwKCAjw-KipBhBtEiwAWjgwrNYNPZMt3VQZ4_SQj7fCd2g5QSH_qNBHNE0DfRGA1hJ8jaBYAQZL3xoCMkQQAvD_BwE&gclsrc=aw.ds#overview. (Accessed on 10/15/2023).
[11] 2023. SmartThings Arrival Sensor. https://www.amazon.com/Samsung-SmartThings-F-ARR-US-2-Arrival-Sensor/dp/B00GM7V8I8. (Accessed on 10/15/2023).
[12] 2023. SmartThings Developers - Automation View. https://developer.smartthings.com/docs/devices/capabilities/capability-presentations#automation-view. (Accessed on 01/04/2024).
[13] 2023. SmartThings Developers - Capabilities. https://developer.smartthings.com/docs/devices/capabilities/. (Accessed on 01/03/2024).
[14] 2023. SmartThings Developers - Presentations. https://developer.smartthings.com/docs/devices/configurations-and-presentations/device-presentations. (Accessed on 01/03/2024).
[15] 2023. SmartThings Motion Sensor. https://www.amazon.com/Samsung-SmartThings-GP-U999SJVLBAA-Optional-Automated/dp/B07F8ZHBLS/ref=asc_df_B07F8ZHBLS/?tag=hyprod-20&linkCode=df0&hvadid=241967399507&hvpos=&hvnetw=g&hvrand=15572158658243508334&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9003565&hvtargid=pla-528889063635&psc=1. (Accessed on 10/15/2023).
[16] 2023. SmartThings Multipurpose Sensor. https://www.amazon.com/Samsung-SmartThings-Multipurpose-Sensor-GP-U999SJVLAAA/dp/B07F956F3B?th=1. (Accessed on 10/15/2023).
[17] 2023. sonoff Zigbee 3.0 dongle. https://sonoff.tech/product/gateway-and-sensors/sonoff-zigbee-3-0-usb-dongle-plus-p/. (Accessed on 01/26/2023).
[18] 2023. Sonos One Gen2 Speaker. https://www.bestbuy.com/site/sonos-one-gen-2-smart-speaker-with-voice-control-built-in-black/6333557.p?skuId=6333557. (Accessed on 01/18/2024).
[19] 2023. Wemo Mini Smart Plug. https://www.amazon.com/Smart-Enabled-Google-Assistant-HomeKit/dp/B01NBI0A6R?th=1. (Accessed on 01/18/2024).
[20] 2023. Yale Assure Lock with Zigbee. https://www.amazon.com/Yale-Touchscreen-Deadbolt-YRD226HA2619-SmartThings/dp/B072LF66YX/ref=sr_1_2?crid=116YIX7V71LRC&keywords=YRD226-HA2-619&qid=1697385837&s=hi&sprefix=yrd226-ha2-619%2Ctools%2C75&sr=1-2&th=1. (Accessed on 10/15/2023).
[21] 2023. Z-Wave. https://www.z-wave.com/. (Accessed on 10/17/2023).
[22] 2023. Z-Wave JS. https://github.com/zwave-js. (Accessed on 01/21/2024).
[23] 2023. ZIGBEE SECURITY: BASICS. https://research.kudelskisecurity.com/2017/11/08/zigbee-security-basics-part-2/#:~:text=A%20default%20global%20trust%20center,at%20the%20time%20of%20joining. (Accessed on 02/06/2023).
[24] 2024. Official Yale Lock Document. https://www.homecontrols.com/homecontrols/products/pdfs/YA-Yale/YAYRD256HA2x-User-Manual.pdf. (Accessed on 03/12/2024).
[25] Dimitrios-Georgios Akestoridis, Madhumitha Harishankar, Michael Weber, and Patrick Tague. 2020. Zigator: Analyzing the security of zigbee-enabled smart homes. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 77–88.
[26] 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)*. IEEE, 209–217.
[27] Zigbee Alliance. 2023. Zigbee Cluster Library. https://zigbeealliance.org/wp-content/uploads/2021/10/07-5123-08-Zigbee-Cluster-Library.pdf. (Accessed on 01/05/2023).
[28] Apple. 2025. Characteristic types. https://developer.apple.com/documentation/homekit/characteristic-types. (Accessed on 04/03/2025).
[29] Ed Bott. 2024. The future may be passwordless, but it's not here yet. https://www.zdnet.com/article/the-future-may-be-passwordless-but-its-not-here-yet/. (Accessed on 01/25/2024).
[30] 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.
[31] John Carlsen. 2023. Outfitting Your Smart Home: Zigbee Devices. https://www.safewise.com/zigbee-devices/#:~:text=What%20is%20the%20range%20of,of%20250%E2%80%93325%20feet%20indoors.&text=Without%20obstructions%2C%20Zigbee%20frequencies%20can,by%20using%20a%20signal%20repeater. (Accessed on 10/17/2023).
[32] Haotian Chi, Chenglong Fu, Qiang Zeng, and Xiaojiang Du. 2022. Delay wreaks havoc on your smart home: Delay-based automation interference attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 285–302.
[33] davidgyoung. 2021. Stackoverflow - Where to find Service Data UUIDs? https://stackoverflow.com/a/57718856/16852539. (Accessed on 04/04/2025).
[34] Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, et al. 2014. The matter of heartbleed. In *Proceedings of the 2014 conference on internet measurement conference*. 475–488.
[35] Jonathan D Fuller and Benjamin W Ramsey. 2015. Rogue Z-Wave controllers: A persistent attack channel. In *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*. IEEE, 734–741.
[36] Google. 2025. Smart Home Device Traits. https://developers.home.google.com/cloud-to-cloud/traits. (Accessed on 04/03/2025).
[37] Tatum Hunter. 2021. Buggy software in off-brand smart home devices is a hacker's playground. https://www.washingtonpost.com/technology/2021/11/18/smart-home-security/. (Accessed on 01/25/2024).
[38] Yan Jia, Bin Yuan, Luyi Xing, Dongfang Zhao, Yifan Zhang, XiaoFeng Wang, Yijing Liu, Kaimin Zheng, Peyton Crnjak, Yuqing Zhang, et al. 2021. Who's in control? On security risks of disjointed IoT device management channels. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 1289–1305.
[39] Kevin Jones. 2018. Unpatched Home Routers and IoT Devices, A Tragedy Waiting To Happen. https://www.hackercombat.com/unpatched-home-routers-and-iot-devices-a-tragedy-waiting-to-happen/. (Accessed on 01/25/2024).
[40] Koenkk. [n. d.]. Zigbee2MQTT. https://www.zigbee2mqtt.io/. (Accessed on 01/05/2023).
[41] Joo Kyung-don. 2020. Over 110 mln people have downloaded Samsung's IoT app: exec. https://en.yna.co.kr/view/AEN20200108006700320.
[42] Silicon Labs. 2021. Zigbee 3.0 Device Interoperability with Legacy ZigBee Devices. https://community.silabs.com/s/article/zigbee-3-0-device-interoperability-with-legacy-zigbee-devices?language=en_US. (Accessed on 04/12/2024).
[43] Li Li, Proyash Podder, and Endadul Hoque. 2020. A formal security analysis of ZigBee (1.0 and 3.0). In *Proceedings of the 7th Symposium on Hot Topics in the Science of Security*. 1–11.
[44] Jannik Lindner. 2023. Alexa Statistics: Market Report & Data. https://gitnux.org/alexa-statistics/. (Accessed on 04/12/2024).
[45] Anna Maria Mandalari, Hamed Haddadi, Daniel J Dubois, and David Choffnes. 2023. Protected or porous: A comparative analysis of threat detection capability of IoT safeguards. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3061–3078.
[46] John Mikulskis, Johannes K Becker, Stefan Gvozdenovic, and David Starobinski. 2019. Snout: an extensible IoT pen-testing tool. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2529–2531.
[47] Philipp Morgner, Stephan Mattejat, Zinaida Benenson, Christian Müller, and Frederik Armknecht. 2017. Insecure to the touch: attacking ZigBee 3.0 via touchlink commissioning. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 230–240.
[48] 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)*. IEEE, 246–251.
[49] Mengfei Ren, Xiaolei Ren, Huadong Feng, Jiang Ming, and Yu Lei. 2021. Z-Fuzzer: device-agnostic fuzzing of Zigbee protocol implementation. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 347–358.
[50] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. 2017. IoT goes nuclear: Creating a ZigBee chain reaction. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 195–212.
[51] Narmeen Shafqat, Daniel J Dubois, David Choffnes, Aaron Schulman, Dinesh Bharadia, and Aanjhan Ranganathan. 2022. Zleaks: Passive inference attacks on Zigbee based smart homes. In *Applied Cryptography and Network Security: 20th International Conference, ACNS 2022, Rome, Italy, June 20–23, 2022, Proceedings*. Springer, 105–125.
[52] SmartThings. 2024. SmartThings APIs. https://developer.smartthings.com/docs/api/public. (Accessed on 08/31/2024).
[53] SmartThings. 2024. SmartThings Edge Driver. https://developer.smartthings.com/docs/devices/hub-connected/get-started. (Accessed on 08/31/2024).
[54] Ivan Vaccari, Enrico Cambiaso, and Maurizio Aiello. 2017. Remotely exploiting at command attacks on zigbee networks. *Security and Communication Networks* 2017 (2017).
[55] Jincheng Wang, Zhuohua Li, Mingshen Sun, and John CS Lui. 2022. Zigbee's Network Rejoin Procedure for IoT Systems: Vulnerabilities and Implications. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*. 292–307.
[56] 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 *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 133–143.
[57] 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*. 2857–2870.
[58] Yucy. 2023. Zigbee Range: You Must Know The Truth. https://reolink.com/blog/zigbee-range/#:~:text=Indoors%2C%20ZigBee%20typically%20manages%20multiple,reach%20to%20about%20300%20meters. (Accessed on 10/17/2023).