

Article

Security Analysis of Low-Budget IoT Smart Home Appliances Embedded Software and Connectivity

Kacper Murat *, Dominik Topyła, Krzysztof Zdulski, Michał Marzęcki, Jędrzej Bieniasz , Daniel Paczesny and Krzysztof Szczypiorski 

Institute of Telecommunications, Warsaw University of Technology, 00-665 Warsaw, Poland

* Correspondence: kacper.murat.stud@pw.edu.pl

Abstract: This paper investigates the challenge of finding and analyzing security vulnerabilities among widely available low-budget Internet of Things smart home appliances. It considers the identification of security vulnerabilities within the appliances' embedded software and connectivity functions over wired and wireless channels in local networks and external communications with manufacturers' cloud services. To analyze the security of these appliances, a universal laboratory test bench is proposed and a set of methodologies for testing the security of smart home devices is described. The proposed testing platform offers a practical solution for security analysis of Internet of Things smart home devices and it can serve as a reference approach for future research. The results from the research indicated varying levels of susceptibility across different types of devices. A list of recommendations for manufacturers and others to improve the security level of these appliances is provided. The findings emphasize the need for regular security assessments of smart home devices, to maintain the protection of personal and sensitive information.

Keywords: cybersecurity; Internet of Things; smart home; firmware extraction; network security; security assessment



Citation: Murat, K.; Topyła, D.; Zdulski, K.; Marzęcki, M.; Bieniasz, J.; Paczesny, D.; Szczypiorski, K. Security Analysis of Low-Budget IoT Smart Home Appliances Embedded Software and Connectivity. *Electronics* **2024**, *13*, 2371. <https://doi.org/10.3390/electronics13122371>

Academic Editor: Cheonshik Kim

Received: 9 March 2024

Revised: 10 June 2024

Accepted: 13 June 2024

Published: 17 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Various commercial and open-source technologies and systems of the Internet of Things (IoT) paradigm have seen a marked increase in attention over the past decade [1]. This can be attributed to the increasing demand for the automation of economic and social processes and their integration with the Internet. The IoT relies on the usage of devices that can communicate with others to create a network with a certain level of smartness, so that they are capable of making automatic decisions based on pre-designed functions or by responding to user actions. The unlimited possibilities in selecting devices and methods of exchanging information implies significant potential in various sectors, such as healthcare [2,3], the automotive industry [4,5], and smart buildings [6]. Moreover, there is a growing demand for IoT solutions in the consumer sector. The proliferation of low-cost solutions has led to the emergence of numerous programmable platforms and components that offer the necessary functions as nodes in the IoT network [7,8].

Given the development and adoption of smart solutions, there is a general need for efficient management of these devices, to facilitate configuration and control, for instance in smart homes and buildings. Various commercial and open-source technologies and systems exist for managing IoT devices, thereby allowing for the implementation of specific operational scenarios. An example of such automation could be a sensor system in a building that turns on the lighting in a certain sector for a defined time upon detecting movement. The aforementioned system requires an element to manage the individual components, allowing for the addition, removal, configuration, monitoring, and updating of devices. Some systems are connected to the management element through a wired connection (e.g., the KNX platform) or wirelessly (e.g., the Home Assistant platform).

The described practical scenario illustrates how IoT systems bring about the convergence of the physical and digital world. Therefore, the term cyber-physical systems (CPS) [9] is increasingly used to refer to digital hardware–software solutions that understand the physical world through sensors to make automatic decisions. Then, such decisions trigger specific control actions to impact both the digital and physical worlds.

With the dissemination of such implementations of practical IoT applications, the cybersecurity of these systems and their interactions becomes an important issue. In recent years, there have been many spectacular attacks on IoT systems in various domains of the economy, e.g., industrial cameras [10], airports [11,12], and hospitals [13]. The general security issues of smart homes and buildings have been discussed in various studies [14–17]. This topic is described in more detail in Section 3. The context of security and privacy in automated IoT scenarios has also been raised by several studies. In [18], it was discovered that the Samsung SmartThings system can generate 162 hidden inter-app interactions, among which 37 were recognized as posing a high risk of being exploited and thus the IoT security being compromised. Refs. [19,20] explored solutions for mitigation of any attacks targeting these automation chains in IoT systems. Ref. [19] applied model-based threat detection by finding anomalies within the dynamically modeled finite state machine (FSM) of automated scenarios. Ref. [20] considered detection of the abuse of security in automation scenarios configured with natural language solutions like IFTTT and Zapier. Then, natural language processing algorithms were utilized to find anomalies in these rules, to expose those which violated aspects of security and privacy.

This article presents three main contributions from the authors' research team in the field of the cybersecurity of IoT solutions in the smart home environment:

1. The design and implementation of a laboratory environment for research on the cybersecurity of IoT systems, which integrates research capabilities in the layers of firmware, wired, and wireless communication. The main contribution of this part is the reference design of such a laboratory and guidance for building it using low-cost components.
2. The development of a preliminary methodology for testing the firmware, wired, and wireless communication of selected representatives of IoT devices for testing. The main contribution of this part is the reference methodology and guidance for security analysis of IoT devices. This can be used by other teams and security researchers.
3. Carrying out theoretical analyses and practical tests on the security of the indicated IoT devices. The main contribution of this part is the quantitative results of the security evaluation of the similarities among the broad scope of IoT devices. Then, it summarizes the IoT cyber threat model and provides recommendations for IoT vendors.

The research activities cover both the endpoint of smart home devices and their embedded software, as well as wired and wireless communication. This research also included both theoretical and practical analyses of the security of these smart home devices. The results of this research will provide valuable insights into the current state of cybersecurity in smart home devices and can be used to inform the development of more secure smart home devices in the future. The testing environment developed in this research allows for a comprehensive examination of smart home device security, by enabling research on firmware, wired, and wireless communication at the device level, thereby providing a valuable contribution to the ongoing efforts to improve the security of smart home devices and IoT systems in general. Research activities related to IoT systems relate to two categories of endpoints:

- Standalone smart home systems with built-in Wi-Fi modules that work directly with a control application or the manufacturer's cloud. The following solutions were selected as representative of this category:
 - Gosund SP111 smart plug,
 - IG019 smart plug,
 - Neo SmartBulb Wi-Fi RGB.

- Infrastructure class of smart home systems, with the selected endpoints performing the role of broker in the control of IoT devices. The classic implementations of this approach are IoT solutions using ZigBee based on 802.15.4 standard [21], Z-Wave [22], or the Bluetooth [23] communication standard, where the relevant bridge transmits control signals to devices and in return receives data, measurements, and their status. This approach also implements more extended and advanced scenarios for smart rooms and even entire buildings. The following solutions were selected for the security testing of such systems:
 - Phillips Hue,
 - Aqara,
 - ConBee II,
 - Modemix,
 - Ikea Dirigera,
 - Sonoff.

This article is structured in six sections. Section 1 provides an introduction to the topic of IoT device security and the motivation behind the research. Section 2 presents the related work on IoT security assessments, including building laboratory environments for IoT security analysis and the widely used methodologies for this. Section 3 presents the state of the art in IoT system architectures, including an overview of the basic components of IoT devices and the current state of Smart Home IoT installations. Then, a reference threat model for IoT systems is provided. In Section 4, we describe the test bench used in this research, including the research methodology and laboratory environment. Section 5 focuses on vulnerability research, including techniques such as flash dumping, uploading malicious firmware, and communication channel sniffing. Wireless sniffing and accessing the root account are also discussed. This section is concluded with the results of the cybersecurity analysis conducted in this research. Section 6 provides conclusions on the key findings and recommendations for future work.

2. Related Work

As of today, several IoT security guidelines have been proposed. Most of them include a list of requirements, such as documents issued by government bodies like NIST (NIST IR 8259 Foundational Cybersecurity Activities for IoT Device Manufacturers [24] and 8259A IoT Device Cybersecurity Capability Core Baseline [25]) or ENISA Baseline Security Recommendations for IoT in Context of Critical Information Infrastructure [26] and Guidelines for Securing the Internet of Things [27]). Another well-recognized standard was proposed by the International Organization for Standardization (ISO). They issued the standard ISO/IEC 27400:2022 Cybersecurity-IoT security and privacy-Guidelines, which provides a comprehensive set of general requirements for IoT security [28]. A review of the existing guidelines is presented in [29]. The authors examined documents and propositions regarding IoT security requirements from 17 organizations. They also evaluated a few existing security mechanisms with the question of if they complied with the reviewed requirements.

There have been several industry propositions focusing on IoT security evaluation based on particular threats, such as OWASP Top 10 for IoT security [30]. The OWASP Top 10 is a list of the ten most common security threats in the context of a particular technology. The most well-known top 10 list is for web application security. The first Internet of Things top 10 threats list was proposed by OWASP in 2014 and updated in 2018. The list contains the following threats:

1. weak, guessable, or hard-coded passwords;
2. unsafe network services;
3. dangerous ecosystem interfaces;
4. no secure update mechanism;
5. using insecure or outdated components;
6. insufficient privacy protection;
7. unsecured data transfer and storage;

8. no device management;
9. unsafe defaults;
10. no physical reinforcement.

This list is intended to draw attention to the most common threats to IoT systems. This list allows developers, service providers, and manufacturers to focus on key risk areas and improve the security of their products and services. It is especially worth paying attention to less obvious points, such as the lack of a secure update mechanism, which may allow an attacker to modify a device's firmware or a lack of physical reinforcement that cannot be added with a software update.

Another proposition for the systematic security evaluation of IoT smart home devices was presented in [31], with the corresponding webpage [32]. It offers a methodology for evaluating the security of IoT appliances, taking into consideration several properties combined together into a scorecard. It is based on a generic model of IoT systems consisting of devices, mobile applications, cloud endpoints, and the communications between them. It is similar to the one shown in Figure 2. The model uses three main categories of evaluation:

- Attack vector—the main category of security aspects taken into consideration during security evaluation.
- Mitigations—this shows how security problems can be resolved.
- Stakeholders—this presents who is responsible for mitigating the encountered security problems.

Each of the categories has subcategories related to the class of object in the generic IoT system (device, mobile application, cloud endpoint, communications). The main security aspects represented by the Attack Vector within the scorecard are related to configuration, access control, data security (at rest and in transit), vulnerabilities, the openness of the system, and being susceptible to man-in-the-middle attacks (MitM). The corresponding webpage [32] complements the original paper with a guideline on the quantitative scoring each of the security aspects by assigning a specific score to a property per each category. The introduced approach could be widely applied as a comparable scoring system for IoT smart home devices. The authors reported the results of application of the methodology for 45 devices in [31]. In [32], they invited vendors and the security community to expand on the idea.

The practical security evaluation of IoT systems is subject to increasing attention within the cybersecurity community, as many challenges regarding them have been identified recently [33,34]. There are very new and important examples of research results for security aspects of IoT systems, especially smart home devices. The author in [35] presented a comprehensive guide for conducting security assessments of hardware-based systems such as IoT devices. An example work on IoT and embedded device security evaluation was presented by [36]. Other recent results were disclosed in the paper [37]. The authors applied a PETIoT kill chain to conduct a vulnerability assessment and penetration testing session on a smart bulb, the Tapo L530E by TP-Link. They found that four vulnerabilities affected the bulb, two of high severity and two of medium severity according to the CVSS v3.1 scoring system. The authentication method was not well accounted for and confidentiality was insufficiently implemented. In the results, they concluded that an attacker who is near the bulb can takeover all devices from the Tapo family. Moreover, an attacker could learn the victim's network Wi-Fi password, thereby escalating their malicious potential considerably. Further research considered buffer overflow vulnerabilities in Wemo smart plugs [38]. Wemo Mini Smart Plug V2 is a popular consumer device for remote-control electric devices. The device can be managed by a mobile application that allows the name of the device to be changed. The authors found that the name length should be limited to a maximum of 30 characters, but this is only enforced by the app itself. They performed a reverse engineering process to discover that they could circumvent the character limit, thereby resulting in a buffer overflow vulnerability. They demonstrated how to use the vulnerability for command injection attacks. However, Belkin (the device manufacturer) stated that the device is at its end-of-life, and thus it will not be patched. This vulnerability

was assigned an official ID of CVE-2023-27217 with a score of 9.8 (critical) [39]. A similar discovery was even escalated into the official advisory issued by the US Cybersecurity and Infrastructure Security Agency (CISA) [40]. It was found that KNX devices that use KNX Connection Authorization and support Option 1 are, depending on the implementation, vulnerable to being locked, and users are unable to reset them to gain access to the device. KNX is a popular protocol for smart home and building systems. As described by CISA: The BCU key feature on the devices can be used to create a password for the device, but this password can often not be reset without entering the current password. If the device is configured to interface with a network, an attacker with access to that network could interface with the KNX installation, purge all devices without additional security options enabled, and set a BCU key, locking the device. Even if a device is not connected to a network, an attacker with physical access to the device could also exploit this vulnerability in the same way. The official ID for this vulnerability is CVE-2023-4346 [39] and it assigned to this vulnerability a score of CVSS 3.1 of 7.5.

Based on the review of the referenced documents and materials, there is still space to propose comprehensive security analysis methodologies, with a complementary design of the lab environment and the practical procedures for such an evaluation. Evaluating the security of IoT devices, especially in smart home applications, seems to be a very important topic, which will only increase in importance in the coming years. Projects such as [32] introduced with the paper [31] is a good example of a milestone in the community effort to find a common language for sharing security concerns in the IoT domain; however, like every methodology or framework, it cannot deepen all parts. This paper addresses these challenges by proposing the unique advantage of combining together the evaluation methodology, the recommended tools, and presenting them in action through a set of experiments (Section 5). The summarized contributions of this approach are as follows:

- Emphasizing the low-level security aspects related to electronic parts of a device and its firmware;
- Deeply understanding the level of security of wired and wireless communication interfaces from different perspectives, not only considering man-in-the-middle attacks;
- Providing a practical insight into how to conduct security evaluation procedures and what tools should be used for that purpose.

A summary of the related work for this paper is present in Table 1. Each reference is shown by category, contribution, recognized limitations, and how this paper is positioned in comparison to them. Generally, we recognize that this paper complements and expands such methodologies as [31] by offering the mentioned contributions in the field of cybersecurity of IoT solutions in the smart home environment:

1. Design and implementation of a laboratory environment low-cost environments, which integrates research capabilities in the layers of firmware, wired, and wireless communication for research on the cybersecurity of IoT systems.
2. Development of a preliminary methodology for testing the firmware, wired, and wireless communication of selected representatives of IoT devices for testing.
3. Theoretical analyses and practical tests on the security of the indicated IoT devices were conducted to obtain quantitative results, with a security evaluation of the similarities with the broad scope of IoT devices.

For example, the approach presented in this paper can enable the selection of scores following the guidelines provided in [32] based on the actual level of security resulting from a practical and deep assessment.

Table 1. Summary of works related to this paper.

Reference	Year	Comment on Purpose and Limitations	Position of This Paper
[24,25]	2020	Provides security requirements for IoT devices only, without details of how to assess or ensure them.	This paper provides practical procedures to practically assess several aspects of IoT smart device security.
[26,27]	2017, 2020	Provides guidelines for securing IoT devices, but details of how to provide or ensure security is outside its scope.	This paper provides a methodology to assess several aspects of IoT smart device security, which could fulfill the provided guidelines.
[28]	2022	Official ISO standard with requirements for the security of IoT devices. Guideline to practically verify them are not included.	Using the approach from this paper such security requirements can be verified practically.
[29]	2021	Summarizes 17 security standards for IoT devices and their requirements.	This paper complements this with a practical security assessment methodology, and the proposition of tooling to conduct them.
[30]	2018	Security assessment methodology based on the most common IoT cyber threat scenarios provided by OWASP (Top 10). It does not provide details of how to test and verify them. It is also an older methodology.	The methodology from this paper provides the low-level procedures that are needed to conduct any of the scenarios.
[31,32]	2019	A good example of the community's effort to find a common language for sharing security concerns in the IoT domain, however, as with every methodology or framework, it cannot deepen all parts.	This paper addresses these challenges by proposing the unique benefit of combining together an evaluation methodology, recommended tools, and presenting them in action through a set of experiments.
[35]	2018	A comprehensive and complex guide on conducting security assessments of hardware-based systems with the proposition of tooling.	This paper concentrates on a practical and easy-to-follow methodology, with the proposition of supporting tooling. This should enable the community to easily start their own experiments based on this paper.
[36]	2023	Shows how to use firmware analysis for vulnerability research with the example of DJI Drone firmware.	This paper recognizes firmware analysis as one of the key steps of IoT security assessments and also goes beyond this.
[37]	2023	Shows the PETIoT kill chain approach for conducting vulnerability assessment and penetration testing of a smart bulb at networking level. The paper supports a similar case of recognizing how IoT devices can be dangerous in terms of being unsecured gateways to our networks.	Networking level analysis is one of the steps of the methodology proposed within this paper. Then, this is expanded, to offer a comprehensive approach for assessing IoT systems on different surfaces.
[38]	2023	The article discusses how to find important security vulnerabilities through IoT firmware analysis.	Firmware level analysis is one of the steps of the methodology proposed within this paper. Then, it is expanded to offer a comprehensive approach for assessing IoT systems on different surfaces.

3. A Security Model for IoT Systems in Smart Home Applications

This section summarizes architectures for application of Internet of Things systems in smart home environments. Then, a generic threat model for such systems is described, considering cyber attacks across all layers. We also refer to the general supply chain security context, which is of increasing concern in the cybersecurity community.

3.1. Introduction to Smart Home IoT Systems

Smart homes are designed to help humans with daily tasks by replacing manual actions with automated ones [41,42]. Rapid technological progress has increased the number of smart devices in buildings, especially the smart home environments considered by this paper [43]. To visualize an example of smart home topology, the diagram in Figure 1 has been presented. In this topology, on every floor, there are bridges connected with a wire. They aggregate traffic from end devices, which are connected to bridges. The system can be administrated locally inside the building or remotely via the cloud.

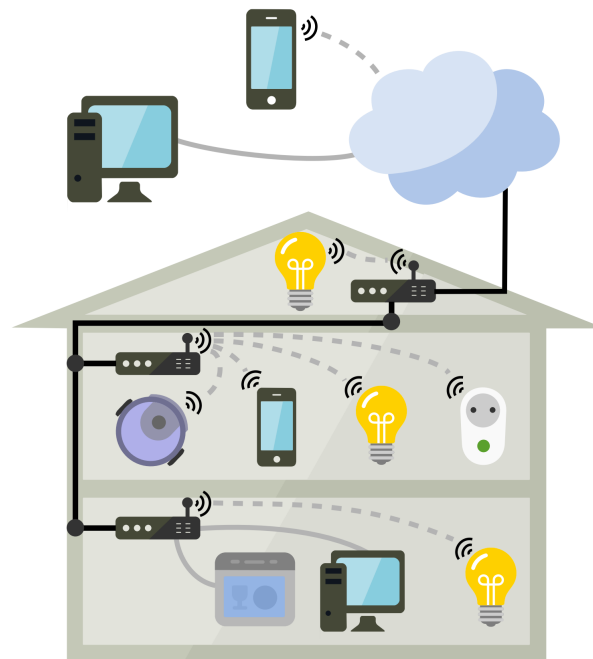


Figure 1. Diagram of a smart home IoT installation, featuring various wired and wireless devices.

During a review of known communication standards for home IoT devices [44], the use of the CoAP (Constrained Application Protocol [45]), MQTT (Message Queuing Telemetry Transport [46]), and HTTP (Hypertext Transfer Protocol [47]) protocols was identified, with MQTT being the most frequently used protocol. Its low complexity, the use of TCP (Transmission Control Protocol [48]) in the transport layer (reliability) and support for message encryption using TLS (Transport Layer Security [49]) make this protocol a common choice for simple IoT devices. The disadvantages of this protocol include a lack of standardization of message formats among various manufacturers and the lack of built-in mechanisms for controlling access to selected resources. The MQTT protocol is based on the pattern of publishing and subscribing using a message distribution node (broker) between end devices. This allows many-to-many communication between simple devices with low bandwidth (the distribution node must have more resources). The security of this protocol is ensured by encapsulation in the TLS protocol. When using the TLS protocol, the chain of trust must be properly implemented. End devices connecting to a distribution node (broker) need a trusted certificate from this node to confirm the security of communication. Compromising the private key within this certificate means that certificates must be exchanged on all end devices, so a very limited number of people in the organization should have access to it.

One of the most important aspects of network communication security in the IoT is securing wireless interfaces. Frequently used wireless protocols include ZigBee [21], Bluetooth [23], and Z-Wave [22]. Vendors prefer the ZigBee standard due to a better energy-efficiency than the other standards, so the device's battery will last much longer. Moreover, ZigBee networks are easy to install and self healing. The topology may be dynamically

changed in the case of the malfunction of one of the devices. Due to the aforementioned benefits, ZigBee is commonly used, and more wireless communication analysis has focused on ZigBee. ZigBee supports two security models [50]: centralized and distributed. The first model can contain three types of devices: router, coordinator, and end device. The router can forward ZigBee packets between end devices within a given network or between networks when it is connected to successive routers. The coordinator only works within one network, stores security keys, and enables authentication of end devices when connecting them to the network. End devices can be added to the ZigBee network, but they cannot create networks and forward packets by themselves. The distributed model does not include a coordinator, and encryption keys are passed through interconnected routers. This solution is less secure, because the encryption keys are not stored by a single device.

3.2. Threat Model for IoT Systems and Supply Chain Context

IoT systems are meant to help us but can also lead to security threats if the implemented devices are not tested properly. Device vulnerability can facilitate smart home attacks [51]. Based on the introduced IoT system topology, a complementary cyber threat model can be established. The complexity of IoT systems means that cyber threat modeling must be performed for every layer [52]. A simple but generalized model of the IoT system is shown in Figure 2.

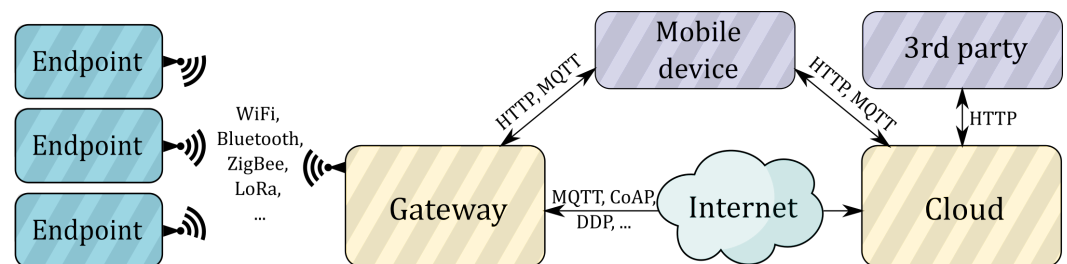


Figure 2. Generalized model of an IoT system.

Threats that may occur in such a system include

- security errors in protocol standards—the protocols used may have security gaps;
- security bypass—the attacker may be able to bypass the implemented security mechanisms;
- denial of service—IoT devices are characterized by low computing power and bandwidth, so it is easy to use these limited resources to saturate such a device and thus make it unavailable;
- weak cryptography—energy limitations may require the use of simpler, but, at the same time, weaker and less resistant to cryptographic attack, data protection algorithms;
- incorrect implementation of the system and protocols—at the stage of creating software for an IoT device, common implementation errors may appear, which may constitute security gaps;
- lack of access control—allowing access to unauthorized resources for users and third parties;
- incorrect configuration—this may allow unauthorized access to various parts of the system, uses contrary to its intended purpose, or open access to other systems;
- vulnerabilities in software and hardware—these may allow the execution of unauthorized actions and code within the various parts of the system, as well as uses contrary to its intended purpose or open access to other systems.

Any element of a particular IoT device can be used as a gate for an attacker [53]. The quantitative demand for IoT solutions in conjunction with the expectation of low device prices results in the unification of IoT hardware platforms, with production concentrated among a few major suppliers [7,8].

End platforms differ from each other only by the end-user software. One of the most popular and cheapest platforms used in Smart Home devices is the ESP8266EX

microcontroller. This is a simple system-on-a-chip (SoC) device equipped with an integrated Wi-Fi module [54]. Its successor is the ESP32 platform, which additionally offers a Bluetooth module. However, it is more expensive, which determines a smaller market share as a component in cheap IoT devices. One of the basic features of many of the available IoT devices is the ability for users to modify them. This aspect can be treated both as a great advantage and a potential source of problems. It is expected that the user will receive a certain degree of freedom in configuring or even modifying IoT devices to their requirements. However, with a low awareness of cybersecurity risks, this opens up a space that cyber attackers can use to implement various attack scenarios. The manufacturer of the described microcontroller, Espressif, has enabled basic programming using an open UART interface. This approach allows for a trivial extraction of the device's source code, and the only security measures exist at the level of the embedded software, i.e., firmware. This approach does not guarantee security in terms of the integrity of the physical and logical interface of the device. One of the possible threat scenarios could include a straightforward placement of malicious executable code. As a result, an attacker could gain access to the Wi-Fi password of a router. The ability to extract embedded software is not limited to devices produced by Espressif. Many other IoT device manufacturers also use open interfaces such as UART, JTAG, and SPI, making firmware extraction relatively easy. Additionally, techniques such as bus sniffing, firmware dumping, and reverse engineering can also be utilized to extract embedded software. However, it is important to note that these techniques may require specialized equipment and knowledge.

4. Design of Lab and Methodology for IoT Smart Device Security Research

In this section a universal lab environment for testing smart home devices is described. More detailed models are presented in the three research methods. Each method focuses on a separate test scenario.

4.1. Lab Environment Design

The main focus of this research was the development of a test bench, as depicted in Figure 3. The test bench was designed to simulate various cyber-attack scenarios, to evaluate the effectiveness of different security mechanisms.

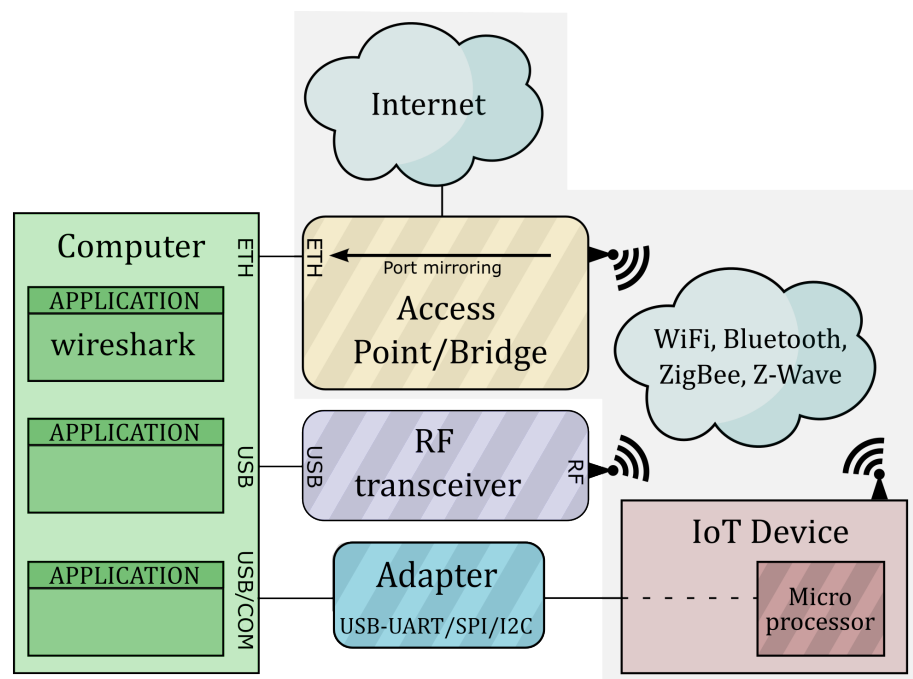


Figure 3. Universal test bench for IoT devices (basic setup marked with gray background).

The proposed setup is a universal and versatile solution, aimed at identifying as many security weaknesses as possible in the targeted device under test (DUT). The test bench consists of two main sections: the nominal system part (marked with a gray background in Figure 3) and the malicious equipment part. The nominal system simulation includes the DUT and an access point or bridge, connecting the device to the packet network, and ultimately to the internet. The DUT (IoT device) connects wirelessly to the broker (access point/bridge) using the Zigbee, Z-Wave, Bluetooth, or Wi-Fi protocols. The research equipment part, on the other hand, includes a computer with the necessary software installed, a radio frequency transceiver, and adapters for wired protocols, such as UART/SPI/I2C/JTAG. The test bench is modular, allowing for certain elements to be omitted during specific tests. This ensures that the test bench can be adapted to the specific requirements of the DUT and the research objective. Furthermore, all elements of the model are, in practice, specific devices that will be detailed later. The first part of the test bench simulates the nominal system in which the DUT operates. This includes the DUT and the access point or bridge, which connects the device to the packet network and ultimately to the internet. The DUT (IoT device) connects wirelessly to the broker (access point/bridge) using Zigbee, Z-Wave, Bluetooth, or Wi-Fi protocols. This part of the test bench serves as a control group, allowing researchers to compare the results of their vulnerability research with the expected behavior of the DUT in a regular operating environment. The second part of the test bench is the research equipment, which is needed to carry out vulnerability research. This includes a computer with the necessary software installed, RF transceiver, and adapters for wired protocols, such as UART, SPI, I2C, or JTAG. The computer is used to control the transceiver and adapters, as well as to analyze the data collected during the research. The mentioned components enable researchers to intercept and manipulate the communication between the DUT and the access point/bridge. This allows for the identification of weaknesses in the communication protocols used by the DUT.

In further studies, the presented components can be replaced by specific tools according to the testing scenario. As an adapter, the research team utilized and recommends the following tools:

- PL2303,
- BusPirate,
- Flipper Zero.

In the course of the experiments, the following RF transceiver tools were utilized, depending on the specific requirements and objectives of the study:

- HackRF One,
- Flipper Zero.

The access point used depended on the system under test and was often uniquely associated with it. The research team recommends the following devices:

- Mikrotik hAP AC3 LTE6,
- ConBee II,
- Philips Hue Bridge 2.0,
- Aqara M2,
- Modemix W-WG001.

4.2. Proposed Security Assessment Methodology

Several research methods are needed to verify the cybersecurity of smart home devices. According to the device protocols, communication method, and hardware elements, individual approaches are required. For the purposes of this article, we proposed a security analysis methodology with utilization of the test bench described in Section 4.1. The three main areas of activities during such security assessment include

- device physical analysis and firmware dumping,
- firmware analysis,

- network communications analysis including different contexts, such as packet-based communication patterns, wired and wireless interfaces, and local and external connections.

The relations between these three areas of assessment are presented in Figure 4.

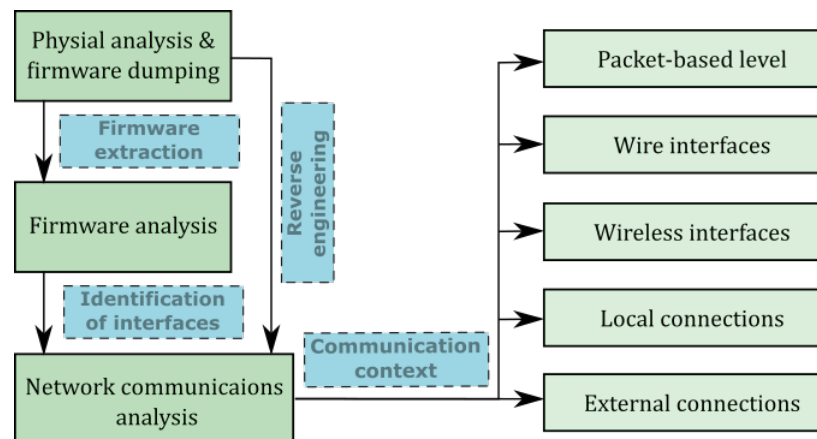


Figure 4. Relations between three main areas of activity within proposed methodology for IoT security analysis.

4.2.1. Device Physical Analysis and Firmware Dumping

The proposed test bench represents a crucial component in conducting a credible process of security analysis of IoT devices. By examining the device's vulnerability to embedded firmware dump and flash through the use of UART, SPI, or I2C interfaces, the test bench can provide valuable insights into the security measures in place in the device's firmware. In some cases, the firmware of the device can be controlled through the aforementioned interfaces, allowing the functionalities of the device to be manipulated, and hence malicious actions can be taken. For this methodology, the setup in Figure 5a is suggested. These wired interfaces are typically used by manufacturers during the production stage to upload embedded firmware, as well as for debugging and repairing purposes during the return process. The test scenario can be carried out with the device operating standalone, without the need for a network connection. Access to the interfaces, however, typically requires taking the device apart. The implementation of this test bench offers a comprehensive analysis of the device's firmware security measures and helps to uncover any weaknesses that may exist. The general process of exchanging and modifying the embedded software can be divided in a few steps:

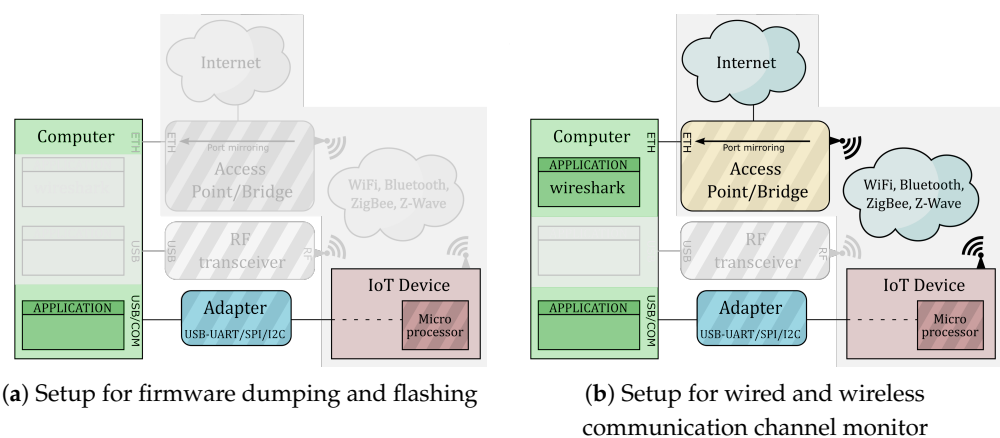


Figure 5. Test bench setup examples.

1. connecting the device being tested via an adapter to the computer;
2. preparing a script or software on a computer, optionally your own malicious firmware;

3. executing proper script or software on the computer;
4. dumping, flashing, or controlling the firmware;
5. impact analysis.

4.2.2. Network Communications Analysis

Most consumer smart home devices rely on wireless communication; however, as previously mentioned, they may also have hidden wired interfaces. In this testing methodology, a testing setup was proposed, as depicted in Figure 5b, which expands upon the setup in Figure 5a to include a wireless communication path. The utilization of an access point or bridge with port mirroring functionality enables the copying of the communication messages between the tested device and the manufacturer's cloud. The recommended Wireshark software [55] additionally allows the analysis of the transmitted packets. It is also worth mentioning that service messages may be transmitted via the wired path, which can be monitored via an appropriate adapter. In this case, the general process can be divided into several steps:

1. connecting the IoT device to the access point or bridge;
2. connecting the device via an adapter to the computer;
3. connecting the computer with the access point;
4. setting up port mirroring on the access point to copy any traffic into the computer interface;
5. powering up the device under test and executing common actions;
6. monitoring and saving both wired and wireless channels;
7. analysis of saved traffic for cyber security purposes.

It is possible to go one step further in this methodology by capturing and decoding radio communications. This can be accomplished through the use of specialized hardware, such as a software-defined radio, which allows for the detection and decoding of wireless signals. For this purpose, the HackRF One is recommended. The test bench setup is shown in Figure 6. Commonly used communication protocols (ZigBee, Z-Wave, Bluetooth, and Wi-Fi) have their own security standards, which ensures the confidentiality of the transmitted data. Hence, each of these protocols needs a separate signal processing procedure. Despite this, the general process of radio sniffing and decoding can be divided into a few steps:

1. executing the program for radio sniffing;
2. setting transceiver parameters to adjust to specific frequency ranges, modulation, and communication protocol;
3. sniffing communication between devices;
4. signal processing to form protocol messages;
5. decoding the data section of the formed messages.

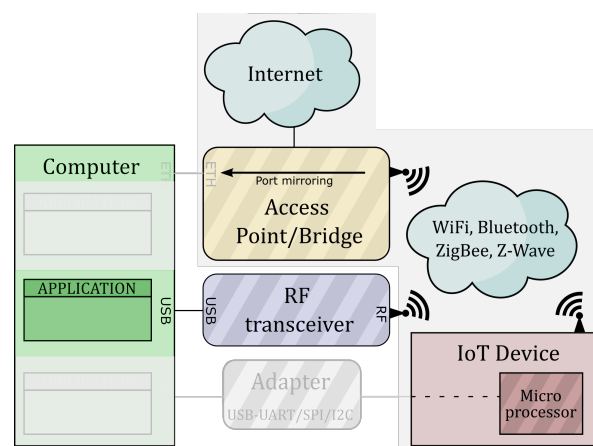


Figure 6. Test bench setup for RF sniffing.

4.2.3. Summary of Security Analysis Methodology Procedures

Table 2 summarizes the analysis methodology steps presented in Sections 4.2.1 and 4.2.2. It also includes example tools to successfully execute each analysis step. The use of the recommended tools is presented throughout the testing scenarios in Section 5.

Table 2. Summary of proposed security analysis methodology and recommended tools.

Type of Analysis	Procedure/Step	Tools
Physical analysis and firmware dumping	Opening the device's case	Screwdrivers, plastic prying and opening tools
	Identification of I/O interfaces	Eye analysis
	Identification of components	Microscope
	Identification of debug and service interfaces	Oscilloscope, multimeter
	Connecting to debug interface	Soldering iron, cables
	Firmware extraction	Soldering iron, cables, manufacturer's software tools
Firmware analysis	Identification of system's version	Terminal—uname, cat
	Identification of running services	Terminal—ps
	Identification of installed packages	Terminal—ls, package manager (e.g., pacman, opkg, apt)
	Reverse engineering of applications	Terminal—grep, find, vim; applications—Ghidra, IDA Pro
	Finding the well-known vulnerabilities	Metasploit, searchsploit, exploitdb
Communication analysis	Port scanning	nmap
	Identification of assets based on services	e.g., web browser or Terminal with wget and curl for HTTP communications
	Identification of services' versions	nmap, metasploit
	Identification of network communication patterns e.g., with manufacturer's cloud	Wireshark
	Finding the well-known vulnerabilities	Metasploit, searchsploit, exploitdb
	Radio sniffing and decoding	HackRF, GNU Radio, Wireshark

5. Experiments and Security Analysis of IoT Smart Home Scenarios

In Section 4, the general idea of the IoT security assessment lab and the following methodology was proposed. As was stated, different methods are needed to verify the cybersecurity of IoT Smart Home devices. For the purposes of this article, a security analysis methodology was proposed to cover three main areas:

- device physical analysis and firmware dumping;
- firmware analysis;
- network communications analysis including different contexts, such as packet-based communication patterns, wire and wireless interfaces, and local and external connections.

Then, the generic realization of these methods was described in the subsequent Sections: (device analysis, firmware dumping, and firmware analysis) and (network communication analysis). Table 2 summarized the security analysis methodology steps. This Section presents the practical application of the proposed methodology in a real lab environment for IoT security analysis. As a reference IoT smart home device to present the practical experimentation, the Gosund SP111 smart plug was selected.

In the following subsections, the proposed methodology is practically applied step by step. In Section 5.1, the physical analysis of the device and the execution of a firmware dumping procedure are presented. Section 5.2 provides techniques for analysis of the wired and wireless interfaces that can be found on IoT devices, especially ZigBee. The proposed methodology and the provided practical procedures of IoT security assessment were then applied to the selected set of devices:

- IoT end devices: Neo SmartBulb Wi-Fi RGB,
- IoT bridges: Phillips Hue, Aqara M2, Modemix W-WG001, Ikea Dirigera and Sonoff ZigBee.

The summarized results are presented in Section 5.3.

5.1. Security Analysis from View of IoT Smart Home Device

In the following Sections 5.1.1 and 5.1.2, the two steps of the proposed methodology are presented. First, it is shown how to conduct the physical analysis of the device, and how to execute the dumping procedure of the firmware from the IoT device is presented. Then, the experiment of flushing the modified firmware with the injected backdoor is described.

5.1.1. Dumping Firmware from Gosund SP111 Smart Plug

The subject of the tests was a Gosund SP111 smart socket. A test environment was built based on the test bench (Figure 5a), as illustrated in Figure 7 in this particular case. Figure 8a,b present the realization of the setup from Figure 7. The lab designed and shown in Section 4.1 can be flexibly reconfigured for the particular test scenarios.

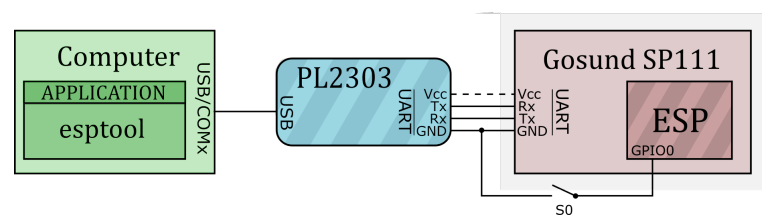
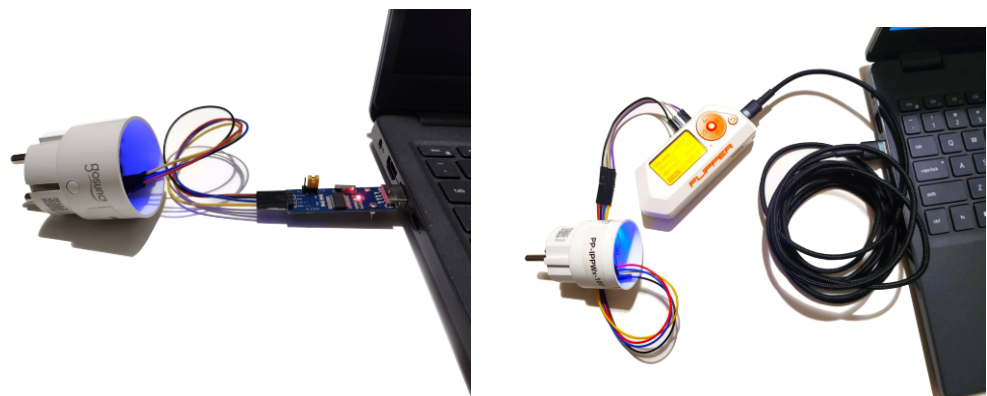


Figure 7. Test bench setup for dumping and flashing malicious firmware on Gosund SP111 smart plug.



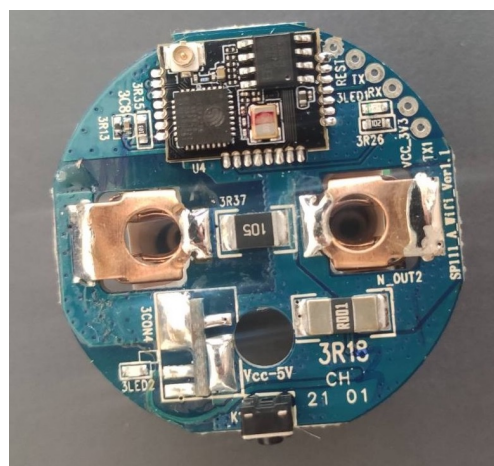
(a) Gosund SP111 connected via USB-UART adapter to the computer with esptool. **(b)** Gosund SP111 connected via Flipper Zero to the computer with esptool.

Figure 8. Setting up the dumping tools for Gosund SP111 analysis.

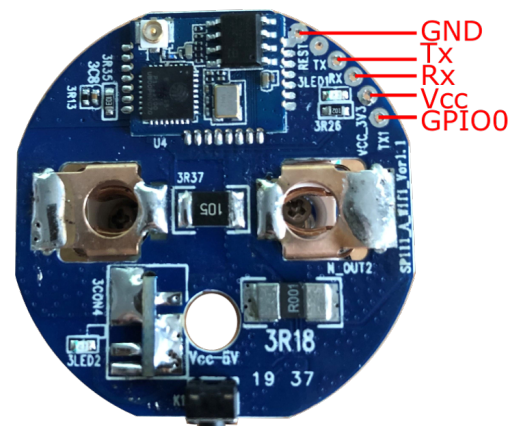
Upon taking the socket apart (Figure 9a), it was found to have well-labeled UART interface pins, but they were not exposed (Figure 9b). Thus, to connect the device to the computer, a PL2033 USB-UART adapter was used. This adapter also allowed powering the device with 3.3V or 5V depending on the physical jumper's configuration. The socket's computing unit is an ESP8266EX microcontroller [56]. Such a socket can be integrated into a smart home network via Wi-Fi. Initialization (pairing) is possible using the Tuya

Smart mobile application. In this experiment, the esptool application was installed on the computer as an open-source software dedicated to communicating with ESP microcontrollers. It allows, among other things, importing and exporting firmware and many other functions [57]. The esptool software makes research work easier, for example, by automatically detecting connected devices. If this step is carried out correctly, firmware extraction is performed with a simple command. In the event of problems, manual configuration is possible, to detect the device for research. This is necessary when connecting multiple devices simultaneously for research when using different computer ports. The firmware was extracted to a binary file. To do this, the addresses of the cells in the register where the code is located had to be provided. The starting address is constant and is 0x000000, but the end address depends on the firmware size. Esptool allowed determination of this size for the tested device of 1 MB, which means that the address of the last memory location was 0x100000. After the extraction, a compiled version of the firmware binary file was obtained. It was noted that ASCII character strings were not converted in any way, making them easily reviewable. It was tested whether the content of the binary file was as expected by

- extracting the first version of the firmware in an offline setup, before connecting the plug to the Internet;
- extracting the second version firmware in an online setup, after connecting the plug to the Internet and performing the update of the firmware.



(a) Gosund SP111 PCB.



(b) Recognition of physical interfaces in Gosund SP111 PCB.

Figure 9. Physical analysis of Gosund SP111.

The result of this test is presented in Figure 10. It confirmed the correctness of the obtained binary file.

Then, the subsequent analysis was started. The distinctive memory area was found at addresses from 000fb000 to 000fb17f, which is shown in Figure 11. The found memory area contained structures in JSON format. The first such structure contains information about the last 12 bits of the device's MAC address (3 characters in hexadecimal notation), the object identifier along with the authorization key and a bool variable named prod_test. The second structure contains the default SSID and password for an unknown Wi-Fi access point. At this stage, the initial conclusion was that the discovered structures may serve as backup access to the device. The correctness of the MAC address was also confirmed by messages from the UART interface.

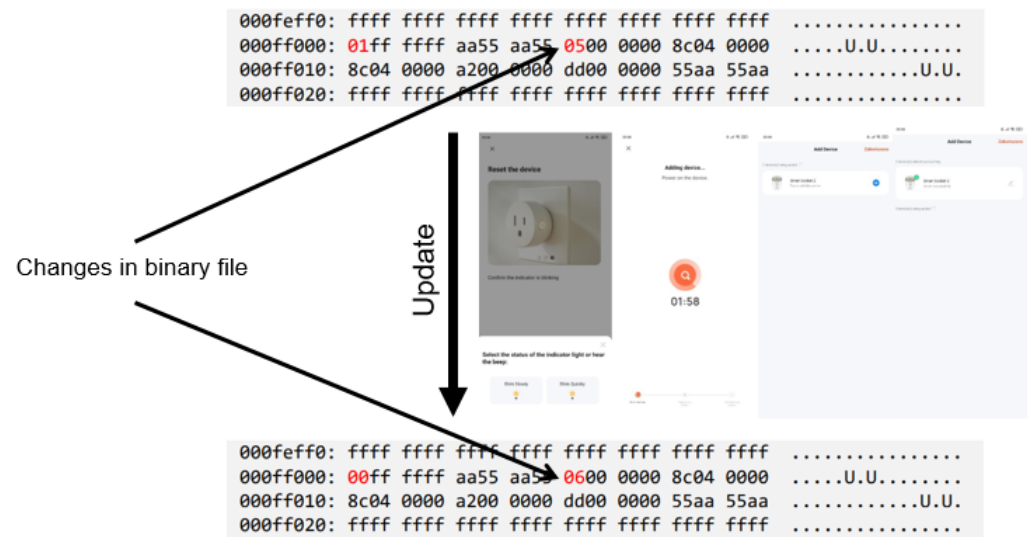


Figure 10. Observation of firmware binary file before and after update.



Figure 11. Analysis of memory area from 000fb000 to 000fb17f in Gosund SP111 firmware.

5.1.2. Flashing Malicious Firmware to Gosund SP111 Smart Plug

The experiment was a continuation of previous research and involved testing the capability of modifying the firmware in a manner that would allow for the transmission of the wireless network's identifier and password to the attacker's Command & Control (C2) server located outside the local network and over the Internet, followed by re-uploading it to the tested IoT device. The attack scenario was carried out without the victim's knowledge and involved configuring the modified IoT device according to instructions, while the victim was performing actions. At the research stage, the procedure assumes the absence of devices blocking or otherwise hindering communication between the local network devices and the rest of the Internet, such as firewalls or intrusion detection/intrusion prevention systems (IDS/IPS). With this open approach, it was also possible to formulate proper conclusions and recommendations for building a secure IoT network architecture. The experiment utilized the setup depicted in Figure 7.

The modification of the compiled firmware was problematic, so the basis of the developed modified version of the firmware was the use of the open-source software Tasmota [58]. This is a universal software platform for ESP8266 based on the popular Arduino framework that allows the management of multiple IoT devices. Its popularity is

due to its low entry threshold for application, as it uses a declarative template mechanism to describe the functions that a given interface performs in the specified device. For the purpose of the experiment, the firmware was modified so that it would send the network identifier and password to the attacking server when the internal HTTP server started. The choice of these execution conditions was not arbitrary. It resulted from the preliminary analysis of the device, as the experimental verification confirmed that at this moment the device already has access to the Internet, which was necessary to establish the simulated C2 communication. The code of the modification is shown in Figure 12. Then, the proper data transmission on the C2 was confirmed by monitoring the logs.

```
/* tasmota/xdrv_01_webserver.ino */
void StartWebserver(int type, IPAddress ipweb) {
  ...
  Wi-FiClient client;
  if(client.connect("<IP OR HOSTNAME>", PORT))
  {
    client.print(SettingsText(SET_STASSID1 + Settings->sta_active));
    client.print(":");
    client.println(SettingsText(SET_STAPWD1 + Settings->sta_active));
    client.flush();
    client.stop();
    Serial.println("[C2] Success");
  }
  else {
    Serial.println("[C2] Error!");
  }
  ...
}
```

Figure 12. Source code modification of Tasmota firmware, server side.

5.2. Security Analysis on Communication Interfaces of IoT Smart Home Devices

In the following Sections 5.2.1 and 5.2.2, the practical analysis of wired and wireless interfaces recognized in the device is described. Section 5.2.2 specifically focuses on analyzing Zigbee communications.

5.2.1. Analysis of Wired and Wireless Communication of Gosund SP111

The selected IoT socket for research cooperates with the Tuya cloud, and pairing relies on inputting the SSID and password into the dedicated application and sending them to the IoT device. The device then authenticates using the obtained data. Given that this communication takes place wirelessly using Wi-Fi, the test bench shown in Figure 13 was used for further research. Communication is monitored with a Mikrotik wireless access point with traffic monitoring on the interfaces, copying the network traffic for real-time analysis. Serial interface communication was observed using a computer station with a USB-UART adapter. The communication log is shown in Figure 14.

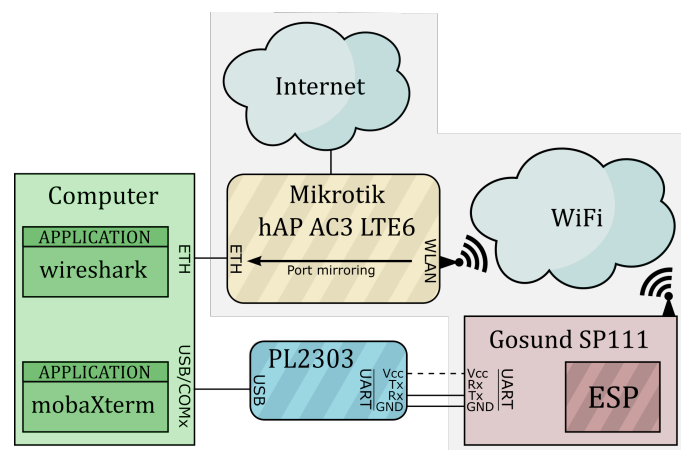


Figure 13. Test bench setup for monitoring wired and wireless communication channels on Gosund SP111 smart plug.

```

[notice]mqtt_client.c:590 gw wifi stat is:1
...
1 → [notice]gw_intf.c:420 Authorization success
mode : sta(70:03:9f:5d:07:e0)
add if0
[notice]device.c:729 ID:(null)
[err]app_dltj.c:193 app_dltj_init
scandone
state: 0 -> 2 (b0)
state: 2 -> 3 (0)
[notice]mqtt_client.c:590 gw wifi stat is:5
state: 3 -> 5 (10)
add 0
aid 2
pm open phy_2,type:2 0 0
cnt
2 → connected with LaVA_Monitored_2.4, channel 1
dhcp client start...
ip:192.168.88.248,mask:255.255.255.0,gw:192.168.88.1
3 → [notice]mqtt_client.c:590 gw wifi stat is:5
[notice]mqtt_client.c:590 gw wifi stat is:6
[notice]mqtt_client.c:633 DNS START 03-09 17:55:54
[notice]mqtt_client.c:660 DNS END 03-09 17:55:54
[notice]mqtt_client.c:666 MQTT CONN START 03-09 17:55:54
[notice]mqtt_client.c:689 MQTT CONN END 03-09 17:55:55
[notice]mqtt_client.c:788 mqtt connect success
[err]smart_wf_frame.c:2243 unsupport frame type:64
4 → [notice]smart_wf_frame.c:3823 firmware self detect upgrade start...
5 → [notice]smart_wf_frame.c:3833 get fw ug info op_ret:2

```

Figure 14. Service messages of Gosund SP111 smart plug monitored via UART interface.

The firmware sends all service information to the UART port according to a specified catalog of labels. These messages provide insights into the code structure and operations performed by the device, including visible file names and locations where actions are taken. Line (1) of Figure 14 shows the successful authorization of the mobile device in the socket network and line (2) shows the successful authorization of the socket to the Wi-Fi access point. Further, the observed actions include connecting to the cloud through the MQTT protocol, checking for firmware updates and starting the update. Another observation is that the interface transmits error information during user interaction with the device. The abovementioned operations could be confirmed in a parallel analysis of the Wi-Fi 802.11 network traffic communication. This was conducted using the Wireshark program, as shown in Figure 15.

No.	Source	Destination	Protocol	Length	Info
10036	192.168.88.248	35.156.42.116	TCP	105	14381 → 8886 [SYN] Seq=0 Win=4380 Len=0 MSS=1460
10051	35.156.42.116	192.168.88.248	TCP	105	8886 → 14381 [SYN, ACK] Seq=0 Ack=1 Win=2048 Len=0 MSS=1460
10052	192.168.88.248	35.156.42.116	TCP	101	14381 → 8886 [ACK] Seq=1 Ack=1 Win=4380 Len=0
10057	192.168.88.248	35.156.42.116	TLSv1.2	168	Client Hello
10062	35.156.42.116	192.168.88.248	TCP	101	8886 → 14381 [ACK] Seq=1 Ack=68 Win=2048 Len=0
10063	35.156.42.116	192.168.88.248	TLSv1.2	249	Server Hello, Server Key Exchange, Server Hello Done
10071	192.168.88.248	35.156.42.116	TLSv1.2	162	Client Key Exchange
10092	35.156.42.116	192.168.88.248	TCP	101	8886 → 14381 [ACK] Seq=149 Ack=129 Win=2048 Len=0
10095	192.168.88.248	35.156.42.116	TLSv1.2	107	Change Cipher Spec
10102	35.156.42.116	192.168.88.248	TCP	101	8886 → 14381 [ACK] Seq=149 Ack=135 Win=2048 Len=0
10103	192.168.88.248	35.156.42.116	TLSv1.2	186	Encrypted Handshake Message
10112	35.156.42.116	192.168.88.248	TCP	101	8886 → 14381 [ACK] Seq=149 Ack=220 Win=2048 Len=0
10113	35.156.42.116	192.168.88.248	TLSv1.2	192	Change Cipher Spec, Encrypted Handshake Message
10121	192.168.88.248	35.156.42.116	TLSv1.2	234	Application Data
10131	35.156.42.116	192.168.88.248	TLSv1.2	170	Application Data
10134	192.168.88.248	35.156.42.116	TCP	101	14381 → 8886 [ACK] Seq=353 Ack=309 Win=4072 Len=0
10143	192.168.88.248	35.156.42.116	TLSv1.2	202	Application Data
10151	35.156.42.116	192.168.88.248	TLSv1.2	170	Application Data
10260	192.168.88.248	35.156.42.116	TCP	101	14381 → 8886 [ACK] Seq=454 Ack=378 Win=4003 Len=0

Figure 15. Packet traffic of Gosund SP111 smart plug monitored via Wireshark tool.

Based on this analysis, it could be confirmed that communication with the manufacturer's cloud was secured via the TLS (transport layer security) session protocol. However, an outdated protocol version 1.2 [59] was used, for which vulnerabilities, such as Poodle [60] and Raccon [61], are well-known.

5.2.2. Analysis of Zigbee Communications of ConBee II Gosund IG019 Smart Plug

ZigBee security is based on two pairs of encrypted 128-bit keys: network and link keys [62]. The network key is the same for the entire private network and is used for broadcast communication, while the link key is assigned to a single connection between two devices. Within a configured ZigBee network, keys are encrypted prior to communication between devices. When a new, previously unconfigured device tries to connect to the ZigBee network, it needs to obtain a network key. This key can be transferred to a new device in several ways:

- by manual entry of the key by the administrator;
- assigned to the device at the production stage;
- provided by the network coordinator using a transport key;
- otherwise specified by the network administrator.

For many end devices, it is impossible to manually enter the network key, owing to the lack of a user interface. The second method can be implemented for systems using only one vendor. The third method is commonly used, and some vendors set the default ZigBee transport key ZigBeeAlliance09. In this instance, the network key can be easily obtained. This vulnerability was checked in the following experiment. A schema for wireless communication sniffing has been proposed (Figure 16). As an example, the schema was used to check the security of the ZigBee bridge ConBee II. The lab realization of the test bench is presented in Figure 17.

The initialization phase is critical for ZigBee security, owing to the network key transport. There is only one network key in the ZigBee network and it is used to encrypt all data transmitted through the network. The initialization of the Gosund IG019 smart plug to Conbee II was sniffed with HackRF One. Full signal processing was performed by the prepared program in GnuRadio. The setup of the program ZigBeePacketsReceiver.grc is shown in Figure 18.

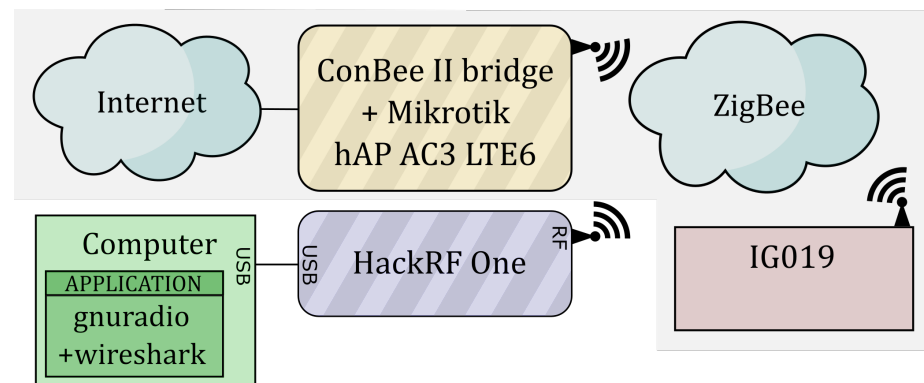


Figure 16. Test bench setup for radio communication sniffing of IG019 smart plug.

ZigBeePacketsReceiver.grc program (Figure 18) uses gr-osmosdr, gr-ieee802-15-4 and gr-foo packages [63]. The Gr-osmosdr package was used to capture the radio signal from HackRF One. It enabled the examined frequencies to be adjusted from one of sixteen ZigBee channels. Afterwards, the signal was passed to gr-ieee802-15-4 to form ZigBee packets based on OQPSK demodulation. Finally, the sniffed packets were forwarded to the Wireshark application with the gr-foo module. The transmission could be saved to a pcap file after the experiment. An example output is shown in Figure 19. The association response message is highlighted in blue. This informs about a successful initialization of the Gosund smart plug to Conbee II, due to the sent PAN (Private Area Network) ID. The subsequent message after the accepted association contains the network key. The data are encrypted with a transport key and hash function. If a gateway uses the default transport key ZigBeeAlliance09, it is trivial to decode the Key-Transport Key message. Such a vulnerability leads to network key exposure. With a known network key, every

message for the ZigBee network can be decoded. A leak of the network key can lead to more complex attacks, such as replay attacks or denial of service attacks. As a result of these attacks, interception of the ZigBee network can be carried out.



Figure 17. Lab realization of the setup from Figure 16.

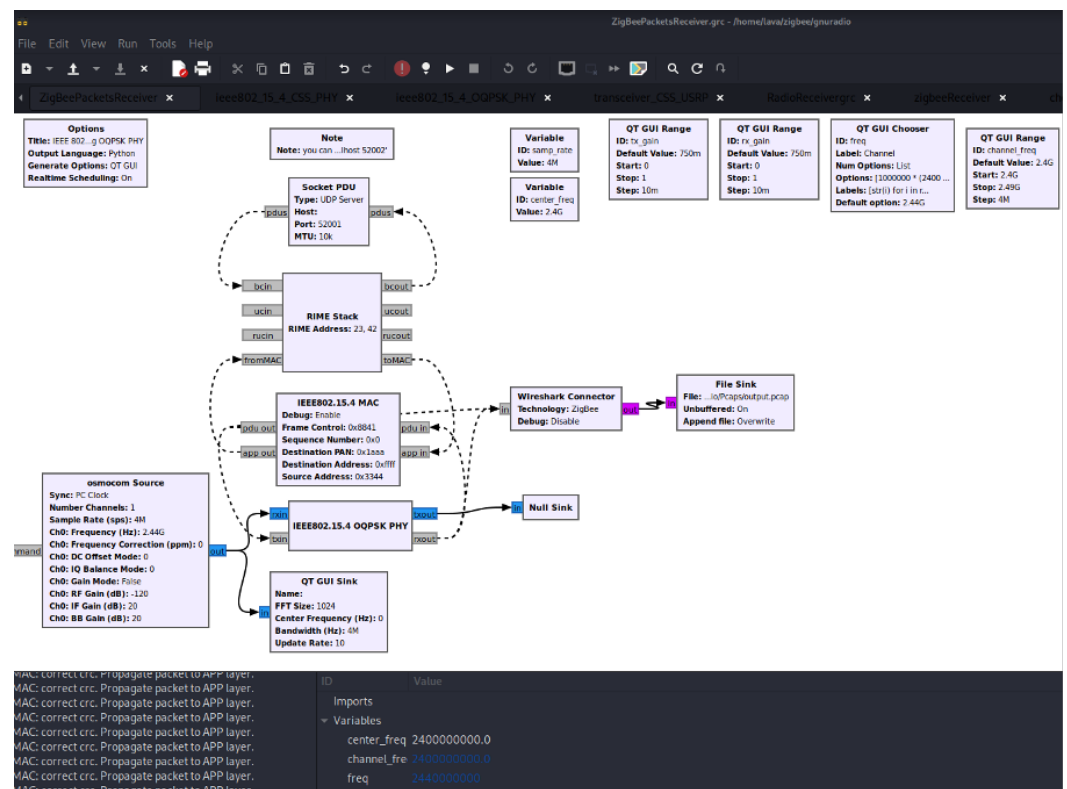


Figure 18. The setup of the GNU Radio program `ZigBeePacketsReceiver.grc` for radio communication sniffing of the IG019 smart plug.

No.	Time	Source	Destination	Key Id	Info
1	0.000000	0x0000	Broadcast	Network Key	Command, Dst: Broadcast, Src: 0x0000
2	2.520457	0x0000	Broadcast	Network Key	Data, Dst: Broadcast, Src: 0x0000
3	2.521594	0x0000	Broadcast	Network Key	Data, Dst: Broadcast, Src: 0x0000
4	3.143810	0x0000	Broadcast	Network Key	Data, Dst: Broadcast, Src: 0x0000
5	3.144360	0x0000	Broadcast	Network Key	Data, Dst: Broadcast, Src: 0x0000
6	13.401898	0x0000	Broadcast	Network Key	Beacon, Src: 0x0000, EPID: dresden_ff:ff:08:08:3b
7	15.106098	0x0000	Broadcast	Network Key	Command, Dst: Broadcast, Src: 0x0000
8	15.213423				Ack
9	15.412276				Ack
10	15.433037	00:21:2e:ff:ff:08:08:3b	a4:c1:38:00:18:27:06:8f		Association Response, PAN: 0x7113 Addr: 0x953f
11	15.438242	0x0000	0x953f	Key-Transport Key	APS: Command
12	15.476113	0x0000	0x953f	Network Key	Data, Dst: 0x953f, Src: 0x0000
13	15.480103				Ack
14	15.498032	0x0000	0x953f	Network Key	Data, Dst: 0x953f, Src: 0x0000
15	15.501443				Ack
16	15.503957	0x0000	0x953f	Network Key	Data, Dst: 0x953f, Src: 0x0000
17	15.509823	0x0000	0x953f	Network Key	Data, Dst: 0x953f, Src: 0x0000
18	15.536251	0x953f	Broadcast	Network Key	Data, Dst: Broadcast, Src: 0x953f

Figure 19. Captured device initialization process for ZigBee network.

A further experiment was conducted with the following scenario. The ConBee II coordinator management software, deCONZ, allows use of one out of four ZigBee network security modes:

- without protection;
- with standard security based on a trusted network key;
- with standard security based on a user-defined network key;
- with higher security involving the use of a trusted key for a given connection and without a slave–master relationship.

The phase of adding a new end device to the ZigBee network was analyzed for each of the above settings. Regardless of the settings, the network key was sent in the Key-Transport Key message and was encrypted using the standard ZigBeeAlliance09 transport key. Then, it was possible to decrypt messages using Wireshark. In the Wireshark options for ZigBee analysis, the network layer security level was configured with

- selection of cryptographic suite to AES-128 with 32-bit Integrity Protection,
- change of the value of the defined keys (*Pre-configured Keys*) with adding the key ZigBeeAlliance09 in the byte representation: 39:30:65:63:6E:61:69:6C:6C:41:65:65:42:67:69:5A,
- selection of Byte Order field as Reverse.

As a result, the decoded key 27c1405317ebfe3ffd07962db5712ec8 should appear in the Key field. It is presented in Figure 20. This key is the same as the key set in the deCONZ application on the Home Assistant platform, which is depicted in Figure 21. Based on this finding, it is recommended that any IoT manufacturer use the higher level security standards than the default ones. Consequently, decoding the network key would become more complicated for any attacker.

No.	Time	Source	Destination	Key Id	Info
12	15.204773	0x0000	Broadcast	Network Key	Data, Dst: Broadcast, Src: 0x0000
13	16.402633	0x0000			Beacon, Src: 0x0000, EPID: dresden_ff:ff:08:08:3b
14	18.203205				Ack
15	18.487270				Ack
16	18.487066	00:21:2e:ff:ff:08:08:3b	a4:c1:38:00:18:27:06:8f		Association Response, PAN: 0x7113 Addr: 0x836f
17	18.516640	0x0000	0x836f	Key-Transport Key	Transport Key

Frame 17: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0 IEEE 802.15.4 Data, Dst: 0x836f, Src: 0x0000 ZigBee Network Layer Data, Dst: 0x836f, Src: 0x0000 ZigBee Application Support Layer Command Frame Control Field: Command (0x21) Counters: 76 ZigBee Security Header Command Frame: Transport Key Command Identifier: Transport Key (0x05) Key Type: Standard Network Key (0x01) Key: 27c1405317ebfe3ffd07962db5712ec8 Sequence Number: 0 Extended Destination: TelinkSe_00:18:27:06:8f (a4:c1:38:00:18:27:06:8f) Extended Source: dresden_ff:ff:08:08:3b (00:21:2e:ff:ff:08:08:3b)					
--	--	--	--	--	--

Figure 20. Decoded ZigBee network key in Wireshark.

Figure 21. Network key setup in deCONZ, the same as decoded from the sniffed network traffic.

5.3. IoT Security Analysis Methodology Applied

Sections 5.1 and 5.2 presented an extensive examination of testing scenarios and a corresponding analysis based on the selected example devices. This examination procedure was further extended to a wider range of devices through the application of the aforementioned testing scenarios. It was important to apply the presented methodology to both IoT end devices and IoT bridges. The analysis beyond Gosund SP111, IG019 and ConBee II presented by Sections 5.1 and 5.2 included

- IoT end devices: Neo SmartBulb Wi-Fi RGB,
- IoT bridges: Phillips Hue, Aqara M2, Modemix W-WG001, Ikea Dirigera, and Sonoff Zig-Bee.

The results of these analyses are summarized and presented in Table 3. This comprehensive analysis demonstrated the significance of implementing stringent security protocols in devices, particularly with respect to safeguarding against the abuse of communication channels and the risk of firmware overwriting in the embedded memory. These findings should act as a catalyst for manufacturers to undertake proactive steps towards securing their devices and protecting against malevolent breaches.

Table 3. Summary of tested device vulnerabilities using proposed scenarios.

Device	Type	Scenario			
		Dumping Firmware	Flashing Firmware	Communication Channels Sniffing	RF Sniffing
Gosund SP111	Smart Plug	-	-	-	N/A
IG019	Smart Plug	-	-	-	-
Neo SmartBulb Wi-Fi RGB	Smart Bulb	-	-	+	N/A
Philips Hue	Bridge	-	N/A	N/A	-
ConBee II	Bridge	N/A	N/A	N/A	-
Aqara M2	Bridge	N/A	N/A	+	-
Modemix W-WG001	Bridge	N/A	N/A	+	-
Ikea Dirigera	Bridge	N/A	N/A	+	-
Sonoff ZigBee	Bridge	N/A	N/A	N/A	-

(-)—vulnerability found, (+)—lack of vulnerability, (N/A)—not applicable *. * it does not indicate that the device is not vulnerable, but rather that the prepared scenario does not apply to that specific device.

The experiments described in Sections 5.1.1 and 5.1.2 revealed a significant vulnerability in the architecture of consumer IoT solutions, such as the tested Gosund SP111 smart plug, as well as others based on the ESP platform. Primarily, it was straightforward to discover the reprogramming interface of the device, and thus uploading modified firmware was not problematic. Additionally, there are large communities of enthusiasts for these systems who can provide a wealth of advice and instructions on how to perform such modifications. It is worth noting that users rarely verify the software they download for such devices, increasing the risk of encountering modified software in deployments. Another distribution method may be the resale of such smart plugs after modification has been performed. This makes it easy to gain unauthorized access to wireless networks in different networks. It should also be noted that in the experiment, in addition to the wireless network identifier and password, the IP address was also stolen, from which the connection was established. It should also be considered that it may be possible to develop more advanced modifications that would turn the smart plug into a relay point or a communication server for an attacker. This is especially useful for bypassing firewalls in networks that only allow incoming traffic when a connection is established from within. This leads to the possibility of implementing more advanced scenarios where cyberattacks can be carried out against other devices in the local network. In recent years, this has been used with IoT cameras [64]. This highlights the generally recognized issue of the cybersecurity of the supply chain for IoT solutions [27] linked with the broad problem of user awareness in cyberspace. This openness of solutions, on the one hand, has many advantages, such as strong community interest, the availability of support, and unlocking of new features. On the other hand, awareness must be increased in terms of the potential use of such simple, cheap, and easily accessible IoT systems as part of implementing larger cybersecurity scenarios. In this area, several important works have emerged in recent times, including proposals for standards aimed at increasing the cybersecurity of end solutions, including the IoT, such as the ENISA [27] study.

The experiments described in Section 5.2.1 revealed that there are messages with sensitive data in wired and wireless communication channels. It is noteworthy that all logs were displayed within a short time frame (several seconds). Upon connecting to the internet, the device was able to download code from the manufacturer's website, without even allowing an opportunity for confirmation. Given the device has full access to the private network, attention should be paid to the threat posed by unscrupulous cloud solution providers. In [65], the possibility of exploiting this update channel to import malicious software into all connected devices to create a malicious botnet was raised. From a monitoring and data analysis perspective for cybersecurity, such a 1 MB firmware in reference to the scale of traffic processed by network devices is negligibly small and transmitted in encrypted traffic. This means that reliance cannot be placed on a network threat detection solution, making research such as that presented in this article and its conclusions for cybersecurity particularly important. It is also worth mentioning that the UART interface, being a debugging channel for the device, is not locked for use in production. Although passwords do not appear in messages, they can be correlated with wireless communication to provide an additional, unencrypted reference source.

Section 5.2.2 raises the problem of insecure transport for the ZigBee network key. The experiments carried out for different bridges showed that this vulnerability appears to not only be for ConBee II. The transmitted data cannot be sent using default passwords, because radio communication can be sniffed with ease. Additionally, by knowing modulation and encryption methods for specific protocols, decryption of the message is not complicated. This breaks the confidentiality of the ZigBee network traffic. Moreover, capturing the network key can lead to denial of service or replay attacks. An attacker can take control of the ZigBee network.

To improve the security of consumer IoT devices, manufacturers should implement secure firmware update processes to prevent unauthorized access and conduct regular security audits and testing of devices. Sensitive data stored on the device and during

transmission should be encrypted, and secure authentication methods should be provided for device access and firmware updates. Additionally, clear warnings and instructions should be included for users on the importance of verifying software authenticity and updating firmware. To further enhance security, manufacturers should remove access to wired communication channels in the production version of devices and make it impossible to overwrite firmware in built-in memory. Moreover, sensitive data, which can reduce the security level of a system, need to be sent through a dedicated channel. Sensitive data ought to be encrypted with a specific algorithm chosen by the network administrator, or manufacturers should implement their own solutions.

6. Conclusions and Future Work

In this article, the focus was placed on highlighting the security concerns surrounding smart home devices. The case and need for such efforts were thoroughly described in Section 3. A few spectacular examples of security vulnerabilities in IoT smart home devices, popular among user around world, were referenced, such as [37,38]. The vulnerability in the KNX protocol has even been part of an official CISA alert [40].

Evaluating the security of IoT devices, especially in smart home applications, seems to be a very important topic, which will only increase in importance in the coming years. This paper addressed these challenges. It proposed a universal methodology for IoT security analysis together with the general design of a lab environment for execution of this methodology. These are presented in Sections 4.1 and 4.2. It was the unique value of this paper to combine the assessment process steps with the recommended tools, and then to present them in action through a set of experiments (Section 5). A universal testing environment with the proposed methodology was utilized to conduct a series of security tests on various devices. The initial tests were executed on standalone devices, such as smart bulbs and plugs. Then, the process was extended to IoT bridges, especially for the ZigBee wireless interfaces. The results of the tests were analyzed to uncover vulnerabilities and the layers of the attack surface for potential exploitation. The findings, presented in Section 5.3, were then used to formulate recommendations for manufacturers of smart home devices to enhance their security measures and protect against malicious attacks.

The formulated and tested methodology for IoT device security assessments showed its robustness and applicability. It can supplement the well-established ecosystem of standards, frameworks, and guidelines for IoT security by providing practical procedures for verification of requirements. The unique combination of the proposition of a methodology, the relevant tools, and procedures enables anyone to easily follow the study, emphasizing affordable options.

There are two main limitations of the current methodology and the prepared test bed for security assessment of IoT devices. First, the firmware analysis is mostly based on clearly readable strings that are easily decoded, without applying deeper reverse engineering methods. Then, it is limited to digging deeper into some parts of firmware to find hidden vulnerabilities. The second limitation of the current setup is that no subsystem has yet been developed within the test bed for automated fuzzing of IoT devices. Recently, this has been recognized as one of the key elements to scale security research efforts in the space of finding vulnerabilities. Future work will address these concerns.

The aim of this study was to raise awareness of the critical importance of ensuring the security of smart home devices and to provide practical recommendations for improvement in this area. The results of this research can provide valuable insights into the current state of security in smart home devices and highlight the need for further work in this field. Future work should involve the extension of this research in two ways:

- by testing of an even wider range of devices, optimization of the testing environment, and the development of additional testing procedures within the proposed methodology for smart home devices. This would broaden the results available for different IoT systems and in different settings.

- by applying other methodologies for security analysis of IoT devices to compare the results between them and the proposed methodology. This would offer a deeper understanding of the findings' robustness and applicability.

Author Contributions: Conceptualization and supervision, K.S., D.P. and M.M.; State of the art, J.B., K.M. and D.T.; Methodology, K.Z., K.M., D.T., M.M. and J.B.; Lab Design, K.Z., K.M., D.T., M.M., J.B., D.P. and K.S.; Investigation, K.M., K.Z., D.T. and M.M.; Validation, K.S., D.P., M.M. and J.B.; Writing, K.M., D.T., K.Z. and J.B.; Editing, J.B. and K.M.; Review, K.S., K.M. and J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by The Polish National Centre for Research and Development under project No. CYBERSECIDENT/488240/IV/NCBR/2021.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

References

1. Hanes, D.; Salgueiro, G.; Grossetete, P.; Barton, R.; Henry, J. *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*; Cisco Press: Indianapolis, IN, USA, 2017.
2. Ukil, A.; Bandyopadhyay, S.; Puri, C.; Pal, A. IoT healthcare analytics: The importance of anomaly detection. In Proceedings of the 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), Crans-Montana, Switzerland, 23–25 March 2016; pp. 994–997. [\[CrossRef\]](#)
3. De Michele, R.; Furini, M. IoT Healthcare: Benefits, issues and challenges. In *ACM International Conference Proceedings Series*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 160–164. [\[CrossRef\]](#)
4. Syafrudin, M.; Alfian, G.; Fitriyani, N.L.; Rhee, J. Performance analysis of IoT-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing. *Sensors* **2018**, *18*, 2946. [\[CrossRef\]](#)
5. Use of IoT Technology to Drive the Automotive Industry from Connected to Full Autonomous Vehicles. *IFAC-PapersOnLine* **2016**, *49*, 269–274. [\[CrossRef\]](#)
6. Risteska Stojkoska, B.L.; Trivodaliev, K.V. A review of Internet of Things for smart home: Challenges and solutions. *J. Clean. Prod.* **2017**, *140*, 1454–1464. [\[CrossRef\]](#)
7. Raza, A.; Ikram, A.A.; Amin, A.; Ikram, A.J. A review of low cost and power efficient development boards for IoT applications. In Proceedings of the 2016 Future Technologies Conference (FTC), San Francisco, CA, USA, 6–7 December 2016; pp. 786–790. [\[CrossRef\]](#)
8. Ojo, M.O.; Giordano, S.; Prociassi, G.; Seitanidis, I.N. A Review of Low-End, Middle-End, and High-End Iot Devices. *IEEE Access* **2018**, *6*, 70528–70554. [\[CrossRef\]](#)
9. Griffor, E.R.; Greer, C.; Wollman, D.A.; Burns, M.J. *Framework for Cyber-Physical Systems: Volume 1, Overview*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2017.
10. Xenofontos, C.; Zografopoulos, I.; Konstantinou, C.; Jolfaei, A.; Khan, M.K.; Choo, K.K.R. Consumer, commercial, and industrial iot (in) security: Attack taxonomy and case studies. *IEEE Internet Things J.* **2021**, *9*, 199–221. [\[CrossRef\]](#)
11. Ukwandu, E.; Ben-Farah, M.A.; Hindy, H.; Bures, M.; Atkinson, R.; Tachtatzis, C.; Andonovic, I.; Bellekens, X. Cyber-security challenges in aviation industry: A review of current and future trends. *Information* **2022**, *13*, 146. [\[CrossRef\]](#)
12. Lykou, G.; Moustakas, D.; Gritzalis, D. Defending airports from UAS: A survey on cyber-attacks and counter-drone sensing technologies. *Sensors* **2020**, *20*, 3537. [\[CrossRef\]](#)
13. Argaw, S.T.; Bempong, N.E.; Eshaya-Chauvin, B.; Flahault, A. The state of research on cyberattacks against hospitals and available best practice recommendations: A scoping review. *BMC Med. Inform. Decis. Mak.* **2019**, *19*, 10. [\[CrossRef\]](#)
14. Mocrii, D.; Chen, Y.; Musilek, P. IoT-based smart homes: A review of system architecture, software, communications, privacy and security. *Internet Things* **2018**, *1–2*, 81–98. [\[CrossRef\]](#)
15. Andrade, R.O.; Ortiz-Garcés, I.; Cazares, M. Cybersecurity attacks on Smart Home during COVID-19 pandemic. In Proceedings of the 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 27–28 July 2020; pp. 398–404.
16. Sapalo Sicato, J.C.; Sharma, P.K.; Loia, V.; Park, J.H. VPNFilter malware analysis on cyber threat in smart home network. *Appl. Sci.* **2019**, *9*, 2763. [\[CrossRef\]](#)
17. Edu, J.S.; Such, J.M.; Suarez-Tangil, G. Smart home personal assistants: A security and privacy review. *ACM Comput. Surv.* **2020**, *53*, 116. [\[CrossRef\]](#)
18. Ding, W.; Hu, H. On the Safety of IoT Device Physical Interaction Control. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, Toronto, ON, Canada, 15–19 October 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 832–846. [\[CrossRef\]](#)

19. Hsu, K.H.; Chiang, Y.H.; Hsiao, H.C. SafeChain: Securing Trigger-Action Programming From Attack Chains. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 2607–2622. [CrossRef]
20. Breve, B.; Cimino, G.; Deufemia, V. Identifying Security and Privacy Violation Rules in Trigger-Action IoT Platforms with NLP Models. *IEEE Internet Things J.* **2023**, *10*, 5607–5622. [CrossRef]
21. IEEE Std 802.15.4-2020 (Revised IEEE Std 802.15.4-2015); IEEE Standard for Low-Rate Wireless Networks. IEEE: New York, NY, USA, 2020; pp. 1–800. [CrossRef]
22. Z-Wave Alliance: Z-Wave Specifications. Available online: <https://z-wavealliance.org/development-resources-overview/specification-for-developers/> (accessed on 26 September 2023).
23. Bluetooth SIG: Bluetooth Specifications and Documents. Available online: <https://www.bluetooth.com/specifications/specs/> (accessed on 26 September 2023).
24. Fagan, M.; Megas, K.N.; Scarfone, K.; Smith, M. *Foundational Cybersecurity Activities for IoT Device Manufacturers*; US Department of Commerce, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2020. [CrossRef]
25. Fagan, M.; Megas, K.N.; Scarfone, K.; Smith, M. *IoT Device Cybersecurity Capability Core Baseline*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2020. [CrossRef]
26. European Union Agency for Network and Information Security. Baseline Security Recommendations for IoT in the Context of Critical Information Infrastructures. Available online: <https://op.europa.eu/en/publication-detail/-/publication/c37f8196-d96f-11e7-a506-01aa75ed71a1/language-en> (accessed on 26 September 2023).
27. Guidelines for Securing the Internet of Things—ENISA. Available online: <https://www.enisa.europa.eu/publications/guidelines-for-securing-the-internet-of-things> (accessed on 30 January 2023).
28. ISO/IEC 27400:2022 Cybersecurity—IoT security and privacy—Guideline. Available online: <https://www.iso.org/standard/44373.html> (accessed on 26 September 2023).
29. Chmiel, M.; Korona, M.; Koziol, F.; Szczypiorski, K.; Rawski, M. Discussion on IoT Security Recommendations against the State-of-the-Art Solutions. *Electronics* **2021**, *10*, 1814. [CrossRef]
30. OWASP Internet of Things Project—Top 10 Threats 2018. Available online: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10 (accessed on 26 September 2023).
31. Alrawi, O.; Lever, C.; Antonakakis, M.; Monroe, F. SoK: Security Evaluation of Home-Based IoT Deployments. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 1362–1380. [CrossRef]
32. YourThings Scorecard: Evaluating and Scoring Smart-Home Devices to Improve Security! Available online: <https://yourthings.info> (accessed on 4 December 2023).
33. Yang, J.; Sun, L. A Comprehensive Survey of Security Issues of Smart Home System: “Spear” and “Shields”, Theory and Practice. *IEEE Access* **2022**, *10*, 124167–124192. [CrossRef]
34. Aldahmani, A.; Ouni, B.; Lestable, T.; Debbah, M. Cyber-Security of Embedded IoTs in Smart Homes: Challenges, Requirements, Countermeasures, and Trends. *IEEE Open J. Veh. Technol.* **2023**, *4*, 281–292. [CrossRef]
35. Carney, M. Pentesting Hardware—A Practical Handbook. Available online: <https://github.com/unprovable/PentestHardware> (accessed on 26 September 2023).
36. Nozomi Networks. DJI Mavic 3 Drone Research Part 1: Firmware Analysis. Available online: <https://www.nozominetworks.com/blog/dji-mavic-3-drone-research-part-1-firmware-analysis> (accessed on 26 September 2023).
37. Bonaventura, D.; Esposito, S.; Bella, G. Smart Bulbs Can Be Hacked to Hack into Your Household. In Proceedings of the 20th International Conference on Security and Cryptography—SECURITY, INSTICC, Rome, Italy, 10–12 July 2023; SciTePress: Setúbal, Portugal, 2023; pp. 218–229. [CrossRef]
38. Amit Serper, R.Y. ‘FriendlyName’ Buffer Overflow Vulnerability in Wemo Smart Plug V2. Available online: <https://sternumiot.com/iot-blog/mini-smart-plug-v2-vulnerability-buffer-overflow/> (accessed on 26 September 2023).
39. NVD-CVE-2023-27217. Available from MITRE, CVE-2023-27217. 2023.
40. US Cybersecurity and Infrastructure Security Agency. Alert ICSA-23-236-01: KNX Protocol. Available online: <https://www.cisa.gov/news-events/ics-advisories/icsa-23-236-01> (accessed on 26 September 2023).
41. Vimal Jerald, A.; Rabara, S.; Bai, T. Internet of things (IoT) based smart environment integrating various business applications. *Int. J. Comput. Appl.* **2015**, *128*, 32–37.
42. Daş, R.; Ababaker, T. Design and Application of a Smart Home System Based on Internet of Things. *Eur. J. Tech. (EJT)* **2021**, *11*, 34–42. [CrossRef]
43. Özdoğan, E.; Daş, R. IoT based a Smart Home Automation System Design: Simulation Case. *Balk. J. Electr. Comput. Eng.* **2021**, *9*, 297–303. [CrossRef]
44. Das, R.; Tuna, G. Machine-to-Machine Communications for Smart Homes. *Int. J. Comput. Netw. Appl.* **2015**, *2*, 196–202.
45. Shelby, Z.; Hartke, K.; Bormann, C. *The Constrained Application Protocol (CoAP)*; RFC 7252; RFC Editor: Fremont, CA, USA, 2014. [CrossRef]
46. OASIS. MQTT Version 5.0. 2019. Available online: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html> (accessed on 26 September 2023).
47. Nielsen, H.; Mogul, J.; Masinter, L.M.; Fielding, R.T.; Gettys, J.; Leach, P.J.; Berners-Lee, T. *Hypertext Transfer Protocol—HTTP/1.1*; RFC 2616; RFC Editor: Fremont, CA, USA, 1999. [CrossRef]
48. Eddy, W. *Transmission Control Protocol (TCP)*; RFC 9293; RFC Editor: Fremont, CA, USA, 2022. [CrossRef]

49. Rescorla, E. *The Transport Layer Security (TLS) Protocol Version 1.3*; RFC 8446; RFC Editor: Fremont, CA, USA, 2018. [CrossRef]
50. Akestoridis, D.G.; Harishankar, M.; Weber, M.; Tague, P. 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, Linz, Austria, 8–10 July 2020; pp. 77–88.
51. Touqeer, H.; Zaman, S.; Amin, R.; Hussain, M.; Al-Turjman, F.; Bilal, M. Smart home security: Challenges, issues and solutions at different IoT layers. *J. Supercomput.* **2021**, *77*, 14053–14089. [CrossRef]
52. Makhdoom, I.; Abolhasan, M.; Lipman, J.; Liu, R.P.; Ni, W. Anatomy of Threats to the Internet of Things. *IEEE Commun. Surv. & Tutor.* **2019**, *21*, 1636–1675. [CrossRef]
53. Abdulla, A.I.; Abdulraheem, A.S.; Salih, A.A.; Sadeeq, M.; Ahmed, A.J.; Ferzor, B.M.; Sardar, O.S.; Mohammed, S.I. Internet of things and smart home security. *Technol. Rep. Kansai Univ.* **2020**, *62*, 2465–2476.
54. Claasen, T.A. An industry perspective on current and future state of the art in system-on-chip (SoC) technology. *Proc. IEEE* **2006**, *94*, 1121–1137. [CrossRef]
55. Wireshark, a Network Protocol Analyzer. Available online: <https://www.wireshark.org> (accessed on 30 April 2023).
56. ESP8266 Pinout Reference and How to Use GPIO Pins. Available online: <https://microcontrollerslab.com/esp8266-pinout-reference-gpio-pins> (accessed on 30 January 2023).
57. GitHub—Espressif/Esptool: Espressif SoC Serial Bootloader Utility. Available online: <https://github.com/espressif/esptool> (accessed on 30 January 2023).
58. GitHub—arendst/Tasmota: Alternative Firmware for ESP8266 with Easy Configuration Using webUI, OTA Updates, Automation Using Timers or Rules, Expandability and Entirely Local Control over MQTT, HTTP, Serial or KNX. Full Documentation at. Available online: <https://github.com/arendst/Tasmota> (accessed on 30 January 2023).
59. Dierks, T.; Rescorla, E. *The Transport Layer Security (TLS) Protocol Version 1.2*; RFC 5246; RFC Editor: Fremont, CA, USA, 2008. Available online: <http://www.rfc-editor.org/rfc/rfc5246.txt> (accessed on 26 September 2023).
60. NVD-CVE-2014-3566. Available from MITRE, CVE-2014-3566. 2014.
61. NVD-CVE-2020-1968. Available from MITRE, CVE-2020-1968. 2020.
62. Gutierrez, J.A.; Callaway, E.H.; Barrett, R.L. *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4*; IEEE Standards Association: New York, NY, USA, 2004.
63. Bloessl, B.; Leitner, C.; Dressler, F.; Sommer, C. A GNU radio-based IEEE 802.15.4 testbed. In *12. Gi/Itg Kuvs Fachgespräch Drahtlose Sensornetze (FGSN 2013)*; IEEE: New York, NY, USA, 2013; pp. 37–40.
64. Blank, R.M.; Gallagher, P.D. *Guide for Conducting Risk Assessments*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2012. [CrossRef]
65. Zhang, X.; Upton, O.; Beebe, N.L.; Choo, K.K.R. IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers. *Forensic Sci. Int. Digit. Investig.* **2020**, *32*, 300926. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.