POLITECNICO DI TORINO

Master's Degree in Computer Engineering
Cybersecurity Track

Politecnico
di Torino

Master's Degree Thesis

Validating a Threat Model for Smart Home Gateways

Supervisors

Prof. Fulvio CORNO

Dott. Luca MANNELLA

Candidate

Antonino SANTA ROSA

JULY 2023

Summary

The Internet of Things (IoT) is an interconnected network of devices that communicate and share data with each other over the internet. Among them, Smart Home systems give users the possibility to connect and control, within a residential environment, connected devices like smart bulbs, plugs, sensors, and more.

This master's thesis focuses on the validation of a threat model that applies to add-ons for Smart Home Gateways that are extensible through add-ons. A Smart Home Gateway is a device, or a software solution, used to control the Smart Home and its smart devices. The Smart Home Gateway investigated in this document is WebThings. It is an open-source project that was incubated at Mozilla for four years and then spun out as an independent project.

The starting point of this work is an already existing threat model that highlights some possible threats that can be introduced into a Smart Home Gateway by add-ons developed for it. To demonstrate that such threats can happen even on WebThings, a set of Proofs of Concept (PoCs) add-ons were produced.

Moreover, to demonstrate that those PoCs can also be the outcome of an inexperienced or careless programmer, they were validated through two different surveys. The first one, involves a team of experts, while the second one involves a larger population of users. The goal of the first survey was to evaluate whether the developed Proofs of Concept can reasonably be considered not malicious by design. In this regard, only the PoCs considered reasonable by the experts were then further validated through the second survey.

Thanks to this second survey, we collected some insights useful to understand whether these malfunctioning inside the developed code were easy to spot and/or consider accidental. Therefore, the preliminary results confirm that some of the presented threats can occur even if the programmer has no malicious intent.

Table of Contents

List of Tables

List of Figures

Listings

Glossary

rules engine The rules engine is a component of the WebThing Gateway. Its task is to instantiate rules and to implement actions. These actions are executed whenever the necessary conditions are met.

rules interface It is the Rules Engine's interface through which instantiate the rules and the actions.

virtual machine A Virtual Machine (A virtual machine (VM) is a software emulation of a physical machine that runs the operating system and apps as if they were in a physical one.

web thing A web thing is a physical or virtual device connected to the network that can be managed through a data communication protocol, e.g., HTTP.

Acronyms

API Application Programming Interface

DBMS Database Management System

DFD Data Flow Diagram

DOM Document Object Model

DoS Denial of Service

IDE Integrated Development Environment

IFTTT If This Than That

IoT Internet of Things

JWT `JSON` Web Token

MVC Model-View-Controller

OWM OpenWeatherMap

PoC Proof of Concept

SDL Security Development Lifecycle

SHG Smart Home Gateway

SPA Single Page Application

SSD Solid State Drive

TM Threat Model

UI User Interface

URI Uniform Resource Identifier

VM Virtual Machine

Chapter 1

Introduction

The Internet of Things (**IoT**) is an interconnected network that extends Internet support and services to control and interact with smart devices, sensors, home appliances, vehicles, factory machines, wearable devices, and so on, with minimal human interaction [1]. Figure 1.1 shows the typical connections and services that it is possible to achieve an IoT environment.

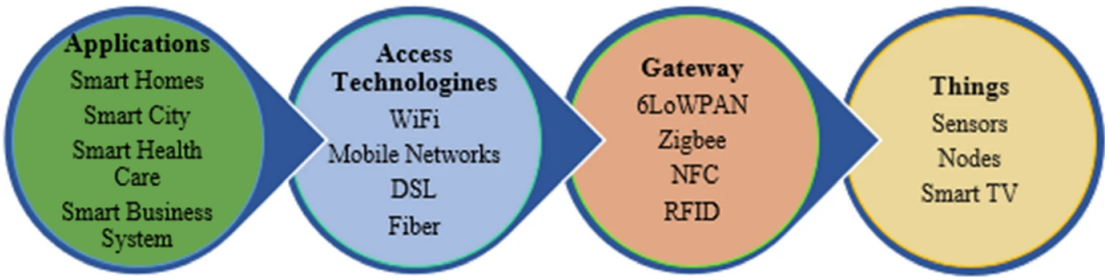

Figure 1.1: Typical IoT Environment

Moreover, IoT is the basis for Smart Homes. A possible definition of a **Smart Home** describes it as an IoT environment where heterogeneous electronic devices and appliances are networked together to provide services ubiquitously to individuals [2]. Hence, a Smart Home can provide innovative and smart services to the user and improve its quality of life, and it is a consequence of the increasing growth in IoT technology [3, 4].

Typical Smart Home devices are smoke sensors, light bulbs, plugs, doors, thermostats, etc. In this regard, it is interesting to think about how the number of connected devices increases day by day due to miniaturization, growth, and power availability, and this trend is predicted to lead to around 500 billion devices connected to the Internet by the year 2030 with a global mobile traffic estimation of 4394 EB/month [5]. Figure 1.2 shows the graphical representation of the described

trends.

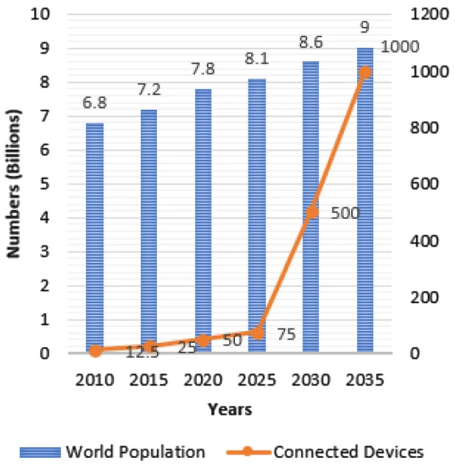

Figure 1.2: Estimated connected devices per world population [1]

Although Smart Homes have increased users' convenience, they are also valuable for attackers. In fact, IoT-enabled home appliances and Smart Home Gateways can be vulnerable to cyberattacks and create a point of entry to the Smart Home [6].

The proposed work focuses on the vulnerabilities that may arise in those Smart Homes where the control center is a Smart Home Gateway that is also **extensible** by means of **add-ons** (also known as *plug-ins* or *integrations*). Add-ons provide additional functionalities to the Smart Home components, a representation of which may be the one in Figure 1.3.

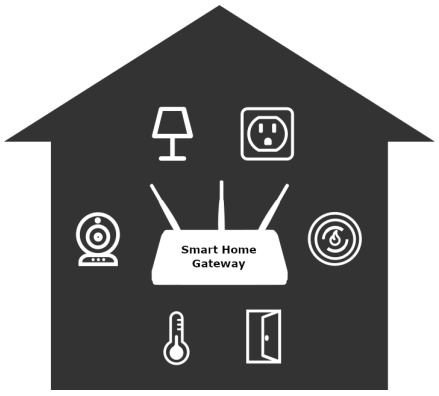

Figure 1.3: Smart Home components [7]

Starting from a threat model as a reference, the objective of this work is to understand whether add-ons developed by amateur or distracted programmers can

cause some vulnerability in the Smart Home. The extensible Smart Home Gateway that will be the target of evaluation is the one provided by WebThings, an open source platform that was incubated at Mozilla for four years before being spun out as an independent open source project [8, 7].

Going further, the following chapters will present in chapter 2 a brief background about Smart Home Gateways (focusing on the one provided by WebThings) and Threat Modeling. Then, in chapter 3, there will be a deep dive into the threat model used as a reference for the work and the related Proofs of Concept (**PoCs**) that have been developed to confirm the validity of the threat model guidelines. Afterwards, chapter 4 will discuss the making of two different surveys having the objective to validate the PoCs. Lastly, chapter 5 will discuss the outcome of the surveys and what could be the possible future work.

Chapter 2

Background

2.1 Smart Home Gateways

The **Smart Home Gateway** (**SHG**), also known as *smart home hub*, *smart hub*, *bridge*, *controller* or *coordinator*, is the control center that allows to monitor and control the Smart Home.

Examples of famous proprietary SHGs for commercial devices are Apple HomePod [9], Google Nest [10], Amazon echo [11], and Samsung SmartThings [12]. On the other hand, examples of free and open source SHGs are Home Assistant [13], WebThings [14], and openHAB [15]. In particular, this work will focus on the Smart Home Gateway provided by WebThings.

2.1.1 Web Things

Web Things is an open source platform for monitoring and controlling the Smart Home. The WebThings project was incubated at Mozilla for four years, before being spun out as an independent open source project [7]. The main components of the platforms are: **WebThings Gateway**, **WebThings Framework** and **WebThings Cloud**.

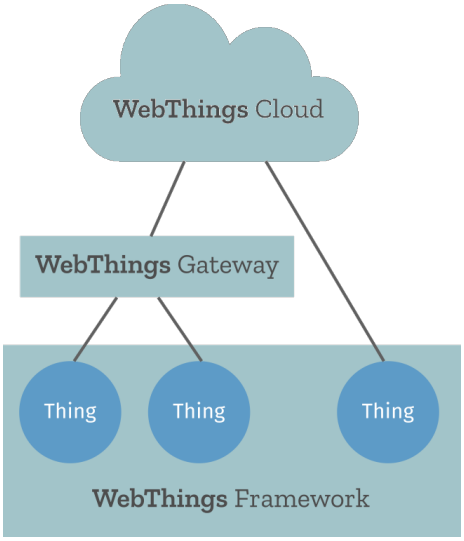

Figure 2.1: Platform overview [7]

2.1.1.1 Gateway

WebThings Gateway is the actual platform component through which the users can monitor and control their Smart Home over the web [16]. It provides:

- **Things UI**: a web user interface (**UI**) to monitor and control the smart home devices.

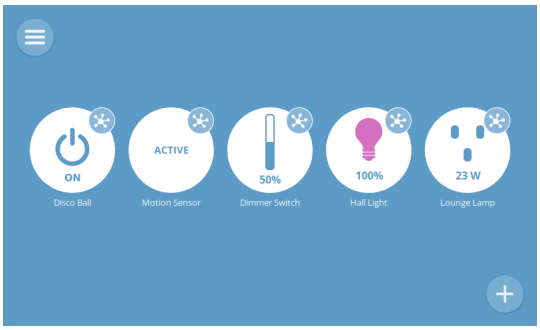

Figure 2.2: Things UI [16]

- **Rules Engine**: a simple drag-and-drop interface that provides the ability to to automate the Smart Home devices by creating rules through an "if this then that" (**IFTTT**)[1] style (e.g., if it is 7:00 PM, turn the hall light on).

[1]IFTTT: ifttt.com

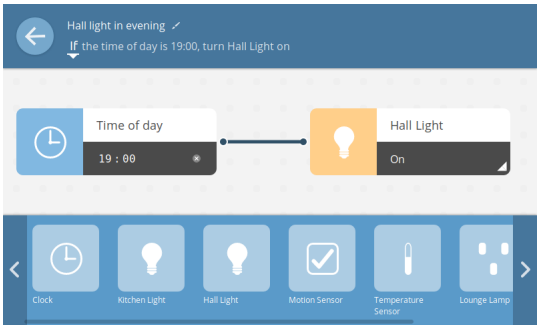

Figure 2.3: Rules Engine [16]

- **Floorplan**: an interactive floorplan of the home on which arrange the devices for a better at-a-glance status and control.

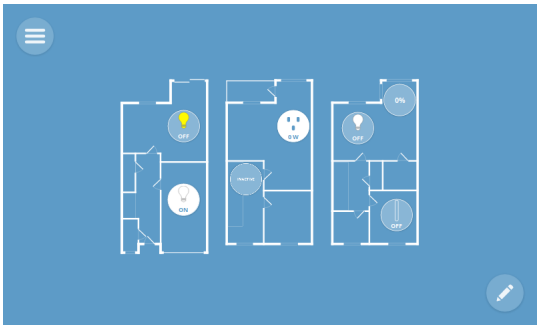

Figure 2.4: Floorplan [16]

- **Add-ons**: to extend the Gateway to support a wide range of existing smart home devices and protocols, but also to add new features to the UI.

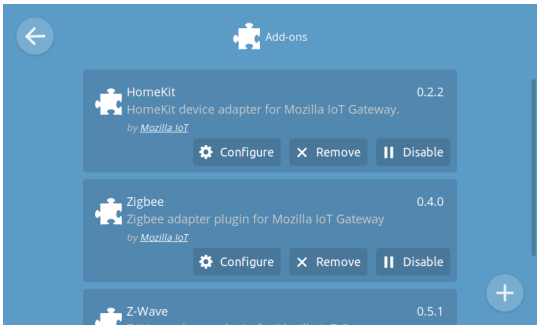

Figure 2.5: Add-ons [7]

2.1.1.2 Framework

WebThings Framework is a collection of reusable software components to help developers handle their **web things** [17]. The framework includes several libraries written in many programming languages, such as `Node.js`, `Python`, `Java`, `Rust`, `Arduino`, and `MicroPython`. Furthermore, the framework also cites third party libraries for other programming languages like `Go`, `IoT.js`, `C#`, and many more.

2.1.1.3 Cloud

WebThings Cloud provides cloud services for remotely managing the web things within the Smart Home [7]. Basically, it provides a client with a remote access service to the Smart Home using an end-to-end encrypted HTTPS tunnel between the Gateway and the client.

2.1.2 Gateway Architecture

The Gateway has two main parts [18]:

- The **back-end** server side: is based on **`Node.js`** [19] and uses **`Express` framework** [20] for routing.

- The **front-end** client side: is a single page web application (**SPA**), i.e., a web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of a web browser loading entire new pages. Here, routing is controlled via the lightweight **`page.js`** library [21].

2.1.2.1 Server Side Components

The server side, shown in Figure 2.6, can be divided into two main portions.

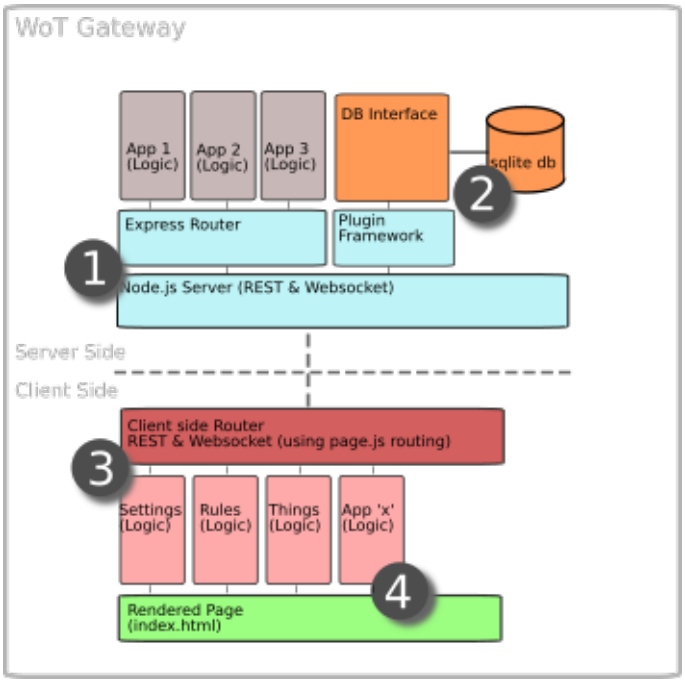

Figure 2.6: Gateway Architecture Overview [18]

In the first portion, when the `Node.js` server starts, the `Express` middleware is bound to the server. This starting point controls how WebSocket tunnels are created and closed in the Application.

The second server side portion is related to the Gateway's Database. By default, the application uses `SQLite` [22] as a `SQL` Database Management System (DBMS).

Lastly, the Gateway logic is based on the Model-View-Controller (**MVC**) paradigm [23] (i.e., a software architectural pattern that divides the related program logic into three interconnected elements), with only one main "view" served to the client as a home page.

2.1.2.2 Client Side Components

Again, keeping Figure 2.6 as a reference, it is possible to notice that the Gateway's client side has two main portions too.

Portion three of the client side focuses on routing. In the client, most calls from the main view are interpreted and routed using the `page.js` library [21]. Then, the router hands off control to the functions designed to handle the specific routes.

Portion four deals with the rendering of the Gateway's web page. As already said, the application is based on the SPA model. Pages are dynamically manipulated in the client side, and changes in the Document Object Model (**DOM**) for the

main view will be driven by hiding or showing different menu options. The server supplies data that populates the client.

2.1.3 Add-ons

The WebThings Gateway can be extended with **add-ons** that, once integrated, provide new features [24]. Each type of device (or service) needs the proper add-on to be installed and configured so that the Gateway can use it to discover devices and embed them as web things to interact with. In WebThings, there are three classes of add-ons: *adapter*, *notifier* and *extension*.

2.1.3.1 Adapter add-on

An **adapter** add-on provides support to the Gateway for some new class of devices (either physical or virtual) by adapting the device into a web thing. For instance, a Zigbee [25] (i.e., a popular industry wireless mesh networking standard for connecting sensors, instrumentation and control systems [26]) adapter communicates with Zigbee devices via the Zigbee protocol and represents them as web things having properties, actions and events.

However, an adapter add-on requires three main components: an adapter object, the device, and (at least) one property.

- The **adapter object** manages the communication with a device or a set of devices. For example, with all the smart devices using the same protocol for communicating over Wi-Fi.
- The **device** can be physical hardware, such as a smart plug, light bulb, or temperature sensor. Otherwise, it could be a virtual device, such as a weather station providing current weather conditions from a cloud service.
- **Properties** are individual traits of a device, such as its on/off state, its power consumption, or the color of its light.

2.1.3.2 Notifier add-on

A **notifier** add-on supplies a new output block for the rules engine of the Gateway, allowing a user to be notified of some type of occurrence via some specific mechanism, e.g., e-mail or SMS. A notifier add-on has two primary components: the notifier object and the notification outlet.

- A **notification outlet** is an individual rule output responsible for performing the actual notification process.
- A **notifier** manages a set of notification outlets presented to the user through the rules interface. A notifier can own and control any number of outlets.

2.1.3.3 Extension add-on

An **extension** add-on provides new functionalities to the Gateway's web interface, from something that is purely an aesthetic change (e.g., a new theme), to creating new panels and menu entries that provide new functionalities. An extension add-on has one primary component, an extension object, and an optional API handler.

- The **extension object** is loaded at run-time to provide the desired new functionality.
- An **API handler** is a back-end resource that can extend the Gateway's REST API to provide more functionalities for that specific extension.

2.2 Threat Modeling

In Cybersecurity it is important to protect **assets** from threats. A **threat** [27] is an event that that exploits a **vulnerability** [27], i.e., a weakness of the asset, and can produce the loss of security properties. An **attack** [27] is a *deliberate* threat occurrence.

The concept of threat modeling is not new. It arose from fields like computer security, cryptography, and risk management many years ago. Threat modeling can be beneficial for any type of system since it involves understanding the complexity of the system and analyzing its representations to highlight concerns about its privacy and security properties giving as output all its possible threats [28, 29]. An early and frequent analysis is the best way to improve those properties. Threat modeling is also a well-defined approach for identifying and assessing, *proactively*, potential threats and vulnerabilities in systems, applications, and organizations. Hence, having a threat model permits anticipating and mitigating potential risks before a malicious actor can exploit them. With threat modeling, organizations can improve the overall security of their systems by making more thoughtful efforts toward security.

Microsoft played an important role in spreading and formalizing threat modeling. In this regard, threat modeling at Microsoft is a documented methodology since 1999 [30]. Moreover, Microsoft introduced its Security Development Lifecycle (SDL), which includes threat modeling, in the early 2000s as an integral part of its software lifecycle, as a key activity to ensure the security of its software products. Since then, many threat modeling methodologies and frameworks have been developed and refined by organizations and security experts. These methodologies typically imply realizing a threat model by looking at the system to assess as an adversary would do. It means looking at the system by identifying its assets, architecture, and potential threats and vulnerabilities. This is done in order to help designers to understand what to asses, but also how and from whom to protect it.

However, a threat modeling process can be summarized into four main phases [31]:

1. **Asset Identification**: it involves identifying security goals, modeling domains, and identifying valuable assets;

2. **Threat Enumeration**: it is focused on identifying threats and vulnerabilities. Also, who the possible attackers are and what their motivations are. Lastly, the resulting threats are enumerated and documented;

3. **Threat Prioritization**: it involves giving a score to the discovered threats and assessing risks. It is based on the results of the prior phase. This phase can be either considered an internal or external activity [32];

4. **Mitigation**: it aims at resolving threats and lowering the risk level by proposing security mitigations and verifying them.

There is no threat model recommended over another. The decision of which ones to pick should be based on the needs of the project and its specific concerns [33]. A couple of examples of very popular threat models are:

- **STRIDE** [34, 33, 35]: its name is an acronym coming from a set of threats, i.e., Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, and Elevation of privilege [36].

  – **Spoofing**: it happens when there an entity impersonates a different entity;

  – **Tampering**: it is the unauthorized alteration of data (either in transit or stored);

  – **Repudiation**: it happens when someone can deny an action after performing it;

  – **Information disclosure**: it is the spread of confidential information to unauthorized parties;

  – **Denial of Service (DoS)**: it is an attack that prevents a system from operating as it should;

  – **Elevation of Privilege**: it happens when an entity gets higher system privileges than it should.

  It is adopted by Microsoft since 2002. It has a high degree of maturity and evolved to include new threats and new threat enumeration methodologies such as the *STRIDE-per-element* and the *STRIDE-per-interaction*. In particular, the former method is appropriate in systems where each software component to

be scanned for potential threats is considered in isolation. The latter method considers the security threats that might occur in a pair of interacting software components. Hence, it is more appropriate to discover threats in end-to-end scenarios where several components interact.

A STRIDE based threat modeling tool acts in two steps [37]. In step one, it takes as input the Data Flow Diagram (DFD) of the system and evaluates the system design having the goal of modeling it. Step two consists of the actual threat discovery and analyzing the modeled system. The two STRIDE variants differ in how the exploration of the modeled system is carried out;

- **PASTA** [33, 38]: it is a risk-centric threat modeling framework focused on attackers' perspective. PASTA is an acronym that stands for Process for Attack Simulation and Threat Analysis. Developed in 2012, it analyzes threats to business logic [39] and involves seven stages, i.e., *objectives definition*, *technical scope definition*, *application decomposition*, *threat analysis*, *vulnerability & weaknesses analysis*, *attack modeling*, and *risk & impact analysis*. Each stage uses a variety of design and analysis tools, e.g., DFDs are used in the application decomposition stage. In the end, the produced output is an assessment in the form of threat enumeration and scoring.

Hence, threat models can help create realistic and meaningful **security requirements**, and programmers can take a threat model as a reference to develop something that does not leave open doors for performing system attacks [31]. Moreover, a threat model helps to recognize what can go wrong in a system and to point out issues that can influence decisions in the subsequent design, development, testing, and post-deployment stages of the system. As already said, it is important starting fixing problems in the early stages because the cost of fixing a defect that can be corrected in the requirements stage would increase exponentially in the following phases [40].

Overall, nowadays threat modeling is an essential practice in the field of cybersecurity and risk management, helping organizations proactively address the security and resilience of their systems and applications.

Chapter 3

Developed Proofs of Concept

3.1 Reference Threat Model

This thesis focuses on further validating a reference Threat Model (**TM**) [41]. To this purpose, I developed a set of add-ons for a `JavaScript`-based Smart Home platform (WebThings). While this section focuses on presenting the reference TM, these Proofs of Concept will be presented in the next one (section 3.2). After illustrating that these threats can occur, the thesis proceeds to demonstrate they can be caused by inexperienced or careless programmers. This demonstration was done through two surveys, both presented in chapter 4.

The reference TM regards extensible Smart Home Gateways (SHGs) and only considers menaces originating from add-ons. The cited TM identifies *11 threats*, coming from add-ons, that may target the main components of the SHG. In this regard, an **attack target** can be *another add-on*, *the SHG application itself*, *applications running alongside the SHG*, *the gateway's operating system*, and *devices controlled by the SHG*. The TM also gives some hints about what some possible implementation of these threats could be. Hence, this TM helps developers to understand possible attacks against the system and not develop add-ons that act like the presented ones. A core concept in this TM, referred to by some threats, is the **add-on's scope**. It can be seen as the set of resources (e.g., access tokens, configuration files, data folders, etc.) an add-on is supposed to access legitimately.

In the following subsections, each threat will be labeled with the uppercase letter "T" plus an incremental integer (e.g., T1) to reference them quickly.

3.1.1 Confidentiality

Confidentiality is a security property that covers the related concepts of data confidentiality and privacy. **Data confidentiality** assures that private or confidential information is not made available or disclosed to unauthorized individuals.

Privacy assures that individuals control or influence what information related to them may be collected and stored and, by whom and to whom that information may be disclosed.

Therefore, a loss of confidentiality is the *unauthorized* disclosure of information. Hence, threats that menace this security property originate from add-ons that may access private data outside their scope. These threats are:

- **T1**: an add-on accesses and uses private data belonging to an attack target. Hence, data outside its scope;
- **T2**: an add-on accesses data outside its scope that belong to the attack target, and then it spreads these data.

3.1.2 Integrity

Integrity property covers the related concepts of data integrity and system integrity. **Data integrity** assures that information (both stored and in transit) and programs are changed only in a specified and authorized way. **System integrity**: assures that a system performs its intended function in an unimpaired way, i.e., free from deliberate or inadvertent *unauthorized* system manipulation.

Accordingly, a loss of integrity arises from the unauthorized modification or destruction of information. In this sense, threats are:

- **T3**: an add-on alters the state of Smart Home devices that are not supposed to be in its scope;
- **T4**: an add-on alters private data of an attack target outside its scope (e.g., by overwriting a measured power consumption).

3.1.3 Availability

Availability property assures that a system works promptly and its services are not denied to authorized users. Accordingly, a loss of availability implies the impossibility of ensuring timely and reliable access to and use of information or an information system. In this sense, threats are:

- **T5**: an add-on delays the regular functionality of another Smart Home component;
- **T6**: an add-on alters one of the regular functionalities of another Smart Home component;
- **T7**: an add-on alters the regular functionality of another Smart Home component, preventing Smart Home users from using it;

- **T8**: an add-on physically damages an attack target in the Smart Home (e.g., the SHG or a device).

3.1.4 Authentication

Authenticity is the property of being able to be verified and trusted. This can mean verifying that users are who they say they are and that each input arriving at the system came from a trusted source. In this regard, **authentication** implies the identification of the actors in the system.

There are different definitions of authentication:

- **RFC-4949** (Internet Security Glossary) [42]: the process of verifying a claim that a system entity or system resource has a certain attribute value;
- **FIPS PUB 200** (Minimum Security Requirements for Federal Information and Information Systems) [43]: verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system.

However, the important point of those definitions is that they define the authentication of *an actor*, meaning that it could be not only a human being (interacting via software running on hardware) but also a software component or a hardware element (interacting via software).

A threat to authentication is:

- **T9**: an add-on interacts with a system component pretending to be a different entity;

3.1.5 Authorization

Authorization is the process for determining whether an entity is authorized to perform a given activity or gain access to the system resources or services.

A threat to authorization can be:

- **T10**: an add-on accesses an authorization level higher than it should be;

3.1.6 Non-repudiation

Non-repudiation, in a general information security context, is the assurance that who sends the information is provided with proof of delivery, and who receives the information is provided with proof of the sender's identity, so that none of the

parties can later deny having process the information [44]. In this sense, a threat is:

- **T11**: an add-on *anonymously* communicates with an attack target, so that there is no way to tell with certainty who were the parties involved in the communication;

3.2 Proofs of Concept

3.2.1 Premise

The following PoCs have been developed and tested on version 1.0.1 of WebThings Gateway, i.e., the latest stable version of the SHG released at the time these PoCs were developed. Moreover, the SHG was directly on a UTM[1] virtual machine (VM) running Ubuntu ARM 22.04[2]. These PoCs are released as open-source code under my personal GitHub profile[3].

Moreover, each of the following Proofs of Concept (PoCs) was developed to try to reproduce programming errors that a distracted or novice programmer could have done, leading to the threat occurrence. Hence, a group of experts was involved to validate these PoCs through a survey (see section 4.1). The goal of the survey was to understand whether experienced programmers consider those errors the outcome of an inexperienced or careless programmer and not malicious by design. Then, after the approval of the experts, the PoCs were submitted, through another survey, to a larger population of users for an assessment (see section 4.2).

The following PoCs were developed using as a starting point add-ons that can be found among the WebThings' add-on list[4]. This list would likely be the starting point for a novice programmer in the development of new add-ons or the customization of already existing ones. These add-ons were then heavily customized to bring the desired functionalities not provided by the starting add-ons. However, the starting point for developing each PoC that is an extension (see subsubsection 2.1.3.3) was an add-on developed by the core team[5] and made available by them to be used for this purpose for creating new extensions. Also, each PoC dealing with a weather station is a personal customization of a standard

[1]UTM: docs.getutm.app

[2]Ubuntu ARM: ubuntu.com/download/server/arm

[3]PoCs source code: github.com/ninosanta/master-degree-thesis/PoCs

[4]WebThingsIO addon-list: github.com/WebThingsIO/addon-list/addons

[5]`example-extension`: github.com/WebThingsIO/example-extension

adapter[6] that was extended with new functionalities. Lastly, each PoC dealing with smart plugs was developed starting from an add-on developed by the core team for testing devices[7].

3.2.2 T1 & T2 - `weather-adapter`

T1 and T2 were implemented using the same add-on called `weather-adapter`. It is an adapter add-on (see subsubsection 2.1.3.1) that provides the SHG with a virtual weather station that uses OpenWeatherMap[8] (**OWM**) as a provider to retrieve weather data.

Figure 3.1 shows the virtual weather station in the SHG dashboard.

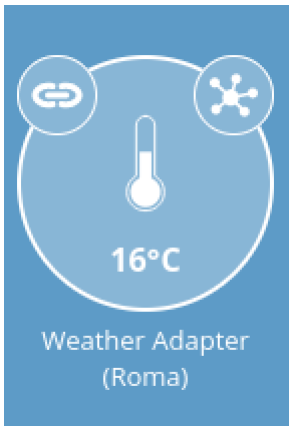

Figure 3.1: Weather Adapter

The adapter requires an `API Key` to use the OWM `APIs` and query the provider to receive the weather data. The user can choose to use a default `API Key` (basically a *free* one) or a personal one (likely a *premium* one). In subsubsection 3.2.2.1, to implement T1 (see subsection 3.1.1), will be considered just the case of a user using a default `API Key` written in a text file named "`default.txt`". In subsubsection 3.2.2.2, to implement T2 (see subsection 3.1.1) will be considered the case in which the user decides to use a *personal* `API Key` to exploit the OWM `APIs` and query the provider to receive the weather data updates.

[6]Standard `weather-adapter`: github.com/WebThingsIO/weather-adapter

[7]Standard `virtual-things-adapter`: github.com/WebThingsIO/virtual-things-adapter

[8]OpenWeatherMap: openweathermap.org

3.2.2.1 T1 - Demonstration

As described in subsection 3.2.2, a user can write its own `API Key` in a text file named "`default.txt`". This file is stored in the data path of the adapter (i.e., `~/.webthings/data/weather-adapter`). Otherwise, if the user does not use such a personal default `API Key`, the adapter itself will put inside `default.txt` a hard-coded key and will use this `API Key` to operate.

The above description is the desired behavior of the adapter. Instead, what happens in this Proof of Concept is that the adapter uses a data path that is `~/.webthings/data/adapter-weather`, but it should have been `~/.webthings/data/weather-adapter`. The following Listing 3.1 shows what was just explained.

Listing 3.1 T1 - Wrong data path

```
/* API Key location*/
const baseDir = path.join(
    os.homedir(),
    ".webthings",
    "data",
    "adapter-weather"  // Error: wrong add-on's directory
);

const defaultToken = path.join(
    baseDir,
    "default.txt"
);

/* Code removed for ease of reading */

if (!fs.existsSync(defaultToken)) {
    fs.writeFileSync(
        defaultToken,
        "XXX2a5XX72f5e832cXXXb708e3XXXe0X"
    );
}
```

Hence, under the hypothesis that in the SHG there is another virtual weather station (e.g., maybe to check the weather of a set of specific locations or to not reach the limit of queries allowed by a single `API Key`) implemented through an adapter called `adapter-weather`, `weather-adapter` will use the default `API Key` of `adapter-weather`. Figure 3.2 shows the two virtual weather stations.

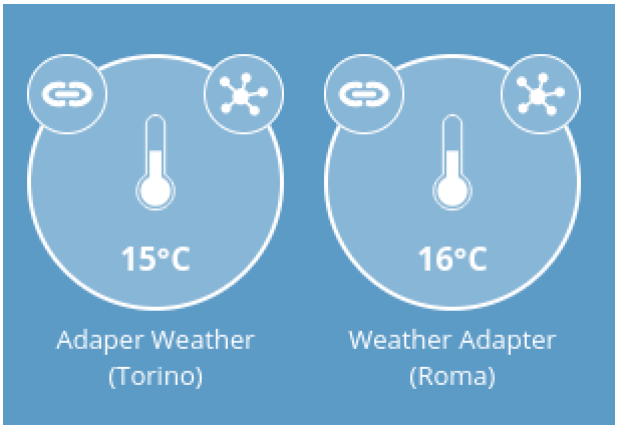

Figure 3.2: Adapter Weather and Weather Adapter

Listing 3.2 shows the rest of the code regarding the configuration of the wrong `API Key` that, once set (line 16), is bind to the virtual device (line 22).

Listing 3.3 shows the fix in the adapter's data path.

Listing 3.2 T1 - Wrong `API Key` configuration

```
1   class WeatherAdapter extends Adapter {
2       /* ... */
3       startPairing() {
4           /* ... */
5           let OWM_API_KEY = "";

7           if(this.config.useDefaultOpenWeatherMapApiKey === false
8               && this.config.apiKey !== "") {
9               /* ... */
10          } else {
11              OWM_API_KEY = fs.readFileSync(
12                  defaultToken,
13                  "utf8"
14              );
15          }
16          this.config.apiKey = OWM_API_KEY;

18          const dev = new WeatherDevice(
19              this,
20              location,
21              this.config.units,
22              this.config.apiKey,
23              this.config.pollInterval,
24          );
25          dev.promise.then(() => this.handleDeviceAdded(dev));
26      }
27  }
```

```
28      /* ... */
29  }
```

Listing 3.3 T1 & T2 - Correct data path

```
1   /* API Key location*/
2   const baseDir = path.join(
3       os.homedir(),
4       ".webthings",
5       "data",
6       "weather-adapter"  // Fix: correct data path
7   );
```

3.2.2.2 T2 - Demonstration

In WebThings, each add-on has its own manifest configuration file [45]. In this case, a `JSON Schema` within the `manifest.json` configuration file permits to specify, through the SHG add-on configuration panel, the value of an `API Key` different from the "default" one. This `API Key` will be checked against the content of a file named "`api-key.txt`" that is stored in the data path of the adapter. If the file already contains an `API Key` different from the new one, the older one is backed up on Dropbox[9] for future uses and then overwritten by the new one.

The adapter here uses the wrong path to retrieve the `API Key` (as we have already seen in subsubsection 3.2.2.1), namely `weather-adapter` retrieves the "`api-key.txt`" file from `~/.webthings/data/adapter-weather` instead of `~/.webthings/data/weather-adapter`. Listing 3.1 shows the wrong data path in the code.

However, T2 arises if the "`api-key.txt`" file belonging to `adapter-weather` already has an `API Key` inside. In fact, that key would be backed up on Dropbox by `weather-adapter` even though it belongs to `adapter-weather` and not to `adapter-weather`. Figure 3.2 shows the weather stations.

The code in Listing 3.4 shows how the back up of the wrong `API Key` is done. Listing 3.3 shows the fix in the adapter's data path.

Listing 3.4 T2 - Key update on Dropbox

```
1   class WeatherAdapter extends Adapter {
2       startPairing() {
3           /* ... */
4           let OWM_API_KEY = "";

6           if(/* ... */ this.config.apiKey !== "") {
```

[9] Dropbox: dropbox.com

```
7               if (!fs.existsSync(apiKey)) {
8                   fs.writeFileSync(apiKey, this.config.apiKey);
9                   OWM_API_KEY = this.config.apiKey;
10              } else if (fs.existsSync(apiKey)
11                  && fs.readFileSync(apiKey, "utf8") !== this.config.apiKey) {
12                  if (this.config.dbxToken !== undefined) {
13                      /* Old API Key backup on Dropbox */
14                      const old_key = fs.readFileSync(apiKey, "utf8");  // Error: apiKey has
        the wrong key's path
15                      /* ... */
16                      dbx.filesUpload({
17                          path: `/userKey_${new Date()}.txt`,
18                          contents: old_key
19                      })
20                      .then( /* ... */)
21                      .catch( /* ... */);
22                  } else { console.error(/* ... */); }
23                  fs.writeFileSync(apiKey, this.config.apiKey);
24                  OWM_API_KEY = this.config.apiKey;
25              } else {
26                  OWM_API_KEY = fs.readFileSync(apiKey, "utf8");
27              }
28          } else { /* ... */ }
29          this.config.apiKey = OWM_API_KEY;

31          const dev = new WeatherDevice(
32              this,
33              this.config.apiKey,
34              /* ... */
35          );
36          /* ... */
37      }
38  }
39  /* ... */
40 }
```

3.2.3 T3 - `lights-off-extension`

The `lights-off-extension` is an extension add-on that implements T3 (see subsection 3.1.2). It shows in the front-end of the Smart Home Gateway a button to click to turn off every light bulbs in the Smart Home, as shown in Figure 3.3.

Figure 3.3: Lights off Extension front-end

However, in the code, there is an array containing the web things of the SHG. The array is not filtered through the `filter()` function to just get the lights among all the web things. Later, each entry of the array having the `on()` property is turned off. Nevertheless, this does not imply turning off every light because there are many others web things that have this property too. Hence, this extension turns off every thing having the `on()` property, e.g., also switches or plugs if present, and this leads to T3. Listing 3.5 shows the wrong implementation of the button click handling.

Listing 3.6 shows how to fix the elements of the array by discarding anything but lights.

Listing 3.5 T3 - Loop in charge of turning light bulbs off

```
/* ... */
off_button.addEventListener('click', () => {
    window.API.getThings().then((res) => {
        const jsonarray = res;  // Array of Things in JSON format -> Error: should be filtered

        for (let n = 0; n < jsonarray.length; n++) {
            let obj = jsonarray[n];
            if (obj.properties.on !== undefined) {
                let uri = obj.id + "/properties/on";
                const body = {on: false};
                window.API.putJson(uri, body)
                  .catch(/* ... */);
             }
        }
         pre.innerText = "Now it is everything off";
    }).catch(/* ... */);
});
/* ... */
```

Listing 3.6 T3 - Array Fixing

```
/* ... */
off_button.addEventListener('click', () => {
    window.API.getThings().then((res) => {
        const jsonarray = res
                .filter(obj => obj['@type'].includes('Light'));  // Fix: filtering
/* ... */
```

3.2.4 T4 - `smart-plugs-adapter`

The `smart-plugs-adapter` implements T4 (see subsection 3.1.2). It is an adapter add-on (see subsubsection 2.1.3.1). This add-on integrates up to two virtual smart plugs as web things in the SHG. Figure 3.4 shows how both plug appear in the dashboard of the SHG.

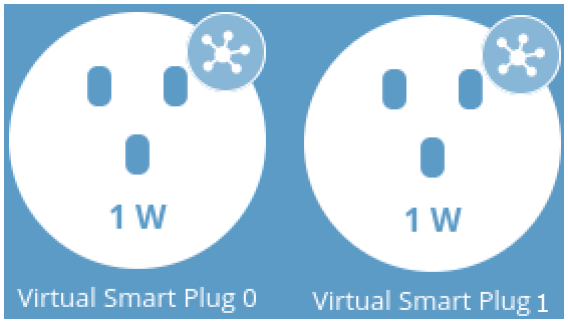

Figure 3.4: Plugs in the SHG Dashboard

Furthermore, this adapter regularly stores on file the instantaneous power consumption of each plug so that, for example, at the end of the day the user may have an idea of the overall consumption of each plug.

Listing 3.7 shows the adapter data path.

Listing 3.7 T4 - `smart-plugs-adapter` data path

```
/* smart-plugs-adapter */

const baseDir = path.join(
    os.homedir(),
    ".webthings",
    "data",
    "smart-plugs-adapter"
  );
```

Even though the adapter data path is correct, it is shared by both plugs. Therefore, if the two plugs are both working and they save the consumption data

on a file having the same name, as shown in Listing 3.8 (lines 11 and 19), the slowest plug to write that file will overwrite the file of the other one.

Listing 3.9 shows a possible fix that makes files unique.

Listing 3.8 T4 - `smart-plugs-adapter` files conflict

```
2  /* Property of a device */
3  class VirtualThingsProperty extends Property {
4      constructor(device, name, descr, value, interval) {
5          /* ... */
6          let i = 0;
7          if (this.name === 'instantaneousPower') {
8              switch(this.device.id) {
9                  case 'virtual-smart-plug-0':
10                     setInterval(() => {
11                         fs.writeFileSync(path.join(baseDir, `reading-n${i}.txt`),
12                             `${i} - ${new Date()} - ${this.device.id} -
13                                 ${(this.value*1000).toFixed(2)}mW`);
14                         i++;
15                     }, interval*1000);
16                     break;
17                 case 'virtual-smart-plug-1':
18                     setInterval(() => {
19                         // Error: the file name `reading-n${i}.txt` is the same as in the previous
       `case`!
20                         fs.writeFileSync(path.join(baseDir, `reading-n${i}.txt`),
21                             `${i} - ${new Date()} - ${this.device.id} -
22                                 ${(this.value*1000).toFixed(2)}mW`);
23                         i++;
24                     }, interval*1000);
25                     break;
26             }
27         }
28     }
29     /* ... */
30 }
```

Listing 3.9 T4 - `smart-plugs-adapter` fixing file names

```
1  /* ... */
2      case 'virtual-smart-plug-0':
3          setInterval(() => {
4              // Fix: ${this.device.id} in the file name makes it unique for each plug
5              fs.writeFileSync(path.join(baseDir,
6                  `${this.device.id}-reading-${new Date()}.txt`),
7              `${i} - ${new Date()} - ${this.device.id} - ${(this.value*1000)
       .toFixed(2)}mW`);
8              i++;
```

```
9         }, interval*1000);
10        break;
11    case 'virtual-smart-plug-1':
12        setInterval(() => {
13            fs.writeFileSync(path.join(baseDir, `${this.device.id}-reading-${new Date()}
      .txt`),
14                `${i} - ${new Date()} - ${this.device.id} - ${(this.value*1000)
      .toFixed(2)}mW`);
15            i++;
16        }, interval*1000);
17        break;
18    /* ... */
19 }
```

3.2.5 T5 - `things-off-extension`

The `things-off-extension` implements T5 (see subsection 3.1.3). It is an extension add-on (see subsubsection 2.1.3.3). The add-on shows in the front-end of the Smart Home Gateway a button "OFF" that when clicked turns off every web thing in the Smart Home, i.e., every device controlled by the SHG having the `on()` property. Figure 3.5 shows the extension in its pane in the SHG.

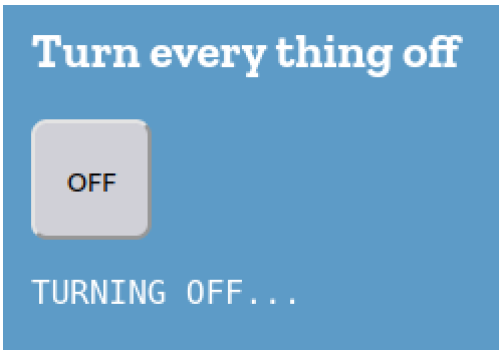

Figure 3.5: Things off Extension front-end

However, this extension updates the Uniform Resource Identifier (URI) passed to the SHG's API in charge to update the `on()` property of each web thing. This property is updated only if the thing in the array has the property `on()`. However, for each iteration, if the URI is not empty, it sends anyway the request to set the property `on()` to `false` using the last URI that has been set. Therefore, the last thing that was shut down keeps being shut down until the URI is updated again or until the end of the array is reached. Consequently, if the array contains a large number of things that do not have the `on()` property following a thing that has the `on()` property, the latter will receive a large amount of requests. Hence, trying

to switch on that thing again, while it is bombed of requests to keep it off, will result in T5. Because, this will imply that it will be turned on with some delay that will increase with the number of interposed things without the `on()` property. Listing 3.10 shows the implementation of what just described.

Listing 3.10 T5 - Loop in charge of turning things off

```
/* ... */
off_button.addEventListener('click', () => {
            window.API.getThings().then((res) => {
                const jsonarray = res;  // Array of Things in JSON format

                let uri = "";  // Error: this must be placed inside the loop
                const body = {on: false};
                for (let n = 0; n < jsonarray.length; n++) {
                    let obj = jsonarray[n];
                    if (obj.properties.on !== undefined) {
                        uri = obj.id + "/properties/on";
                    }
                    if (uri !== "") {
                        window.API.putJson(uri, body)
                            .then(() => {
                                if (n === jsonarray.length - 1) {
                                    pre.innerText = "Now it is everything off";
                            }})
                            .catch(/* ... */);
                    }
                }
            }).catch(/* ... */);
        });
/* ... */
```

Note that the delay starts to become noticeable with about 500 things involved. This number, in a SHG like WebThings, is not so unrealistic because there can be many virtual o physical things, notifiers (see subsubsection 2.1.3.2), extensions (see subsubsection 2.1.3.3), and adapters (see subsubsection 2.1.3.1) in multiple instances.

Listing 3.11 shows a possible fix that consists in bringing the reset of the `uri` variable inside the loop.

Listing 3.11 T5 - Fix

```
    /* ... */
    const jsonarray = res;  // Array of Things in JSON format

    const body = {on: false};
    for (let n = 0; n < jsonarray.length; n++) {
        let uri = "";  // Fix: moved inside the loop
```

```
7         let obj = jsonarray[n];
8         if (obj.properties.on !== undefined) {
9             uri = obj.id + "/properties/on";
10        }
11    /* ... */
```

3.2.6 T6-v1 - `power-cons-extension`

The `power-cons-extension` is an extension add-on (see subsubsection 2.1.3.3). The add-on shows two buttons at the front end of the Smart Home Gateway, as shown in Figure 3.6. Clicking the *Calculate* button shows the global instantaneous power consumption of the Smart Home. Clicking the *Reset* button resets the instantaneous power calculus.

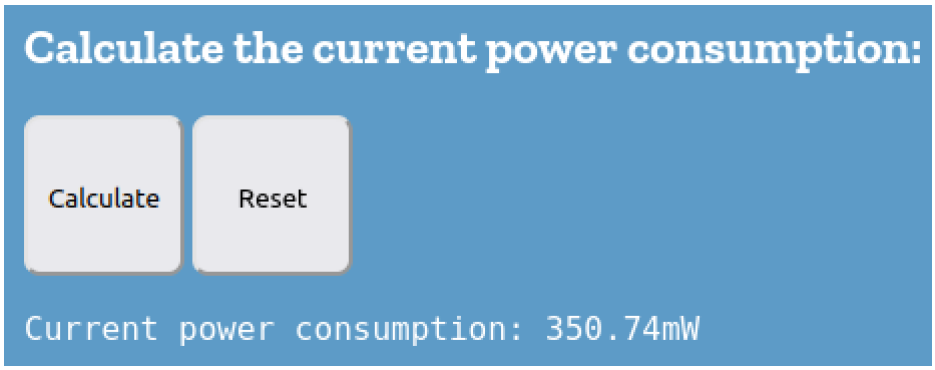

Figure 3.6: Power off Extension front-end

The portion of the extension that handles the reset leads to T6 (see subsection 3.1.3) because it is implemented in such a way that once the Reset button is clicked, it will temporarily alter the `instantaneousPower()` property, setting it to 0. Of course, this extension will work only on smart device implementations that have the `instantaneousPower()` property not set as `readOnly`, as shown in Listing 3.12. Moreover, differently from the other PoCs, here the error is more on a conceptual level than on a programming level.

Hence, Listing 3.13 shows how the reset event should be handled by just clearing the front-end.

Listing 3.12 T6-v1 - Reset Event

```
1  /* ... */
2  reset_button.addEventListener('click', () => {
3          window.API.getThings().then((res) => {
4            const jsonarray = res;  // Array of Things in JSON format
```

```
6           // Error: this loop resets the instantaneous power of the devices
7           for (let n = 0; n < jsonarray.length; n++) {
8             let obj = jsonarray[n];

10            if (obj.properties.instantaneousPower !== undefined &&
11              obj.properties.instantaneousPower.readOnly !== true) {
12              let uri = obj.id + "/properties/instantaneousPower";

14              const body = {instantaneousPower: 0};
15              window.API.putJson(uri, body)
16                .catch( /* ... */ );
17            }
18          }
19        });
20        pre.innerText = "0mW";
21      });
22    }
23  }
24  /* ... */
```

Listing 3.13 T6-v1 - Fixed Reset

```
1 /* ... */
2 reset_button.addEventListener('click', () => {
3     // Fix: removed the wrong code lines
4         pre.innerText = "";
5       });
6 /* ... */
```

3.2.7 T6-v2 - `plug-smart-adapter`

The `plug-smart-adapter` is an adapter add-on (see subsubsection 2.1.3.1). It integrates a virtual smart plug in the SHG and regularly stores its instantaneous power consumption data in a file. The adapter then reads all these data to calculate and show the average power consumption in the SHG dashboard among the device's properties.

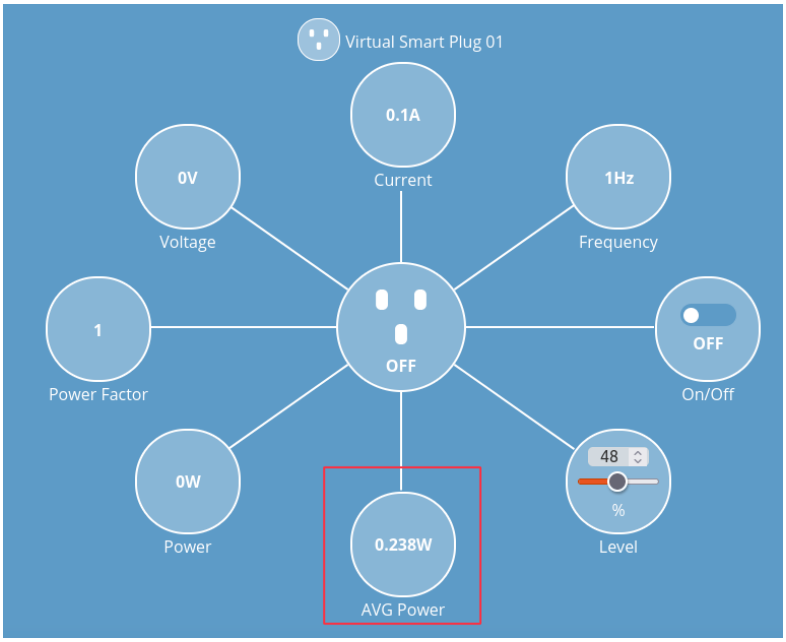

Figure 3.7: Properties of a smart plug integrated through `plug-smart-adapter`

This adapter leads to T6 (see subsection 3.1.3) because it has been implemented by using as a data path `~/.webthings/data/smart-plug-adapter`. The correct data path is `~/.webthings/data/plug-smart-adapter` instead. This error implies that if `smart-plug-adapter` exists and collects the same power consumption data in a file named as the one of `smart-plug-adapter`, the average power consumption output in the dashboard will be wrong for both the adapters. Listing 3.14 shows the implementation details.

A possible fix of the adapter is shown in Listing 3.15.

Listing 3.14 T6-v2 - Plug Smart Adapter

```
1  /* plug-smart-adapter */
2
3  const baseDir = path.join(
4    os.homedir(),
5    ".webthings",
6    "data",
7    "smart-plug-adapter"  // Error: wrong data path
8  );
9
10 /* Property of a device */
11 class VirtualThingsProperty extends Property {
12   constructor(device, name, descr, value, interval) {
13     /* ... */
```

```
15      if (/* ... */
16        && this.name !== 'averagePowerConsumption') {
17        /* ... */
18      } else if (this.name === 'averagePowerConsumption') {
19        this.interval = setInterval(() => {
20          let sum = 0;
21          const date = new Date();
22          const year = /*...*/, month = /*...*/, day = /*...*/;
23          let values = fs.readFileSync(
24            path.join(baseDir, `powerValues-${year}${month}${day}.txt`), 'utf-8')
25              .split('\n')
26              .filter(value => value !== '');
27          for (let i = 0; i < values.length; i++) {
28            sum += parseFloat(values[i]);
29          }
30          this.value = (sum / values.length).toFixed(3);
31          this.setCachedValue(this.value);
32          this.device.notifyPropertyChanged(this);
33        }, interval * 1000);
34      } else if (this.name === 'instantaneousPower') {
35        this.interval = setInterval(() => {
36          const date = new Date();
37          const year = /*...*/, month = /*...*/, day = /*...*/;
38          fs.appendFileSync(path.join(baseDir,
39                  `powerValues-${year}${month}${day}.txt`),
40            `${(this.value).toFixed(3)}\n`);
41        }, interval*1000);
42      }
43    }
44  }
45  /* ... */
```

Listing 3.15 T6-v2 - Data Path Fix

```
1  /* plug-smart-adapter */
2
3   const baseDir = path.join(
4     os.homedir(),
5     ".webthings",
6     "data",
7     "plug-smart-adapter"  // Fix: correct data path
8  );
9  /* ... */
```

3.2.8 T7 - `things-off-extension`

This version of `things-off-extension`, like the one presented in subsection 3.2.5, is an extension add-on (see subsubsection 2.1.3.3). The add-on shows in the front-end of the Smart Home Gateway an OFF button. This button, when clicked, turns off every web things in the Smart Home, i.e., every device controlled by the SHG having the `on()` property, with one second of delay from each other. Figure 3.8 shows the extension in the front-end.

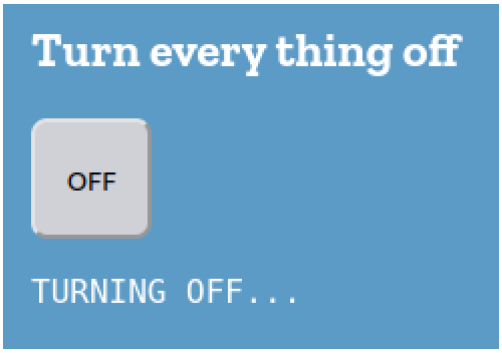

Figure 3.8: Things off Extension front-end

However, the devices to turn off are stored inside an array. A loop iterates over each entry of the array of web things — that was filtered to just contain devices having the `on` property — to turn them off. The problem is that the loop has a termination condition that is always true (see Listing 3.16). Hence, once reached the last web thing, the add-on will keep turning it off. Therefore, the user will lose control over the last web thing and will not be able anymore to turn and leave it on whenever it wants because it will be turned off again a few moments later resulting in T7 (see subsection 3.1.3).

Listing 3.17 shows the fixed `for` loop.

Listing 3.16 T7 - Turning off loop

```
1  /* ... */
2  off_button.addEventListener('click', () => {
3                  pre.innerText = "TURNING OFF...";
4                  window.API.getThings().then((res) => {
5                      const jsonarray = res  // Array of Things in JSON format
6                          .filter(obj => obj.properties.on !== undefined);
7
8                      let uri = "";
9                      const body = {on: false};
10                     for (let i = 0; jsonarray.length; i++) {  // Error: no termination condition
11                         let obj = jsonarray[i];
12                         if (obj !== undefined) {
```

```
                    uri = obj.id + "/properties/on";
                }
                window.API.putJson(uri, body).catch(/* ... */);
                sleep(1000);  // 1s delay
            }
        }).catch(/* ... */);
    });

/* ... */
```

Listing 3.17 T7 - Fixed loop

```
/* ... */
for (let i = 0; i < jsonarray.length; i++) {  // Fix: added the termination condition
    let obj = jsonarray[i];
    if (obj !== undefined) {
        uri = obj.id + "/properties/on";
    }
    window.API.putJson(uri, body).catch(/* ... */);
    sleep(1000);  // 1s delay
}
}).catch(/* ... */);
/* ... */
```

3.2.9 T8 - `smart-plug-adapter`

The `smart-plug-adapter` is a WebThings Gateway adapter add-on (see subsubsection 2.1.3.1). This adapter, every 15 minutes, stores the instantaneous power consumption data of a virtual smart plug in a file. Figure 3.9 shows a plug in the Smart Home Gateway dashboard.

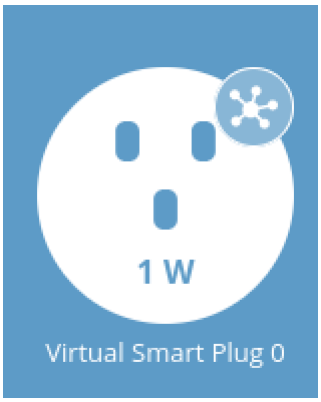

Figure 3.9: Generic virtual plug in the dashboard

This implementation of the adapter leads to T8 because in the `setInterval()` function, the argument that represents the delay is not 15 minutes but 15 seconds (see Listing 3.18). Therefore, assuming that the device which runs the SHG likely has a Solid State Drive (SSD) memory [46] or a flash memory card whose life is affected by the number of writings [47], this implementation error will likely accelerate by 60 times the breakdown of the device's memory and the memory fill time, other than consuming more resources and causing other related problems.

Listing 3.19 shows how to easily fix the adapter.

Listing 3.18 T8 - `instantaneousPower` memorization

```
/* ... */

/* Property of a device */
class VirtualThingsProperty extends Property {
    constructor(device, name, descr, value, interval) {
      /* ... */

     if (this.name === 'instantaneousPower') {
        this.interval = setInterval(() => {
          const date = new Date();
          const year = /*...*/, month = /*...*/, day = /*...*/;
          fs.appendFileSync(path.join(baseDir, `powerValues-${year}${month}${day}.txt`)
      ,
            `${(this.value).toFixed(3)}\n`);
        }, 15 * 1000);  // Error: 15 seconds
      }
    }
}
/* ... */
```

Listing 3.19 T8 - `setInterval()` fix

```
/* ... */

/* Property of a device */
class VirtualThingsProperty extends Property {
    constructor(device, name, descr, value, interval) {
      /* ... */

     if (this.name === 'instantaneousPower') {
        this.interval = setInterval(() => {
          const date = new Date();
          const year = /*...*/, month = /*...*/, day = /*...*/;
          fs.appendFileSync(
            path.join(baseDir, `powerValues-${year}${month}${day}.txt`),
            `${(this.value).toFixed(3)}\n`);
        }, 15 * 60 * 1000);  // Fix: 15 minutes
```

```
16      }
17    }
18  }
19  /* ... */
```

3.3 Limitations

3.3.1 T9, T10, T11

To briefly summarize which are the exploited threats, the related implementations and the security properties they break, this section groups them in Table 3.1.

Security Property	Threat	Exploited
Confidentiality	T1 (`weather-adapter`) T2 (`weather-adapter`)	✓ ✓
Integrity	T3 (`lights-off-extension`) T4 (`smart-plugs-adapter`)	✓ ✓
Availability	T5 (`things-off-extension`) T6 (`power-cons-extension` `plug-smart-adapter`) T7 (`things-off-extension`) T8 (`smart-plug-adapter`)	✓ ✓ ✓ ✓
Authentication	T9	-
Authorization	T10	-
Non-repudiation	T11	-

Table 3.1: Exploited threats

During the development of this thesis some attempts to implement Proofs of Concept (PoCs) exploiting T9, T10, and T11 were made.

T9 is about the authentication of an add-on. More precisely, an occurrence of this threat happens whenever an add-on successfully interacts with a system component pretending to be a different entity. T10, instead, is about authorization and it occurs when an add-on successfully accesses an authorization level higher than the one it should have. Hence, having these two threats in mind, I tried to exploit both at once. In WebThings the only category of add-on allowed to use the SHG's `APIs` are the extensions (see subsubsection 2.1.3.3). Therefore, an adapter add-on (see subsubsection 2.1.3.1) that successfully exploits these `APIs` would be an occurrence of both T9 and T10. However, the SHG's authorization mechanisms are based on `JSON` Web Tokens (**JWTs**), i.e., an open standard that defines a compact and self-contained way for securely transmitting information between parties as

a `JSON` object [48]. Each `APIs` call is authenticated through a JWT, and I did not find a way to break this security mechanism to achieve the occurrences of T9 and T10 following this path. I also wondered about the authentication system of WebThings, and I tried to understand whether only the users that installed an add-on can use it. In this regard, the SHG, by default, has a multi-user capability, and each user authenticates itself through a username. Then, once inside the SHG, each user has access to the whole SHG add-ons and settings. Hence, each user has the same privileges and is not restricted to their personal user space and so do its add-ons.

T11, instead, is about non-repudiation and an occurrence of this threat implies that an add-on anonymously communicates with an attack target in such a way that there is no way to tell with certainty who were the parties involved in the communication. In this regard, I checked whether the SHG has some mechanisms to keep track of actions performed by its components (e.g., logging). However, WebThings, by default, just logs on file what is printed out on the console and does not specify to whose add-on that print belongs to. Thus, it becomes difficult to associate the actions performed within the platform with the agent who committed them.

3.3.2 Devices

It is important to point out that the PoCs have been developed and tested using virtual devices (i.e., weather stations, plugs and bulbs). However, this does not invalidate the outcomes of the study because what matters is not the device itself but the properties of that device and the way of interacting with it. In this sense, within the SHG, the properties of a virtual device are modeled like the ones of a physical device. For example, the SHG will model the `on` property of a physical light bulb in the same way as the `on` property of a virtual light bulb.

3.4 Designing a new threat

Since the beginning of this work, the importance of the add-ons' scope has been clear. It is the set of resources an add-on is supposed to access legitimately (e.g., access tokens, configuration files, data folders, etc.). In this regard, the threat model used as a reference for this work is plenty of references to this scope. As the study and development of the threats' PoCs proceeded, the lack of a mechanism that circumscribed the writing and reading space, or the permissions of a running add-on emerged. What has been said can become a problem when, for example, those who develop add-ons end up making two domains overlap on purpose, or by mistake (for example, in subsubsection 3.2.2.1, subsubsection 3.2.2.2 and subsection 3.2.7). Hence, during the development of this work a new threat, called **T0**, was designed.

- **T0**: a plug-in scope could overlap the scope of another attack target.

This threat occurrence can lead to a very dangerous situation. More reasonably, the attack target can be another add-on. Indeed, if two add-ons share the same scope, it is possible to observe many of the threats presented in the threat model. For instance, an add-on could access, use, alter, and spread private data of the add-on which is sharing the scope with (T1, T2, T4). It could alter the state of other smart home devices (T3). It could delay the regular functionality of an attack target if it keeps using a resource within the scope needed also by other add-ons (T5). It could alter the regular functionality of an attack target (T6 or T7), and it could even damage the attack target by configuring wrong values. If the add-on has access to the access token of another one could even interact with other entities pretending to be a different actor (T9), with a higher level than expected (T10) and even in an anonymous way (T11).

Although it seemed promising, during the study it was decided not to take it to the PoC development and validation stages, limiting the work to only developing PoCs for the threats already highlighted in the threat model. Therefore, a validation of this new proposal is left for future works.

Chapter 4

Case Studies

To validate the Proofs of Concept (PoCs) two online surveys were conceived with different goals and aiming at different figures as respondents for skills and experiences.

4.1 Experts Survey

The evaluation of the PoCs was a crucial task. Hence, my supervisors and I organized a brainstorming meeting session to define the best way to proceed. The meeting was held in the presence of the members of the e-Lite research group[1] of the Politecnico di Torino. The group's main research fields are *Human-Computer Interaction*, *Ubiquitous Computing*, and *Artificial Intelligence*. During this meeting, we presented the idea behind the thesis project and some drafts of the PoCs implemented. Afterward, we discussed the necessity of validating the quality of the developed PoCs. The main goal was to demonstrate that those PoCs could be developed by inexperienced programmers. Finally, we decided to take advantage of a validation through a survey with a large sample of users. However, before giving the PoCs to these users, we felt the necessity of a preliminary validation that should have been done by a set of experts. This team was composed by five members that were involved in the brainstorming session. Among them, there was: one full professor, one associate professor, two assistant professors, and one Ph.D. student. Each of them belongs to the e-Lite research group of Politecnico di Torino. These experts had to evaluate the PoCs through an online survey before moving on to the user study.

The conducted survey contains a snippet for each Proof of Concept and the description of the expected behavior of the add-on running that code. The experts

[1] e-Lite Research Group: elite.polito.it

had to find out the problems in the snippets' code and evaluate whether each error in the Proofs of Concept could be considered a development error. The experts expressed their evaluation through a numeric five-level Likert Scale [49]. The higher the evaluation of the snippet, the higher the confidence that its problem is just a simple development error. In the end, we took into account also the cases in which no error in the snipped was detected.

4.1.1 Results

Table 4.1 shows the results of the experts' survey specifying the score the experts assigned to each add-on and how many experts, out of five, have found the threat inside the code by performing the survey. Furthermore, the fact that some of the experts did not find any threat in some add-ons was considered a positive thing — for what concerns the PoCs — because it enforced the belief that some errors are not so easy to spot even to experienced programmers.

Add-on (threat)	**Times that was spotted**	**Average score**
`weather-adapter` (T1)	4	3
`weather-adapter` (T2)	5	3.8
`lights-off-extension` (T3)	4	3.5
`smart-plugs-adapter` (T4)	4	4.75
`things-off-extension` (T5)	5	4.4
`power-cons-extension` (T6-v1)	3	3.3
`plug-smart-adapter` (T6-v2)	5	3.4
`things-off-extension` (T7)	5	3.8
`smart-plug-adapter` (T8)	3	4.33

Table 4.1: Experts' survey result

During the brainstorming session, it was decided that only the snippets that would have obtained a score greater than 3 would go to the next phase. Since the results were above that minimum threshold, all of the developed add-ons — with the exception of `power-cons-extension` (see section 4.2) — were selected for the next stage, i.e., the user study.

4.1.1.1 Comments

The experts gave also some interesting comments about some of the snippets. Among the things that were pointed out, I'll report the most remarkable ones.

- **`power-cons-extension`** - Assistant professor: *"the error that results in the*

threat occurrence is a conceptual error. While the other PoCs contain programming errors";

- `smart-plugs-adapter` - Associate professor: *"this PoC has the error that is the hardest one to spot"*;

- `things-off-extension` **(T7)** - Full professor, associate professor, and assistant professor: *"this PoC contains a very naïve programming error is not so easy to spot by reading the code but some Integrated Development Environments (IDEs) may point it out"*.

4.1.1.2 Other Proofs of Concept

The add-ons that were presented in this thesis and submitted to the experts do not represent the total amount of developed PoCs. In fact, during the development phase of this thesis, other PoCs were conceived, tested, and discussed. However, these PoCs were rejected because, during the preliminary evaluation phase with my supervisors, they seemed deliberately malicious rather than seem developed by a novice or a careless developer. This is also why they were not presented and described in this thesis, other than for brevity reasons.

4.2 User Surveys

The total amount of developed PoCs is nine. Since, the objective was to asses a PoC for each threat and, among the PoCs, two regard T6 (i.e., `plug-smart-adapter` and `power-cons-extension`), for the sake of a balanced threats distribution of two snippets implementing threats per survey, and a total amount of four surveys, the only add-on between them that took part to the user study was the one that received a higher evaluation from the experts' feedback, i.e., `plug-smart-adapter`. Furthermore, to avoid answers being influenced by the snippets' order, the order of appearance of each snippet was randomized. Moreover, to lower the abandonment rate, the survey should not last for too long and not contain too many snippets. Hence, for each survey, a participant has to assess four snippets and understand whether a snippet leads the Smart Home to an undesired behavior and explain where the problem is and why. To not bias the participants, each survey contains half of the snippets (i.e., two out of four) that implement a threat while the other half of the snippets (i.e., two out of four) are fixed ones taken among our set of PoCs.

Each survey also contains a final part that asks the participants questions about their background (e.g., study title, years of experience in programming, etc.) and knowledge about `JavaScript` and SHGs.

To get a high participation rate and a user base as heterogeneous as possible, we shared the surveys:

- On WebThings' official forum[2];
- Among a sample of Politecnico di Torino students that we were sure had the background to answer the questions because they attended a course of web development strongly focused on `JavaScript` (`JS`) and `JS` frameworks;
- On a subreddit about surveys[3] by specifying who our participant targets were;
- On SurveyCircle[4] (i.e., a large community for online research) & its related groups;
- On a Telegram[5] group of Computer Engineering students at Politecnico di Torino.

Table 4.2 shows the content of each survey and indicates with **Broken** a snippet coming from an add-on containing a threat occurrence; with **Fixed**, instead, a snippet coming from an add-on where the threat occurrence was removed:

Add-on (threat)	**Survey 1**	**Survey 2**	**Survey 3**	**Survey 4**
`weather-adapter` (T1)	Broken	-	Fixed	Fixed
`weather-adapter` (T2)	-	Broken	-	-
`lights-off-extension` (T3)	Broken	Fixed	-	-
`smart-plugs-adapter` (T4)	Fixed	-	Fixed	Broken
`things-off-extension` (T5)	-	Fixed	Broken	-
`plug-smart-adapter` (T6)	-	-	Broken	Fixed
`things-off-extension` (T7)	-	-	-	Broken
`smart-plug-adapter` (T8)	Fixed	Broken	-	-

Table 4.2: Content of the user surveys

4.2.1 Results

While I am writing this thesis the user study is still ongoing to collect a higher number of answers. Therefore, the following are preliminary results.

[2]Mozilla Discourse - IoT - WebThings: discourse.mozilla.org/c/iot/252

[3]Reddit - SampleSize: reddit.com/r/SampleSize

[4]SurveyCircle: surveycircle.com

[5]Telegram Messenger: telegram.org

Each survey is composed of four snippets of code. Two of them contain a threat occurrence, while the other two do not contain any unexpected behavior. The latter were developed to fix the PoCs previously presented. The surveys were started 182 times in *total*, there were 157 *partial* compilations, and 25 *full* compilations considering also two participants who did not give their consent to participate to the study but still submitted their surveys. Table 4.3 summarize these information.

	Compilations
Total	182
Partial	157
Full	25
No consent	2

Table 4.3: Survey compilations

The results of the snippets containing threats will be now discussed.

- `weather-adapter` **(T1)**: 1 participant out of 4 thought they have found the threat. However, the code lines and the undesired behavior described in its answer lead to the conclusion that the participant has not found the threat occurrence;
- `weather-adapter` **(T2)**: none of the 5 participants reported to have found the threat;
- `lights-off-extension` **(T3)**: 2 participants out of 4 thought they have found the threat. But, in this case, just one of them has really found it and considers it something accidental;
- `smart-plugs-adapter` **(T4)**: 2 participants out of 5 stated to have found the threat. However, just one of them have really found it. Even in this case, it considers the threat something accidental;
- `things-off-extension` **(T5)**: 3 participants out of 9 said to have found the threat, but their following answers lead to the conclusion that they have not found the threat occurrence;
- `plug-smart-adapter` **(T6)**: as before, 2 participants out of 9 said to have found the threat, but they have not;
- `things-off-extension` **(T7)**: again, 2 participants out of 5 thought to have found the threat, but they have not;
- `smart-plug-adapter` **(T8)**: none of the 5 participants reported to have found the threat.

Then, it is also worth to mention that some snippets with no threat occurrences were flagged as dangerous even though they were not. To be precise, 9 out of 46 total answers about snippets without threat reported false positives. These snippets served to not bias the respondents and do not let them thinks that each PoC contains a threat occurrence.

Table 4.4 shows, for each threat occurrence, how many users assessed the PoC, how many of them claimed to have found the threat, how many actually did, and — among who found it — how many labeled it as intentional.

PoC's threat	Users	Claimed found	Actually found	Intentional
T1	4	1	0	-
T2	5	0	-	-
T3	4	2	1	0
T4	5	2	1	0
T5	9	3	0	-
T6	9	2	0	-
T7	5	2	0	-
T8	5	0	-	-

Table 4.4: User study results

To summarize these results, a very small amount of participants (i.e., 2 out of 23), spotted the threats and none of them categorized the behaviour as deliberate. Moreover, this happened for just two PoCs out of eight while, in the other ones, no one found the threat occurrence. Furthermore, since we chose to consider as validated each PoC recognized as containing a threat occurrence and then marked as not intentional by at least the 75% of users, the results are excellent.

It is also interesting to report the data extracted from the questions about the background of the survey respondents. Thence, among the 23 participants involved in the study, just one regularly use the WebThings SHG and another one used it in the past but not anymore. However, that number triples if we take into consideration the users that have heard in depth about WebThings, e.g., because they have read its documentation or saw a project that involved the SHG. Instead, 16 participants never heard about WebThings. The number of users that have heard about other SHGs is ten. The most known among the other SHGs is Home Assistant [13] because, among the previous ten participants, the number of participants that have heard about it is five, instead, just one have ever heard of openHAB [15]. Four are the participants that regularly use other SHGs, e.g., Home Assistant, Google Nest [10], Amazon echo [11] etc. Another relevant data is about the number of participants that have ever developed an add-on for a SHG,

which is one. The SHG in question is Home Assistant. Table 4.5 summarizes these statistics about SHGs' popularity.

WebThings	**Participants**
Regularly use it	1
Used it in the past	1
Just heard of it	3
Use another SHG (e.g., Amazon echo, Home Assistant, etc.)	4
Never heard about it	16
Have heard about other SHGs	10
Home Assistant	**Participants**
Have heard about Home Assistant	5
Have developed an add-on for Home Assistant	1
openHAB	**Participants**
Have heard about openHAB	1

Table 4.5: SHG background statistics

Most of the participants, 18 out of 23, were Italian. Other smaller percentages of participants were Indian (two), Canadian (one), Polish (one), and Iranian (one). Accordingly, Table 4.6 shows shows a recap of the participants' nationalities. Among them, there were a few beginners with less than one year of programming background, but the vast majority of them were multi-years programmers and the average years of programming experience of the users population was 4.65 years. Therefore, their average level of confidence from 1 to 5 with `JavaScript` was 2.9.

Nationality	**Participants**
Italians	18
Indians	2
Polish	1
Canadian	1
Iranian	1

Table 4.6: Participants' nationality

Lastly, among the channels through which the surveys were shared, the most effective one was the e-mail sent to the Web Application students of the Politecnico di Torino because it attracted 12 participants to the study. Sharing the surveys in the Telegram[6] group of Computer Engineering students at Politecnico di Torino

[6]Telegram Messenger: telegram.org

brought four participants. Then, three were participants from the SampleSize[7] subreddit, and two participants joined the survey after receiving it through direct message. Lastly, two participants were from SurveyCircle[8] related groups. Table 4.7 shows these statistics.

Source	**Participants**
Email	12
Telegram	4
Reddit	3
Direct Message	2
SurveyCircle	2

Table 4.7: Surveys sharing platforms

[7]Reddit - SampleSize: reddit.com/r/SampleSize

[8]SurveyCircle: surveycircle.com

Chapter 5

Conclusions

The basis of this thesis was a recently proposed threat model [41]. My focus was put on developing eight Proofs of Concept (PoCs) for implementing the first eight threat occurrences listed within the threat model.

To demonstrate that those PoCs can also be the outcome of an inexperienced or careless programmer, they were validated through two different surveys. The first one involved a team of experts with the goal to understand whether the conceived PoCs were plausible as the outcome of a novice programmer (see section 4.1). The second one involved a larger population of users to further validate these PoCs (see section 4.2).

This thesis shows the preliminary results of the user study. However, even though the user study is not yet concluded, the results are encouraging as far as concerns the aim of this thesis.

The expert survey results (subsection 4.1.1) showed off that the ideas behind the Proofs of Concept (PoCs) presented in chapter 3 were solid. This solidity was then confirmed by the current outcome of the user study presented in subsection 4.2.1. In fact, having conducted the study on a sample of users well representative of a population of *novice* developers permits to express some conclusions and state that the guidelines expressed in the reference threat model [41] can be applied to a modern extensible Smart Home Gateway (SHG) like WebThings [7]. Furthermore, it applies not only in those situations where there is an attacker with malicious intent, but also when an inexperienced programmer may commit some errors. In fact, according to the collected preliminary results, the written add-ons (which implement the threats described in section 3.1) could be the outcome of a distracted or novice programmer. Hence, having the threat model as a reference while developing add-ons or a feature for a SHG should be encouraged as a good practice.

However, the user study and the related surveys are still open at the time of publication of this thesis, and they are getting more and more participants. Their outcome will be resumed in the future since a more in-depth study will be published

as a follow-up to this thesis. Furthermore, it will possibly include and compare data obtained from other SHGs.

5.1 Future Works

In section 3.4 was introduced the threat T0, i.e., *a plug-in scope could overlap the scope of another attack target*. Since it was not further elaborated on in this thesis, it would be interesting if it were explored and developed in future studies and included in the threat model guidelines.

The user study evidenced the difficulties in which a work like this thesis may incur to get valid survey participants. In this case, the more remarkable issue was the high survey abandonment rate highlighted in Table 4.3. I would recommend for future works, to avoid issues like this, to proceed by possibly submitting the surveys in person. Another possibility to encourage the participation may be to give some reward to those participants who fully complete the survey (e.g., gift cards, coupons, gadgets, etc.). This can be done by drawing a few winners or by giving the reward to each of them.

Moreover, this work has focused on a single smart home gateway (WebThings), and the development of add-ons for it. It would be interesting to see threat implementations for other SHGs as well, e.g., Home Assistant [13], and openHAB [15]. In particular, to understand whether the PoCs that have not been developed during this work (for the reasons indicated in subsection 3.3.1) can be developed for platforms having different characteristics than WebThings. A further study is currently underway to investigate Home Assistant.

Bibliography

[1] Haseeb Touqeer, Shakir Zaman, Rashid Amin, Mudassar Hussain, Fadi Al-Turjman, and Muhammad Bilal. «Smart home security: challenges, issues and solutions at different IoT layers». In: *The Journal of Supercomputing* 77.12 (2021), pp. 14053–14089. DOI: 10.1007/s11227-021-03825-1 (cit. on pp. 1, 2).

[2] Zaied Shouran, Ahmad Ashari, and Tri Priyambodo. «Internet of things (IoT) of smart home: privacy and security». In: *International Journal of Computer Applications* 182.39 (2019), pp. 3–8 (cit. on p. 1).

[3] Sudeendra Kumar K, Sauvagya Sahoo, Abhishek Mahapatra, Ayas Kanta Swain, and K.K. Mahapatra. «Security Enhancements to System on Chip Devices for IoT Perception Layer». In: *2017 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*. 2017, pp. 151–156. DOI: 10.1109/iNIS.2017.39 (cit. on p. 1).

[4] A Vimal Jerald. «Internet of things (IoT) based smart environment integrating various business applications». In: *International Journal of Computer Applications* 128.8 (), pp. 32–37 (cit. on p. 1).

[5] Anastasia Yastrebova, Ruslan Kirichek, Yevgeni Koucheryavy, Aleksey Borodin, and Andrey Koucheryavy. «Future Networks 2030: Architecture & Requirements». In: *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. 2018, pp. 1–8. DOI: 10.1109/ICUMT.2018.8631208 (cit. on p. 1).

[6] Abhay Kumar Ray and Ashish Bagwari. «IoT based Smart home: Security Aspects and security architecture». In: *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*. 2020, pp. 218–222. DOI: 10.1109/CSNT48778.2020.9115737 (cit. on p. 2).

[7] Krellian Ltd. *About WebThings*. [Online; accessed 30-March-2023]. URL: https://webthings.io/about/ (cit. on pp. 2–7, 45).

[8] Benjamin T Francis. «Mozilla WebThings: An open source implementation of the Web of Things». In: (2019). URL: https://w3c.github.io/wot/workshop/ws2/Papers/29%20-%20Francis,%20Mozilla%20-%20Mozilla%20Web%20Things.pdf (cit. on p. 3).

[9] Apple Inc. *Apple HomePod Support*. [Online; accessed 30-March-2023]. 2023. URL: https://support.apple.com/homepod (cit. on p. 4).

[10] Google LLC. *Google Nest*. [Online; accessed 30-March-2023]. 2023. URL: https://store.google.com/us/category/connected_home?hl=en-US&GoogleNest&utm_source=nest_redirect&utm_medium=google_oo&utm_campaign=homepage&pli=1 (cit. on pp. 4, 42).

[11] Amazon.com. *Amazon echos*. [Online; accessed 13-July-2023]. 2023. URL: https://www.amazon.com/smart-home-devices/b?node=9818047011 (cit. on pp. 4, 42).

[12] SmartThings Inc. *SmartThings*. [Online; accessed 30-March-2023]. 2023. URL: https://www.smartthings.com (cit. on p. 4).

[13] Nabu Case Inc. *Home Assistant*. [Online; accessed 30-March-2023]. 2023. URL: https://www.home-assistant.io (cit. on pp. 4, 42, 46).

[14] Krellian Ltd. *WebThings*. [Online; accessed 30-March-2023]. 2023. URL: https://webthings.io/ (cit. on p. 4).

[15] openHAB Community and the openHAB Foundation e.V. *openHAB*. [Online; accessed 30-March-2023]. 2023. URL: https://www.openhab.org (cit. on pp. 4, 42, 46).

[16] Krellian Ltd. *WebThings Gateway*. [Online; accessed 30-March-2023]. URL: https://webthings.io/gateway/ (cit. on pp. 5, 6).

[17] Krellian Ltd. *WebThings Framework*. [Online; accessed 30-March-2023]. URL: https://webthings.io/framework/ (cit. on p. 7).

[18] WebThingsIO contributors. *Gateway Architecture*. [Online; accessed 30-March-2023]. URL: https://github.com/WebThingsIO/wiki/wiki/Gateway-Architecture (cit. on pp. 7, 8).

[19] OpenJS Foundation and Node.js contributors. *Node.js*. [Online; accessed 30-March-2023]. 2023. URL: https://nodejs.org/en (cit. on p. 7).

[20] OpenJS Foundation. *Express*. [Online; accessed 30-March-2023]. 2023. URL: https://expressjs.com (cit. on p. 7).

[21] visionmedia/page.js contributors. *page.js*. [Online; accessed 30-March-2023]. 2023. URL: https://github.com/visionmedia/page.js (cit. on pp. 7, 8).

[22] SQLite Consortium. *SQLite Home Page*. [Online; accessed 30-March-2023]. 2023. URL: https://www.sqlite.org/index.html (cit. on p. 8).

[23] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. «Design patterns: Abstraction and reuse of object-oriented design». In: *ECOOP'93—Object‡ Oriented Programming: 7th European Conference Kaiserslautern, Germany, July 26–30, 1993 Proceedings 7*. Springer. 1993, pp. 406–431 (cit. on p. 8).

[24] WebThingsIO contributors. *HOWTO: Create an add-on*. [Online; accessed 30-March-2023]. URL: https://github.com/WebThingsIO/wiki/wiki/HOWTO%3A-Create-an-add-on (cit. on p. 9).

[25] Connectivity Standards Alliance. *Zigbee | Complete IOT Solution - CSA-IOT*. [Online; accessed 24-April-2023]. URL: https://csa-iot.org/all-solutions/zigbee/ (cit. on p. 9).

[26] Joshua Wright. «Killerbee: practical zigbee exploitation framework». In: *11th ToorCon conference, San Diego*. Vol. 67. 2009 (cit. on p. 9).

[27] Nicolas Mayer, Patrick Heymans, and Raimundas Matulevicius. «Design of a Modelling Language for Information System Security Risk Management.» In: *RCIS*. 2007, pp. 121–132 (cit. on p. 10).

[28] Suvda Myagmar, Adam J Lee, and William Yurcik. «Threat modeling as a basis for security requirements». In: (2005). URL: http://d-scholarship.pitt.edu/16516/ (cit. on p. 10).

[29] Zoe Braiterman et al. *Threat Modeling Manifesto*. [Online; accessed 30-March-2023]. 2023. URL: https://www.threatmodelingmanifesto.org (cit. on p. 10).

[30] Adam Shostack. «Experiences Threat Modeling at Microsoft.» In: *MODSEC@ MoDELS* 2008 (2008), p. 35 (cit. on p. 10).

[31] Nan Messe, Vanea Chiprianov, Nicolas Belloir, Jamal El-Hachem, Régis Fleurquin, and Salah Sadou. «Asset-Oriented Threat Modeling». In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2020, pp. 491–501. DOI: 10.1109/TrustCom50675.2020.00073 (cit. on pp. 11, 12).

[32] K. Tuma, G. Calikli, and R. Scandariato. «Threat analysis of software systems: A systematic literature review». In: *Journal of Systems and Software* 144 (2018), pp. 275–294. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2018.06.073. URL: https://www.sciencedirect.com/science/article/pii/S0164121218301304 (cit. on p. 11).

[33] Nataliya Shevchenko, Timothy A Chick, Paige O'Riordan, Thomas P Scanlon, and Carol Woody. *Threat modeling: a summary of available methods*. Tech. rep. Carnegie Mellon University, Software Engineering Institute, Pittsburgh, United States, 2018 (cit. on pp. 11, 12).

[34] Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack. «Threat modeling-uncover security design flaws using the stride approach». In: *MSDN Magazine-Louisville* (2006), pp. 68–75 (cit. on p. 11).

[35] Katja Tuma and Riccardo Scandariato. «Two Architectural Threat Analysis Techniques Compared». In: *Software Architecture*. Ed. by Carlos E. Cuesta, David Garlan, and Jennifer Pérez. Cham: Springer International Publishing, 2018, pp. 347–363. ISBN: 978-3-030-00761-4 (cit. on p. 11).

[36] Daniel Magin, Rahamatullah Khondoker, and Kpatcha Bayarou. «Security analysis of OpenRadio and SoftRAN with STRIDE framework». In: *The 24th international conference on computer communications and applications (ICCCN 2015). IEEE, Las Vegas, Nevada, USA (3–6 Aug 2015)*. Vol. 38. 2015 (cit. on p. 11).

[37] Rafiullah Khan, Kieran McLaughlin, David Laverty, and Sakir Sezer. «STRIDE-based threat modeling for cyber-physical systems». In: *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. 2017, pp. 1–6. DOI: 10.1109/ISGTEurope.2017.8260283 (cit. on p. 12).

[38] Tony UcedaVelez and Marco M. Morana. «Intro to Pasta». In: *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. 2015, pp. 317–342. DOI: 10.1002/9781118988374.ch6 (cit. on p. 12).

[39] Kyoung Ho Kim, Kyounggon Kim, and Huy Kang Kim. «STRIDE-based threat modeling and DREAD evaluation for the distributed control system in the oil refinery». In: *ETRI Journal* 44.6 (2022), pp. 991–1003. DOI: https://doi.org/10.4218/etrij.2021-0181. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.4218/etrij.2021-0181. URL: https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.2021-0181 (cit. on p. 12).

[40] Ljubomir Lazic and Nikos Mastorakis. «Cost effective software test metrics». In: *WSEAS Transactions on Computers* 7.6 (2008), pp. 599–619 (cit. on p. 12).

[41] Fulvio Corno and Luca Mannella. «A Threat Model for Extensible Smart Home Gateways». In: *2022 7th International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE. 2022, pp. 1–6. DOI: 10.23919/SpliTech55088.2022.9854235 (cit. on pp. 13, 45).

[42] Robert W. Shirey. *Internet Security Glossary, Version 2*. RFC 4949. Aug. 2007. DOI: 10.17487/RFC4949. URL: https://www.rfc-editor.org/info/rfc4949 (cit. on p. 15).

[43] NIST Computer Security Division IT Laboratory. *Minimum Security Requirements for Federal Information and Information Systems*. Tech. rep. National Institute of Standards and Technology, 2019. DOI: 10.6028/NIST.FIPS.200 (cit. on p. 15).

[44] Elaine Barker and William Barker. *Recommendation for key management, part 2: best practices for key management organization*. Tech. rep. National Institute of Standards and Technology, 2019. DOI: 10.6028/NIST.SP.800-57pt2r1 (cit. on p. 16).

[45] W3C. *Web Application Manifest*. [Online; accessed 27-April-2023]. URL: https://w3c.github.io/manifest/#json-schema (cit. on p. 20).

[46] Rino Micheloni, Alessia Marelli, and Kam Eshghi. «Inside solid state drives (SSDs)». In: (2013) (cit. on p. 33).

[47] Qiang Li, Hui Li, and Kai Zhang. «A Survey of SSD Lifecycle Prediction». In: *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. 2019, pp. 195–198. DOI: 10.1109/ICSESS47205.2019.9040759 (cit. on p. 33).

[48] Michael B. Jones, John Bradley, and Nat Sakimura. *JSON Web Token (JWT)*. RFC 7519. May 2015. DOI: 10.17487/RFC7519. URL: https://www.rfc-editor.org/info/rfc7519 (cit. on p. 35).

[49] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. «Likert scale: Explored and explained». In: *British journal of applied science & technology* 7.4 (2015), p. 396 (cit. on p. 38).