Validating a Threat Model for Smart Home Gateways

Candidate: Antonino Santa Rosa
Supervisors: Prof. Fulvio Corno, Dott. Luca Mannella

Introduction

Smart Home systems give users the possibility to connect and control, within a residential environment, connected devices like bulbs, sensors, and more. Although Smart Homes have increased users' convenience, they can present vulnerabilities.

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

Using as a reference an already existing threat model, the objective of this thesis is to understand whether add-ons developed by novice or careless programmers can cause threat occurrences in the Smart Home. The extensible SHG that was the target of evaluation is WebThings, an open-source platform incubated at Mozilla for four years before being spun out as an independent open-source project.

Reference Threat Model & Proofs of Concept

The reference Threat Model (TM) applies to extensible SHGs and only considers menaces originating from add-ons. This TM identifies *11 threats* that may target the main components of the SHGs. 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*. Moreover, the TM also refers to the **add-on's scope** like the set of resources (e.g., access tokens, configuration files, data, etc.) an add-on is supposed to access legitimately.

The TM gives hints about what a threat occurrence could be. In this regard, I focused on the first eight threats of the TM and used the TM itself as a reference to develop eight **Proofs of Concept** (PoCs) containing these threats occurrences. Each PoC was developed trying to reproduce programming errors that a careless or novice programmer could have done, leading to the threat occurrence.

Threat number one (**T1**), i.e., *an add-on accesses and uses private data outside its scope belonging to an attack target*, and **T2**, i.e., *an add-on spreads data outside its scope belonging to the attack target*, breach *confidentiality.* These threats were implemented using the same add-on. In particular, this add-on provides the SHG with a virtual weather station that retrieves weather data from a provider. The user can choose to use a *default* Application Programming Interface (API) Key (basically a *free* one) or a *personal* one (likely a *premium* one) to query the provider. The latter is also backed up on Dropbox before being overwritten. The add-on, due to a typo, uses a data path to retrieve the API keys that is slightly different from the one it should use and that belongs to another add-on. Hence, this add-on will use the API key of another add-on, resulting in T1. Moreover, it will spread the API key of the other add-on if it backs it up on Dropbox, resulting in T2.

T3, i.e., *an add-on alters the state of Smart Home devices that are not supposed to be in its scope*, breaches *integrity.* It was implemented through an add-on meant to turn off every light within the Smart Home environment. However, the code misses a filtering function. The function would filter from the list of devices to turn off only the lights. Thus, this implementation of the add-on turns off every device having an `on` property and not only lights, resulting in T3.

T4, i.e., *an add-on alters private data of an attack target*, is about *integrity.* It was implemented by an add-on that integrates up to two virtual smart plugs in the SHG. It also 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. However, it turns out that in the code it is not assigned an unique identifier to each file corresponding to each plug. Thence, the slowest plug to write that file will overwrite the file of the other one, resulting in T4.

T5, i.e., *an add-on delays the regular functionality of another Smart Home component*, breaches *availability.* It was implemented through an add-on meant to turn off every device within the Smart Home. Therefore, the add-on iterates on every Smart Home component, if it is a device, it will be turned off. However, even when the selected component is not a device, the last selected device receives a request to be turned off. This behaviour is iterated until a new device is reached. Consequently, trying to turn on the device while receives requests to keep it off will not be temporarily effective, resulting in T5.

T6, i.e., *an add-on alters one of the regular functionalities of another Smart Home component*, is about *availability.* It was implemented through an add-on that integrates a virtual smart plug in the SHG and regularly stores its instantaneous power consumption data in a file. Then, the adapter reads all these data to calculate and show the average power consumption in the SHG's dashboard. However, the data path written in the code slightly differs from the correct one and corresponds to another add-on's path. Accordingly, the average power consumption output in the dashboard will be wrong for both the add-ons, resulting in T6.

T7, i.e., *an add-on alters the regular functionality of another Smart Home component, preventing Smart Home users from using it.* It breaches *availability.* The related add-on was designed to turn off every Smart Home's device. The threat occurrence is due to an infinite loop in the code that keeps turning off the last device to turn off. Thus, the users will not be able anymore to turn and leave on the device because it will be turned off again a few moments later, resulting in T7.

T8, i.e., *an add-on physically damages an attack target in the Smart Home*, is about *availability.* It is implemented through an add-on that every 15 minutes, stores in a file the instantaneous power consumption data of a virtual smart plug. This implementation of the add-on leads to T8 because the argument of the function that represents the time delay is not 15 minutes but 15 seconds. Therefore, this can accelerate by 60 times the breakdown of the device's flash memory and the memory fill time, other than consuming more resources and causing related problems.

PoCs' Validations & Results

A group of experts belonging to the e-Lite research group was involved to validate the PoCs through a survey. The goal of this survey was to understand whether experienced programmers can consider the errors in the PoCs the outcome of an inexperienced or careless programmer and not malicious by design. In this evaluation, each PoC got a score higher than the minimum threshold. Hence, the PoCs were submitted, through another survey, to a larger population of users for an assessment. Each of the 23 respondents assessed two PoCs. Among these users, just a small amount spotted the threats and then categorized them as something not done intentionally. **Table 1** 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 1: User study results

The expert survey results showed off that the ideas behind the PoCs presented were solid. This solidity was confirmed by the outcome of the user study. In fact, it permitted to state that the guidelines expressed in the reference threat model can be applied to modern extensible SHGs like WebThings. Also, the results demonstrate that the discussed threats are indeed plausible in code developed by inexperienced developers. Hence, having the threat model in mind while developing add-ons for a SHG should be encouraged as a good practice.