MeshVPR: Citywide Visual Place Recognition Using 3D Meshes

A Threat Model for Extensible Smart Home Gateways

Fulvio Corno
Dipartimento di Automatica e Informatica
Politecnico di Torino
Turin, Italy
fulvio.corno@polito.it

Luca Mannella
Dipartimento di Automatica e Informatica
Politecnico di Torino
Turin, Italy
luca.mannella@polito.it

***Abstract*—This paper proposes a threat model for a specific class of components of IoT infrastructures: smart home gateways extensible through plug-ins. The purpose of the proposed model is twofold. From one side, it helps to understand some possible issues that could be generated from a malicious or defective implementation of a plug-in and affect the gateway itself or other smart home devices. Consequently, the model could help programmers of gateway applications, plug-ins, and devices think about possible countermeasures and develop more resilient solutions. On the other side, the model could be regarded as a set of guidelines. Indeed, plug-in developers should not create plug-ins acting like the threats reported in the paper. To provide a first validation of the model, the paper presents a use case based on Home Assistant, an open-source smart home gateway application.**

***Index Terms*—Cybersecurity, Gateways, Home Assistant, Internet of Things, Threat Modeling, Smart Home**

I. Introduction

There are several definitions for a smart home [1]. One of the simplest ones, proposed by the Oxford Dictionary, is: "a home equipped with lighting, heating, and electronic devices that can be controlled remotely by smartphone or computer". According to recent literature, smart home systems are nowadays considerably widespread and exhibit various security issues. Indeed, a study published in 2019 [2] states that around 40% of private dwellings worldwide have at least an Internet of Things (IoT) device. Moreover, this number grows to approximately 70% considering only North America's houses. Analyzing data coming from these devices, the researchers discovered that in 62% of the involved homes, there was at least one device affected by one known vulnerability. Furthermore, as demonstrated in another study [3], it is important to consider that the insecurity of a device connected to a network could have severe consequences for all the others.

To manage all the connected devices, smart home gateways are often involved. These specific machines work as the central control unit of a smart home system. They communicate with (and coordinate) the smart devices connected to the local network, establish communications to the internet and, sometimes, implement additional security mechanisms [3], [4].

In line with the broad possibilities offered by smart homes and the related IoT devices, there are many open-source smart home gateways designed to be extensible (e.g., EventGhost, Home Assistant, OpenHAB, Pimatic, etc.). Indeed, extending software released by a third party is not a new concept in software development. Indeed, many other products and platforms offer developers the possibility of integrating new functionalities [5]. To provide other examples, browsers and Integrated Development Environments (IDE) are often extensible. These software expansions assumed various names like add-ons, extensions, integrations, or plug-ins. This paper uses "plug-in" to reference a generic extension of the gateway's core functionalities.

Considering that smart home gateways have to interact with almost all the devices available in the house, they are particularly critical. Moreover, plug-in development is another critical and less controlled process because it is often conducted by independent developers not belonging to the "core team". Therefore, this paper focuses on proposing a threat model that could be applied during the development (or the analysis) of plug-ins for extensible smart home gateways.

In line with a systematic literature review recently conducted on the topic [6], there are several definitions of threat modeling. These definitions are used in many different and sometimes also incompatible ways. However, to provide a simple definition of this concept, the paper considers the description provided by the Threat Modeling Manifesto [7]: "threat modeling is analyzing representations of a system to highlight concerns about security and privacy characteristics".

The aim of the threat model presented in this paper is twofold. From one side, it helps developers understand some possible attacks that could affect the main components of a smart home. Indeed, having these menaces in mind could help them think about possible countermeasures and develop more resilient solutions. On the other side, this threat model could be considered as a set of guidelines for plug-in developers. Indeed, programmers should not develop plug-ins that act like the described threats.

The rest of the paper is organized as follows: Section II analyzes the related work, and Section III presents a smart home reference architecture. Then, the article defines the proposed threat model in Section IV. Section V describes an application of the threat model to a possible use case. Section VI summarizes the outcomes of this work, and Section VII concludes the paper and proposes insights on future activities.

II. Related Work

Numerous documented attacks in the literature show that starting from a device is possible to compromise many others. For instance, researchers demonstrated a lateral privilege escalation attack in a smart home equipped with Google's Nest and Philips Hue [3]. However, adopting a smart home gateway could increase the security of the smart home. For example, this kind of gateway could be equipped with an Intrusion Detection System (IDS) to discover possible attacks from the beginning [4]. Moreover, Yang et Al. proposed another interesting solution to improve gateway security [8]. In their work, they developed a smart home gateway able to use onion routing to transmit IoT data streams efficiently.

Nevertheless, considering that the gateway interacts with almost all smart home devices, it could create severe issues if compromised. Hence, it is essential to design the gateway and its plug-ins in a secure way to reduce the possibility of exposing the house to attacks. Indeed, programmers must develop plug-ins having two concepts in mind. From one side, developed plug-ins could expose the gateway and the elements inside the house to new threats. On the other side, their plug-ins could be targeted by attacks and unexpected events.

Following a security-by-design approach, one of the best tools to secure systems from the very beginning is threat modeling. The more significant resources used to design the proposed model were the previously cited Threat Model Manifesto [7] and the documentation provided by the Open Web Application Security Project (OWASP) [9], with a particular focus on the OWASP IoT Project [10]. Indeed, these documents offer constructive insights to design an adequate threat model.

Among the more significant related works, the paper considers the STRIDE model [11] proposed by Microsoft. The model's name is an acronym that helps remember the class of threats described by the model itself. They are Spoofing, Tampering, Repudiation, Information disclosure, Denial of service (DoS), and Elevation of privilege. This model was one of the first to be highly adopted by practitioners, and it was also involved in many IoT research activities. For instance, it was used to secure Smart Cities [12], Smart Grids [13], and Cyber-Physical Systems [14]. However, for the aim of this paper, this model is too generic, which leads us to more specific resources.

Considering not only the smart homes but the entire IoT domain, one of the more complete threat models designed for IoT systems that we were able to identify was proposed by Pal et Al. [15]. This model divides IoT threats into five macro-categories (i.e., communications, device/services, users, mobility, and integration of resources) and discusses possible attacks from different perspectives. Another relevant resource for smart home security is the contribution of Heartfield et Al., [16] which helps understand security threats related to this domain and their possible impacts. However, these contributions have a broad view of the IoT domain, while our focus is specifically on extensible smart home gateways.

III. Reference Architecture

This section describes a generic smart home architecture with an extensible gateway as its central focus. A smart home gateway usually has a software application designed to manage the smart home and possibly other applications. All these programs are managed by an operating system. As already specified, this paper considers smart home gateways that could be expanded through plug-ins. Indeed, this kind of gateway application could be viewed as a container for several plug-ins that run in parallel. In addition, the reader should not forget that the gateway interacts with other IoT devices inside the house. Furthermore, the gateway is also typically connected to the internet and, in certain situations, could be the only device able to communicate outside the smart home local network.

Figure 1 graphically represents the reference architecture explained so far. We designed and considered this architecture to have a specific reference during the creation of the proposed threat model.

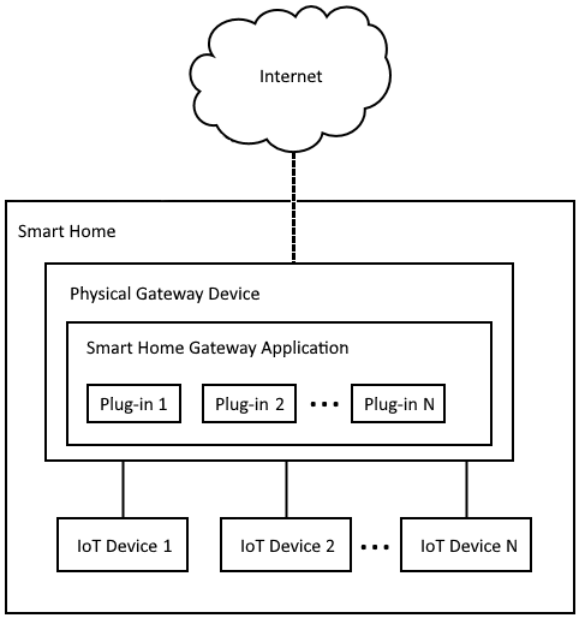

Fig. 1. Smart Home architecture.

IV. Proposed Threat Model

To create the proposed threat model, we started from the leading security properties defined in the Federal Information Security Management Act (amended in 2014 [17]) and deeply described in Federal Information Processing Standards (FIPS) Publication 199 [18]: *Confidentiality*, *Integrity*, and *Availability*. Moreover, we considered two other essential features of complex applications: *Authentication* and *Authorization* [19]. To conclude, we also took into account *Non-Repudiation*.

This section highlights a set of threats associated with the mentioned properties. In particular, the proposed threat model focuses its attention on menaces that target the smart home. For this reason, attacks addressing what is outside

the house's network are not considered. Therefore, if not differently specified, each of the threats described below could affect one of the following *attack targets*:

- other plug-ins;
- the smart home gateway application;
- other application(s) running on the physical gateway device;
- the gateway's operating system;
- devices belonging to the smart home.

It is also important to clarify that the model only considers menaces originating from a plug-in. To quickly reference the following threats, the paper assigns each of them a label composed of the letter "T" and a digit.

Some threats mention the plug-in scope. The latter is considered as the set of variables, files, or resources that the plug-in can legitimately access. To legitimately access a resource, the plug-in documentation has to include its possible use. Furthermore, access to the resource has to be carried out through the modalities envisaged by the platform (e.g., using the platform APIs). For instance, considering two plug-ins, "A" and "B", the scope of "A" includes its configuration file and does not include the configuration file of "B". Moreover, if a plug-in declares control only over a specific device (e.g., a lamp), every other device is outside its scope.

A. Confidentiality

The threats belonging to this class are related to illicit access to private information stored outside the plug-in's scope. Examples of private data could be a password, an access token, or users' sensitive information like their homes' locations.

- T1: a plug-in could *access and use* private data of other attack targets (i.e., data outside its scope).
- T2: a plug-in could *access and spread* private data of other attack targets (i.e., data outside its scope).

B. Integrity

The threats belonging to this class generally compromise the integrity of an attack target without impacting its availability. Therefore, even if some relevant information could be altered, the target may seem to work regularly. This behavior is particularly tricky because it could be much more difficult to detect that the target is compromised.

- T3: a plug-in could *alter the state* of other smart home *devices* outside its scope (e.g., turning on a lamp out of its scope).
- T4: a plug-in could *alter private data* of other *attack targets* outside its scope (e.g., changing a variable out of its scope, for instance, a measured temperature).

C. Availability

The following threats alter the availability of an attack target. The lack of availability could be temporary, or it could be so intrusive to deny the usage of the target completely. T5 and T6 usually make the target partially unavailable, while the other threats could generate complete unavailability.

- T5: a plug-in could *delay* the regular functionality of an attack target (e.g., a lamp took several seconds before turning on).
- T6: a plug-in could *alter* one of the regular functionalities of an attack target (e.g., users can turn on or off a lamp through its compromised integration, but power consumption data are not available anymore).
- T7: a plug-in could *alter* the regular functionality of an attack target, preventing the smart home users from using it. This threat implies two different scenarios: no one has control over the attack target (e.g., gateway's resource saturation up to a frozen state), or there is a third party able to replace the user in controlling the attack target (e.g., attackers could turn on and off the lights whenever they want).
- T8: a plug-in could physically *damage* an attack target (i.e., the gateway or another device). E.g., continuously repeated activation of a specific light until it burns out; gateway overheating.

D. Authentication

The threats belonging to this class occur when a plug-in can pretend to be a different entity (e.g., another plug-in, the gateway itself, or another house's device). When this happens, it could access functionalities and resources outside its scope.

- T9: a plug-in could interact with an attack target, pretending to be a different entity.

E. Authorization

The threats belonging to this class happen when a plug-in can have access to an authorization level higher than necessary. This escalation could be achieved through modifying the privileges assigned to it by the gateway or exploiting a mechanism for bypassing an authorization process. A higher authorization level could create unforeseen situations in the smart home system.

- T10: a plug-in could access an authorization level higher than expected.

F. Non-Repudiation

As specified in Section 1, many plug-ins can coexist inside a smart home gateway. Non-Repudiation of the communications between a plug-in and an attack target could be helpful to increase the degree of control over the entire gateway application. If a plug-in starts to compromise an attack target, having Non-Repudiation could help easily detect it. Compromising this security mechanism could be the starting point for other malicious behavior.

- T11: a plug-in could anonymously communicate with an attack target (i.e., there is no way to know that the plug-in establishes a specific communication).

V. Use Case

To validate the proposed threat model, the paper presents a use case based on an open-source smart home gateway software: Home Assistant [20]. In more detail, we developed

a number of plug-ins that, if analyzed by means of the model, reveal the presence of some of the previously described threats.

A. Home Assistant

Home Assistant (HAss) is a Python-based open-source software for home automation. Paulus Schoutsen started the project in 2013 and presented it in 2016 at the *Embedded Linux Conference and Open IoT conference*[1]. The purpose of Home Assistant is to be a central control system for smart home devices. It supports many different protocols, and it offers a front-end interface accessible both from web browsers and mobile applications (Android and iOS). As also declared in its documentation, Home Assistant was designed with a specific focus on users' privacy and local control. It has four different releases – HAss Core, HAss Container, HAss Supervised, and HAss Operating System (OS) – each presenting additional features with respect to the core version. According to Home Assistant analytics, at present time there are more than 150,000 active installations[2]. However, considering that analytics is not enabled by default, this number is indeed estimated to be four-fold[3].

Developers could expand HAss through two different types of plug-ins: *integrations* and *add-ons*. Integrations extend HAss Core directly and, for this reason, they are available on all the previously cited releases. Developers could also create custom integrations that override the behavior of default HAss integrations. Therefore, in line with their name, they are deeply integrated inside Home Assistant. On the contrary, add-ons are available only on HAss Supervised and HAss OS. They extend the capabilities of Home Assistant by installing additional applications more isolated from the core functionalities of the platform. Practically speaking, add-ons are Docker containers executed and managed by Home Assistant.

HAss provides its users with many different integrations and add-ons. Many of them are already available when the platform is installed; others can be downloaded directly from the platform's front end. Currently, Home Assistant offers 1010 integrations and 63 add-ons[2]. However, like many other open-source platforms, HAss was designed to be extensible by every programmer. Indeed, users can also retrieve extensions' source code (e.g., from GitHub) and manually configure HAss for running them. Among them, a popular place where users can retrieve extensions is the Home Assistant Community Store (HACS) [21]. This custom integration can be installed in HAss to simplify retrieving extensions developed by third-party programmers. Currently, HACS has 1094 available repositories.

In this vast space of possible plug-ins, installing an extension with malicious purposes could be easy. To demonstrate this hypothesis, the paper proposes in the next section some proof of concept integrations able to exploit some of the threats previously described in Section IV.

[1]https://www.linux.com/topic/embedded-iot/home-assistant-python-approach-home-automation/, last visited on March 3rd, 2022.

[2]https://analytics.home-assistant.io/, last visited on May 6th, 2022.

[3]https://www.home-assistant.io/blog/2021/11/12/100k-analytics/, last visited on May 6th, 2022.

B. Proof of Concept

This section aims to apply the proposed threat model to Home Assistant. According to the nature of the two kinds of plug-ins offered by the platform (i.e., integrations and add-ons), integrations are closer to the traditional plug-in concept. For this reason, they are more suitable for a first validation of the model. In this section, the paper describes some custom integrations used to demonstrate how the previously described threats could impact the platform's security properties.

Among the attack targets specified in Section IV, the developed integrations target only other Home Assistant integrations. For reasons of simplicity and brevity, this analysis addresses some of the previously described threats. We plan to demonstrate in future work all the presented threats on different attack targets. Table I summarizes which threats are exploited in the developed integrations.

1) Smart Home Configuration and Implementation Details: To validate the model, we developed three different integrations: *switch_target*, *light_altering_state*, and *light_simple_access*. These three integrations were developed and tested on a HAss Core "devcontainer" (a Docker container version of HAss designed for development)[4] and further tested on a HAss OS. The HAss OS was installed on a VirtualBox virtual machine running on a desktop PC with Windows 10. On both configurations, only the minimum default integrations and add-ons were enabled. No additional devices were connected to the virtual network.

For these testing components, we developed only the code necessary for the integrations to interact with Home Assistant. I.e., the developed plug-ins do not integrate any physical device. However, the lack of physical devices does not invalidate the outcomes of the analysis. The integrations could be easily expanded to interact with physical devices by developing an appropriate Python package or including an existing one.

As the name suggests, the first integration (*switch_target*) is targeted by the other two integrations. It represents a fake switch with two possible states: on and off. In addition, this integration contains a secret value that should remain private (i.e., it should be used only by the integration itself). Instead, the second and the third integrations are designed to manage two lights. The first light is used to demonstrate T3, T7, T9, and T11, while the second is used for T1, T2, T4, and T6. They have a brightness value and two possible states: on and off. Section V-B2 discusses how to generate the cited threats.

All the developed proofs of concept are available on GitHub under the MIT license[5].

2) Exploited Threats: The *light_simple_access* and the *light_altering_state* integrations behave like lights with additional malicious characteristics. Indeed, both these integrations are able to modify the status and the private data of the *switch_target*, an integration outside their scope.

[4]https://developers.home-assistant.io/docs/development_environment/, last visited on March 9th, 2022.

[5]https://github.com/Sarcares/HAss-TM-Integrations, last visited on March 23th, 2022.

To demonstrate how to interact with this integration using an illicit approach, we programmed a simple method able to get from Python's Garbage Collector (gc) a reference to the instance of the *switch_target* (Listing 1). To retrieve this reference, it is enough to know the name of the class (or the name of a parent class) and the name of the integration (in this specific example, we used the *SwitchEntity* class and "Switch Target", i.e., the name of the integration). Both integrations use this method to demonstrate some of the previously described threats.

```
def _get_target(self):
    for obj in gc.get_objects():
        if isinstance(obj, SwitchEntity):
            if obj.name == "Switch Target":
                return obj
```

Listing 1. Getting reference to an integration through the Garbage Collector.

Starting from confidentiality threats, when the *light_simple_access* is turned on in the front-end dashboard, the integration uses the previously obtained reference to access the secret stored inside the *switch_target* (a piece of information outside its scope). This behavior could lead to T1 or T2 according to what the malicious developer desires to do with the stolen data. Indeed, it could directly use the secret to access a private resource (e.g., if the secret is an access token) by materializing T1 or sending this information on a remote server (T2). Furthermore, to exploit integrity (T4) and availability (T6) threats, when this light is turned off, *light_simple_access* modifies the *switch_target*'s secret (T4). If the target integration uses this variable to perform its tasks (e.g., it is an access token used to send power consumption data to a different dashboard), this change could also create a partial lack of availability (T6). I.e., the switch seems to work regularly, but a part of its functionalities is compromised. Part of the code implemented for these threats is shown in Listing 2.

```
def read_secret(self):
    target_integration = self._get_target()
    return target_integration._my_secret

def alter_secret(self, new_value):
    target_integration = self._get_target()
    target_integration._my_secret = new_value
```

Listing 2. Reading and altering the secret of *switch_target*.

As we already said, to demonstrate more threats, we developed another proof of concept integration: the *light_altering_state*. To materialize T3, every time a user turns on or off this light, the integration alters the *switch_target*'s state—which is a state out of the integration's scope. If the switch is on, its state becomes off and vice versa (i.e., the switch is toggled). Even in this case, the integration uses the method shown in Listing 1 to retrieve the reference to the *switch_target*. Through this reference, it is possible to call a method for altering the state of the switch (Listing 3). Furthermore, the *light_altering_state* also demonstrates T7. Indeed, even if the user is always able to change the state of the switch, the presented integration is partially in control of it, revealing an availability threat in the use case.

```
def alter_state(self):
    target_integration = self._get_target()
    target_integration.toggle()
```

Listing 3. Altering the state of *switch_target*.

In addition, the developed integrations were also designed to demonstrate T9 and T11. Indeed, invoking the *toggle* method directly from the reference to the *switch_target* integration, the platform could not detect who changed the device's state (that seems to be changed by the original integration itself).

To conclude, Table I summarizes the threats described in this analysis grouped by category. The exploited threats have a checkmark (✓) while the menaces left out of this use case for reasons of brevity have a cross mark (✗).

TABLE I
THREATS EXPLOITED IN THE USE CASE INTEGRATIONS.

Threat Category	Threat	Exploited
Confidentiality	T1	✓
	T2	✓
Integrity	T3	✓
	T4	✓
Availability	T5	✗
	T6	✓
	T7	✓
	T8	✗
Authentication	T9	✓
Authorization	T10	✗
Non-Repudiation	T11	✓

VI. DISCUSSION

As described in Section III, smart homes could be complex systems formed by many distinct components, and extending such a system is never an easy task. Section IV highlights a set of possible threats to keep in mind while developing a plug-in for a smart home gateway, the central piece of the previously described architecture. Having a reference threat model during the development of their components could help programmers create more robust software solutions and avoid potentially critical behaviors. Furthermore, plug-in programmers could accidentally introduce security issues in their plug-ins, which could be particularly true if they do not have much experience on the platform they are expanding.

Applying the proposed model to Home Assistant integrations, we discovered that the platform has no firm isolation of its components. Hence, it is easy for plug-ins to violate the other integrations' confidentiality and obtain access to sensitive information. In addition, the platform does not check the integrity of the data stored inside its integrations. Therefore, each plug-in could modify data stored in others without being stopped by the platform. A plug-in could also easily target the availability of other components in the smart home. In Section V-B, we demonstrated only T6 and T7 (for availability property), but it is not difficult to deploy an integration able to exploit other threats (e.g., actuating a denial-of-service attack). Moreover, according to what we already exposed in Section V-B, Home Assistant does not implement sharp authentication

mechanisms, and it is relatively easy for the integrations to act without being identified (T10 and T12). To conclude, even if the analysis considers T11 as not found, this is a lack of the platform that, from our point of view, should be better addressed in future HAss releases.

However, Home Assistant's extensive set of possibilities is not only a downside. Indeed, being highly customizable led many developers and tinkerers to contribute to the platform's growth. At the time of writing, almost 3.000 people contribute to HAss core and its default integrations on GitHub[6]. Hence, considering how many people worldwide are involved in the development of this project, it is particularly important that plug-in programmers have a set of guidelines to follow during the creation of their additional components.

To sum up, to reduce the presence of the previously described threats is necessary to act on different components. First of all, the gateway itself (and its APIs) should enforce more controls. Then, the plug-ins should authenticate themselves while interacting with the platform (or among them). To conclude, it is necessary to give developers more support, providing them adequate documentation and a set of tools for validating their outcomes. We are convinced that having a reference threat model could be a first step to help developers ask themselves questions during their implementation choices and decrease the possibility of malicious behaviors.

VII. Conclusions

This paper focuses on smart home gateways extensible through plug-ins. Section III describes a typical gateway-centered smart home architecture, while Section IV proposes a threat model that could be applied in this scenario. The model represents a set of menaces that could have an impact on the main security properties (Confidentiality, Integrity, Availability, Authentication, Authorization, and Non-Repudiation) of a smart home system. Even if the model's primary focus is on the plug-ins, these threats could impact other attack targets in the described architecture (i.e., other plug-ins, other devices, and the gateway itself). To validate the proposed model, we presented a concrete use case. We developed a number of Home Assistant integrations that, if analyzed through the model, reveal the presence of some of the described threats.

Therefore, we recommend that developers take into account these menaces while programming smart home components. From one side, they should not implement these behaviors in their plug-ins. On the other side, programmers could create more robust and secure solutions by considering these threats while developing their smart home components.

A. Future Work

In a future study, as we already declared in Section V-B, we want to deeply validate the proposed model realizing a specific proof of concept for each presented threat. Moreover, we want to demonstrate that plug-ins could also address the other highlighted attack targets. In addition, we are interested in analyzing a large set of Home Assistant extensions to observe how many plug-ins present the described threats. Furthermore, we want to understand if developers could accidentally fall into one of these issues developing such a component. The final aim of these future researches will be to help developers create more secure and reliable smart home systems.

[6]https://github.com/home-assistant/core, last visited on May 6th, 2022.

References

[1] M. Schiefer, "Smart home definition and security threats," in *2015 Ninth International Conference on IT Security Incident Management IT Forensics*, 2015, pp. 114–118.

[2] D. Kumar, K. Shen, B. Case, D. Garg, G. Alperovich, D. Kuznetsov, R. Gupta, and Z. Durumeric, "All things considered: an analysis of IoT devices on home networks," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1169–1185. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/kumar-deepak

[3] K. Kafle, K. Moran, S. Manandhar, A. Nadkarni, and D. Poshyvanyk, "Security in centralized data store-based home automation platforms: A systematic analysis of nest and hue," *ACM Trans. Cyber-Phys. Syst.*, vol. 5, no. 1, dec 2021.

[4] M. Gajewski, J. M. Batalla, G. Mastorakis, and C. X. Mavromoustakis, "A distributed ids architecture model for smart home systems," *Cluster Computing*, vol. 22, no. 1, pp. 1739–1749, 2019.

[5] M. G. D'Souza, "Introduction to plug-ins," in *Expert Oracle Application Express Plug-Ins*. Springer, 2011, pp. 1–5.

[6] W. Xiong and R. Lagerström, "Threat modeling – a systematic literature review," *Computers & Security*, vol. 84, pp. 53–69, 2019.

[7] T. M. W. Group, "Threat modeling manifesto," 2022, [Online: accessed 10-Feb-2022]. [Online]. Available: https://www.threatmodelingmanifesto.org/

[8] L. Yang, C. Seasholtz, B. Luo, and F. Li, "Hide your hackable smart home from remote attacks: The multipath onion iot gateways," in *Computer Security*, J. Lopez, J. Zhou, and M. Soriano, Eds. Cham: Springer International Publishing, 2018, pp. 575–594.

[9] OWASP, "Threat modeling cheat sheet," 2022, [Online: accessed 10-Feb-2022]. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html

[10] OWASP, "OWASP Internet of Things Project," 2022, [Online: accessed 25-Feb-2022]. [Online]. Available: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project

[11] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, "Threat modeling-uncover security design flaws using the stride approach," *MSDN Magazine-Louisville*, pp. 68–75, 2006.

[12] M. N. Anwar, M. Nazir, and A. M. Ansari, "Modeling security threats for smart cities: A stride-based approach," *Smart Cities—Opportunities and Challenges. Springer*, pp. 387–396, 2020.

[13] B. Jelacic, D. Rosic, I. Lendak, M. Stanojevic, and S. Stoja, "Stride to a secure smart grid in a hybrid cloud," in *Computer Security*. Springer, 2017, pp. 77–90.

[14] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "Stride-based threat modeling for cyber-physical systems," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE, 2017, pp. 1–6.

[15] S. Pal, M. Hitchens, T. Rabehaja, and S. Mukhopadhyay, "Security requirements for the internet of things: A systematic approach," *Sensors*, vol. 20, no. 20, 2020.

[16] R. Heartfield, G. Loukas, S. Budimir, A. Bezemskij, J. R. Fontaine, A. Filippoupolitis, and E. Roesch, "A taxonomy of cyber-physical threats and impact in the smart home," *Computers & Security*, vol. 78, pp. 398–428, 2018.

[17] U. Congress, "Federal information security modernization act of 2014," *Public Law*, pp. 113–283, 2014.

[18] F. Pub, "Standards for security categorization of federal information and information systems," *NIST FIPS*, vol. 199, 2004.

[19] H. Kim and E. A. Lee, "Authentication and authorization for the internet of things," *IT Professional*, vol. 19, no. 5, pp. 27–33, 2017.

[20] N. C. Inc., "Home assistant," 2022, [Online: accessed 10-Feb-2022]. [Online]. Available: https://www.home-assistant.io/

[21] D. Goltsman, R. Snodgrass, J. Hills, and J. Sørensen, "Home Assistant Community Store," 2022, [Online: accessed 08-Mar-2022]. [Online]. Available: https://hacs.xyz/