



UP-Home: A Self-Adaptive Solution for Smart Home Security


Josival Silva

(Universidade Federal de Pernambuco, Centro de Informática, Brazil
 <https://orcid.org/0000-0001-9422-0781>, jss3@cin.ufpe.br)

Nelson Rosa

(Universidade Federal de Pernambuco, Centro de Informática, Brazil
 <https://orcid.org/0000-0001-9374-6351>, nsr@cin.ufpe.br)

Fernando Aires

(Universidade Federal Rural de Pernambuco, Departamento de Computação, Brazil
 <https://orcid.org/0000-0002-4007-3891>, fernandoaires@ufrpe.br)

Abstract: Smart home devices are vulnerable to attacks that put their users' security at risk. Vulnerabilities are discovered very frequently and can expose these devices through unsecured services. Meanwhile, the lack of standardisation in upgrade methods makes smart homes a potentially vulnerable environment. Furthermore, many manufacturers release their products and then abandon them, refusing to support security updates. As a result, security updates are needed to deal with the emergence of new attacks. There are several proposals to promote security in smart homes. However, there are rare solutions where changes for security purposes occur with little or no human intervention. This paper presents UP-Home, a self-adaptive solution that manages the security of smart homes. UP-Home aims to ensure that smart home devices meet the security requirements set by industry standards. The solution can continually identify and mitigate smart home security vulnerabilities. With autonomous computing techniques, UP-Home seeks to ensure the self-protection of devices and, consequently, the entire smart home. With the UP-Home evaluation, it was possible to notice significant improvements in the security of the smart home without any human intervention.

Keywords: IoT, smart home, IoT security, self-adaptive systems, vulnerability.

Categories: D.1.2, D.4.6, K.6.3, K.6.5

DOI: 10.3897/jucs.107050

1 Introduction

The use of technological devices has become significantly common, so homes have received the term smart [Marikyan et al., 2019]. A smart home is a residence with IoT (Internet of Things) devices connected to different clouds and home automation applications [Gubbi et al., 2013]. Smart homes have dozens of these devices, providing accessibility and convenience in an automated way never seen before in homes.

Expanding knowledge about the acceptance and adoption of technologies in the smart home and the willingness to recommend these technologies to others has motivated many researchers [Ferreira et al., 2023] [Hussin et al., 2023].

Home automation causes technological objects to create and exchange data without human intervention. Thus, a smart home can incorporate advanced features of automation systems to provide inhabitants with sophisticated monitoring and control over the functions of the environment [Schiefer, 2015].

Despite the benefits of smart homes, the risks to information security may outweigh the benefits of connectivity. First, the home environment has security vulnerabilities, which can lead to the leakage of sensitive data to hackers [Hassija et al., 2019] [Zhou et al., 2019]. Second, failure to properly manage device software updates leads to critical security vulnerabilities [Gu et al., 2020], e.g., outdated software stack and application obsolescence. Finally, many manufacturers abandon their devices and do not develop new updates. This abandonment allows vulnerable parameters or components to be ignored, compromising the entire network, devices and users.

Solutions for updating the software stack of IoT devices have adopted the OTA (Over The Air) method [Bauwens et al., 2020] as a basis, as it enables updates through the network. The proposed solutions allow the firmware update [Chandra et al., 2016], can use blockchain to improve the update process [Lee and Lee, 2017], cryptography [Frisch et al., 2017] or still focus on the integrity of the updates [Zandberg et al., 2019]. Finally, updates can even be scheduled to occur at pre-established times [Lin and Bergmann, 2016].

Although these works address update security, some solutions make intensive use of devices' computational resources [Lee and Lee, 2017] [Frisch et al., 2017], others are restricted to embedded devices that use specific microcontrollers [Frisch et al., 2017], or in some cases [Lin and Bergmann, 2016], they create schedules for updates. In this last solution, as vulnerabilities can appear anytime, the update must occur as soon as the vulnerability appears. Finally, an essential aspect of existing approaches is the absence of updates and actions to face security vulnerabilities without human intervention.

This paper presents a solution, named UP-Home, able to carry out automatic updates of devices to mitigate the security vulnerabilities of smart homes. UP-Home has been designed following the MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) [Salehie and Tahvildari, 2009]. In practice, UP-Home continuously monitors the devices' software stack (e.g., applications, middleware, operating system, firmware) and updates this stack so that it meets the security standards defined by the industry.

Experiments were carried out in a smart home where two gateways centralised the device management. With the result of monitoring smart home parameters, it was possible to adapt the software stack present in the devices. Given this scenario, it was also possible to verify that the level of security was improved by reducing vulnerabilities.

The remaining sections of this paper are organised as follows. Section 2 presents the concepts of IoT, IoT security and self-adaptive systems. Then, Section 3 describes the proposed solution. Section 4 presents an experimental evaluation of the proposed solution. Section 5 discusses work related to UP-Home. Finally, Section 6 presents contributions, limitations, and future work.

2 Background

This section presents the main concepts used in this paper: smart homes, IoT, IoT security, and self-adaptive systems.

2.1 Smart homes

The concept of smart home refers to the futuristic homes that have become a reality in recent decades with the growth of IoT in the most different domains. Darby [Darby, 2018] emphasises two main definitions of smart homes. The first focused on the home and the user, understanding the smart home as a highly automated residential building with integrated appliances, emphasising modern technology, convenience and (domestic) efficiency. The second definition focuses on systems that concentrate on the energy performance of buildings, ancillary services and distributed energy generation and how they can be addressed using information and communication technology.

2.2 IoT

The Internet of Things (IoT) is an open network of intelligent objects that can self-organise, share information, data, and resources, and respond and adapt to environmental changes [Madakam et al., 2015]. The IoT consists of the ubiquity of various objects or things, including sensor technologies, actuators, and mobile devices. These devices interact with one another to accomplish shared goals using both wired and wireless networks [Tan and Wang, 2010]. On this basis, achieving higher integration with the actual world through IoT is possible, even reducing the demand for human intervention. IoT helps transform traditional objects, such as lights, locks, and televisions, into smart objects.

IoT devices are typically limited in computational resources, including power, CPU, and memory capacity [Zikria et al., 2018]. Because of those limitations, different types of software operate under the most appropriate storage. The software stack comprises a set of software systems that operate by layering on top of each other to provide the required functionality on a device. The stack includes software systems that range from IoT applications to device-specific software, e.g., firmware.

The IoTSEF (Internet of Things Security Foundation) categorises IoT devices into different classes, focusing on how these devices integrate into a hub [IoTSEF, 2018]:

- Class 1. Fully controlled and/or connected, where interfaces such as IoT device control, data collection, and management are fully possible.
- Class 2. Partially controlled and/or connected. The hub device can perform some tasks but with some limitations under the device, such as sending updates and managing traffic.
- Class 3. The most basic type of integration is where the hub does not control or manage IoT device functions such as updating or data collection

2.3 IoT security

According to Nieves et al., [Nieves et al., 2017], information security protects information and systems. This protection defends against unauthorised access, use, disclosure, interruption, modification, or destruction of the information to provide confidentiality, integrity, and availability. This concept becomes even more critical in an IoT environment as new security issues arise daily. Finally, one of the main reasons that increase the complexity of improving security in IoT environments is the heterogeneity and large number of devices [Zhang et al., 2014].

Security is indispensable for smart homes [Komninos et al., 2014], as they can be exposed to various services provided through wired and wireless networks. In this environment, vulnerabilities may occur due to flaws in the development, and maintenance of applications or even improper use by users. It is important to keep devices up to date as required by industry standards such as ETSI TS 103 645¹ of the European Telecommunications Standards Institute (ETSI).

Software systems such as CVE (Common Vulnerabilities and Exposures)² and NVD (National Vulnerability Database)³ focus efforts on identifying, defining, and cataloguing publicly disclosed cybersecurity vulnerabilities. These databases are used as a reference in penetration testing tools like OpenVas⁴.

The severity or criticality of a vulnerability is a measure of the potential risk associated with it. Vulnerabilities are evaluated based on several factors, such as how easily they can be exploited, the potential impact on the affected system, and the sensitive information that can be obtained or altered. For example, the criticality levels in OpenVas are classified as high, medium, and low, log and false positive, depending on the severity of the vulnerability [Aksu et al., 2019]. A vulnerability, e.g., High, can have a significant impact and is therefore considered a top priority for remediation. In turn, a low critical vulnerability, e.g., Log, may have a more limited impact and be considered a lower priority for correction.

2.4 Self-adaptive systems

A self-adaptive software evaluates its behaviour and changes it when the evaluation indicates that this software is not doing what it is intended to do or when it can improve functionality or performance [Lehman, 1996]. Salehie and Tahvildari [Salehie and Tahvildari, 2009] state that the critical point of self-adaptive software is that its life cycle should not be interrupted after its initial development and configuration. Weyns [Weyns, 2017] defines that a self-adaptive system must deal autonomously with environmental changes and uncertainties, impacting the system and its objectives. Autonomy means adjusting the system with little or no human intervention. According to Kephart [Kephart and Chess, 2003], self-management is a characteristic of autonomous computing systems. This characteristic aims to lessen the need for system administrators to perform operation and maintenance actions. Thus, self-management provides users with a machine

¹ <https://shre.ink/9pB7>

² <https://cve.mitre.org/>

³ <https://nvd.nist.gov/>

⁴ <https://openvas.org/>

running at peak performance without interrupting its execution. Autonomous computing [Salehie and Tahvildari, 2009] has four self-adaptive properties: self-configuring, self-healing, self-optimising, and self-protecting.

3 UP-Home

UP-Home (UPdated Home) is a solution that aims to ensure that the software stack on smart home devices is always up to date, thus guaranteeing their security. In this way, it is sought that these devices meet the security standards established by the industry and have continuous security updates that can mitigate vulnerabilities in the smart home. In addition, the level of security is improved without the need for human intervention or complete stoppage of the applications deployed in the smart home.

3.1 Principles

UP-Home was designed according to some principles:

Self-protection. UP-Home focuses on the self-protection property to detect and mitigate potential security threats at runtime. Using self-protection strategies through UP-Home allows the devices' software stack to be always up-to-date and able to mitigate security breaches (reactive strategy) or anticipate attacks (proactive strategy). The environment where smart home devices are inserted is dynamic, where changes can occur at any moment, e.g., with the entry of a new device. In addition, many systems must operate without interruption, even if there are changes in the operating environment, resource availability, user requirements, or failures. Therefore, UP-Home must enable the software stack to adapt to context changes and make the necessary adjustments without human intervention, for example, to act upon the perception of outdated devices that result in vulnerabilities. UP-Home proposes a secure environment that acts autonomously on security vulnerabilities. In addition, UP-Home promotes security by monitoring parameters, e.g., Telnet and FTP services. Hence, actions are taken that can fix issues that increase risks to the smart home and its users, such as hacking and device unavailability.

Parametric and compositional adaptation. Adaptive systems commonly implement two types of adaptation: compositional and parametric [McKinley et al., 2004]. Parametric adaptation involves adjusting the system's behaviour through changes in one or more parameters. In turn, compositional adaptation changes the system's behaviour through the addition, exchange, removal, or reconnection of components.

5W-1H. Self-adaptive solutions [Salehie and Tahvildari, 2009] usually consider the following questions: where to adapt, when to adapt, what to adapt, why to adapt, who adapt, and how to adapt. Therefore, as UP-Home is a self-adaptive solution, its project also explicitly addresses each of these points [Krupitzer et al., 2015]:

- Where. Adaptation is performed on the device where the update is required, thus maintaining it with security parameters that meet the standards set by the industry.

- **When.** Adaptation occurs reactively, that is, in reaction to an event: detection of device vulnerability; when a new firmware version is available, if there is any sudden change in network traffic; or if monitoring detects a new device in the smart home.
- **What.** The reason for adaptation is the violation of some security goals. For example, if the entire software stack must always be up-to-date, a change is necessary every time the software on any smart home device becomes outdated.
- **Who.** This question defines the degree of human automation and adaptation involvement. In UP-Home, automation is carried out without human intervention.
- **How.** How adaptation occurs can be seen in three sublevels: approach, decision criterion, and degree of decentralization. The UP-Home approach is external, dividing the system into adaptation logic and managed resources. Decision criteria are based on goals, detailed in Section 2. Regarding the degree of decentralization, logic is centralized, reducing its vulnerability.

MAPE-K. MAPE-K [Kephart and Chess, 2003] was used as a reference model for the adaptation logic of the proposed solution. In this case, Telnet and FTP services and components of the device's software stack are monitored, e.g., firmware and OS. Based on this monitoring, vulnerabilities that violate security targets are checked. Finally, actions are defined to adjust the software stack according to the observed violations.

Use of security standards. The UP-Home's design was inspired by good security practices aimed at updating software for IoT devices, e.g., ETSI (European Telecommunication Standards Institute). ETSI defines IT standards, including those that support parties involved in developing and manufacturing consumer IoT, guiding them on how to apply protection to their products. A subset of the requirements of these standards works as a checklist of security requirements observed by UP-Home. In UP-Home, there is the management of security parameters, with monitoring of the smart home and cloud data set. This way, the knowledge base grows and supports autonomous security management, thus applying self-protection to the smart home.

Figure 1 presents an overview of the scenario where the UP-Home solution operates under the general concepts of self-adaptive systems.

The two main components of this figure are the Environment and the Self-adaptive System. The environment is everything that interacts with the self-adaptive system but does not itself make up a component of it. Physical and virtual entities not controlled by the self-adaptive system are present in the environment. Despite this, the Self-adaptive System keeps an eye on it and can use the collected data to determine whether an adjustment is necessary. The Managed System and the Managing System components comprise the self-adaptive system. Adaptations happen when the objectives defined by the Self-Adaptive System are violated.

In Figure 1, the Managed System includes all devices in the smart home and receives actions in their behaviour to deal with changes occurring in the Environment. Supporting these changes in the Managed System consists of two main elements: Sensors and Actuators. The Sensors capture information about

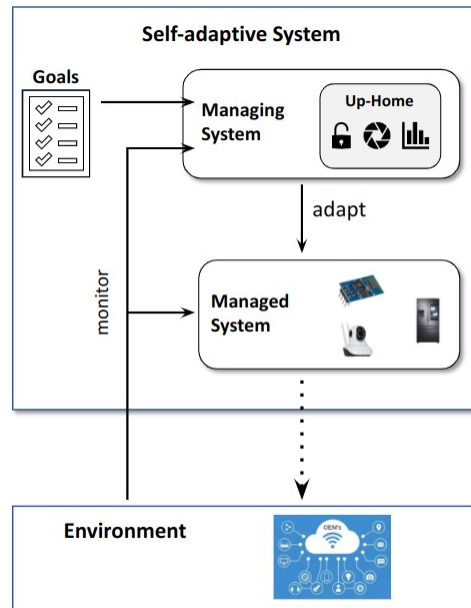


Figure 1: General overview

the monitored parameters at runtime, and the actuator performs the changes in the Managed System. The central element of Self-Adaptive Systems is the Managing System. It is the adaptation engine, which, in the case of UP-Home, is structured according to MAPE-K. The strategy used instantiates MAPE-K to promote self-protection in the smart home. The Managing System monitors the Environment and the Managed System, and when there are violations of the Goals, it carries out adaptation actions in the Managed System. Violation can be characterized as a change in the monitored parameters leading to a disallowed value.

Finally, UP-Home is a solution that provides self-protection, adopting security measures to keep IoT devices safe to meet the Security Goals recommended by the industry. The software stack, for example, needs to be updated to mitigate the presence of vulnerabilities. In addition, changes can occur in Goals; new Goals can appear or be removed at runtime, requiring updates to the Managing System, sensors, or actuators. Goals define high-level invariants that must be preserved at runtime. Thus, the Managed System is the element that needs to be adapted so that the invariants are satisfied.

3.2 Architecture

Following the principles presented in Section 3.1, UP-Home is the Managing System implementing MAPE-K to improve smart home security. Figure 2 presents details of the UP-Home architecture. The Monitor collects (through sensors) data from the Managed System and the Environment. It then sends this data to the Analyzer and updates the Knowledge. The second component, the Analyzer,

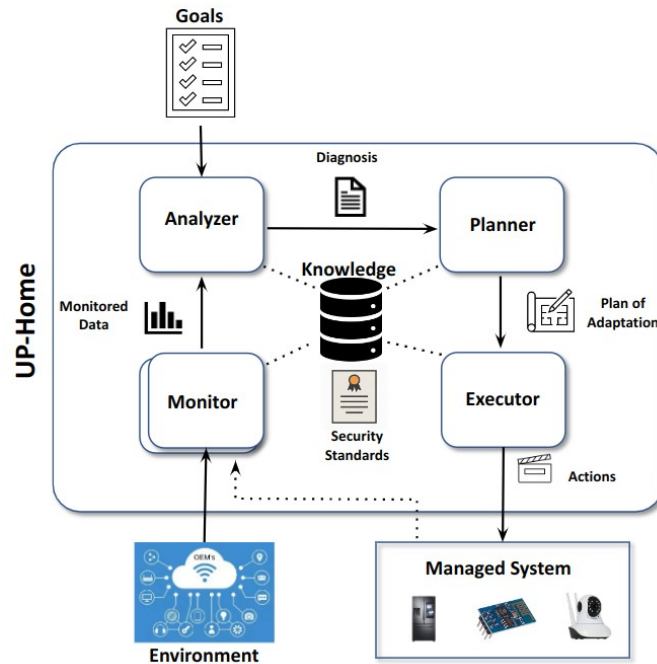


Figure 2: UP-Home architecture

receives the data from the monitor and decides if there is a need to adapt the Managed System. When deciding, the Analyzer uses the monitored data and the Goals. If there is a need for adaptation, the Analyzer forwards this information to the Planner. The Planner receives the information from the Analyzer and builds an adaptation plan. This plan contains all actions needed to restore the system to the defined Goal. The plan is then passed to the Executor. The Executor is responsible for executing all actions contained in the change plan. These actions are performed in the Managed System. Finally, Knowledge stores all the information used by the other components. At each MAPE-K interaction, Knowledge is updated, improving knowledge about the Managed System.

3.3 Implementation

Section 3.2 mentions that UP-Home is the Management System that adapts the Managed System (smart home). The Managed System is composed of IoT devices from different manufacturers, an HA gateway (homeassistant)⁵ that integrates the smart home devices, and another gateway (RoutersOS)⁶ that controls the connections of these devices. The HA gateway allowed monitoring of device parameters, e.g., the device software version. The RouterOS gateway manages

⁵ <https://www.home-assistant.io/>

⁶ <https://mikrotik.com/>

access to the network, for example, controlling bandwidth, allowing or blocking access.

The implementation of UP-Home was performed with open source solutions, e.g., Openvas, ADB commands (Android Debug Bridge)⁷ and CLI (Command Line Interface)⁸. In the case of a compositional adaptation, when a component of the software stack is obsolete, it is replaced. In parametric adaptation, parameters that may cause security vulnerabilities are adjusted, e.g., an open port on a given device where an active service is prone to intrusion.

The compositional adaptation of UP-Home was performed using the HA and RouterOS gateways. HA allows the integration of different devices into a single component. HA can oversee and provide security information, including the firmware version of devices. HA was instantiated in a Docker container, installed on the Raspberry PI OS⁹. The routerboard, which operates with the RouterOS operating system, was used as a hub for the smart home's network connections. Through an open-source API¹⁰, it was possible to keep the RouterOS updated, thus changing the composition of this device. This API was used in UP-Home for the parametric self-adaptation actions so that the parameters were changed to mitigate the vulnerabilities linked to the enabled ports.

In addition to monitoring parameters in the HA, OpenVas was used, which made it possible, through penetration tests, to monitor vulnerabilities in the smart home. The Analyzer used the data monitored by HA and OpenVas (see Section 3.2) to decide the necessary actions in the compositional and parametric adaptations.

3.4 Adaptation of composition

Regarding the compositional adaptation, UP-Home performs OTA updates considering the different classes of smart home IoT devices. Some devices are fully managed, others are partially managed, and those operate with restrictions on collecting device information [IoTFS, 2018]. The type of device management directly impacts the change actions that can be performed. Table 1 shows an overview of these actions.

Class	Integration	Monitoring	Development	Update	Rollback
C1	✓	✓	✓	✓	✓
C2	✓	✓	✗	✓	✗
C3	✓	✗	✗	✗	✗

Table 1: Classes of devices and their corresponding firmware intervention options

As seen in this table, there are three different classes of devices:

Class C1. Class C1 devices operate without restrictions; they are open source, and anyone can develop their solution. Thus, in this class of devices, it is possible to monitor, develop, update, and revert the firmware with security measures,

⁷ <https://developer.android.com/studio/command-line/adb?hl=en>

⁸ <https://learn.microsoft.com/pt-br/appcenter/cli/>

⁹ <https://www.raspberrypi.com/>

¹⁰ <https://pypi.org/project/RouterOS-api/>

such as authentication and validation of firmware integrity. The most common devices in this class are microcontrollers, such as Esp32 and Esp8266.

Class C2. Class C2 devices are partially managed; they have firmware with proprietary code and can be integrated into the gateway, but they only have permissions for monitoring their parameters. According to IoTSF [IoTSF, 2018], most devices in a smart home are of Class C2. Class C2 devices do not support enforcing security measures, for example, ensuring authenticity and integrity in firmware updates. Despite this, it is possible to identify the firmware version running on the devices.

Class C3. Class C3 devices are proprietary code and operate under greater restriction than Class C2. Because of this, they do not allow monitoring of security parameters. Commonly, Class C3 devices are IP cameras, smart TVs that do not have the Android operating system, and ECU (Engine Control Unit), controllers used in photovoltaic systems. Despite this difference, some devices can switch from one class to another by changing the firmware, for example, the Sonoff¹¹. Sonoff is a Class C2 device with some restrictions on parameter management imposed by the manufacturer. However, these devices become managed by replacing the original firmware with Tasmota¹². This action enables managing security parameters to develop, monitor, and control firmware updates.

3.5 Parametric adaptation

In addition to changing components of the devices' software stack, UP-Home can modify parameters in the hub to mitigate vulnerabilities (parametric adaptation). For this purpose, OpenVas was used, which, like most vulnerability scanning tools, allows observing the smart home network connections, e.g., telnet, FTP, and SSH access. This tool uses a database of vulnerabilities that is updated daily. Furthermore, such vulnerabilities are catalogued with their respective level of criticality. It is essential to note the criticality level of vulnerabilities in decision-making, allowing the most critical vulnerabilities to be mitigated first. Parameter changes using the RouterOS gateway API, e.g., port blocking.

4 Evaluation

The UP-Home evaluation has three objectives: (1) to show the effectiveness of UP-Home through experiments where smart home vulnerabilities are detected and resolved and to evaluate the time spent on (2) parametric and (3) compositional adaptations. With this, it became possible to determine whether the security level of the smart home is improved due to the actions of UP-Home. In addition, it allowed evaluation of the response time of the solution when dealing with different amounts of vulnerabilities.

A real scenario was used to carry out all the evaluation experiments, as shown in Figure 3. In this scenario, the smart home has 26 IoT devices, a gateway (RouterOS) that provides the connections of the IoT devices, and another gateway (HA) to integrate the devices. In addition, a scanner (OpenVas) was used that verifies vulnerabilities with penetration tests.

¹¹ <https://sonoff.tech/>

¹² <https://tasmota.github.io/docs/>

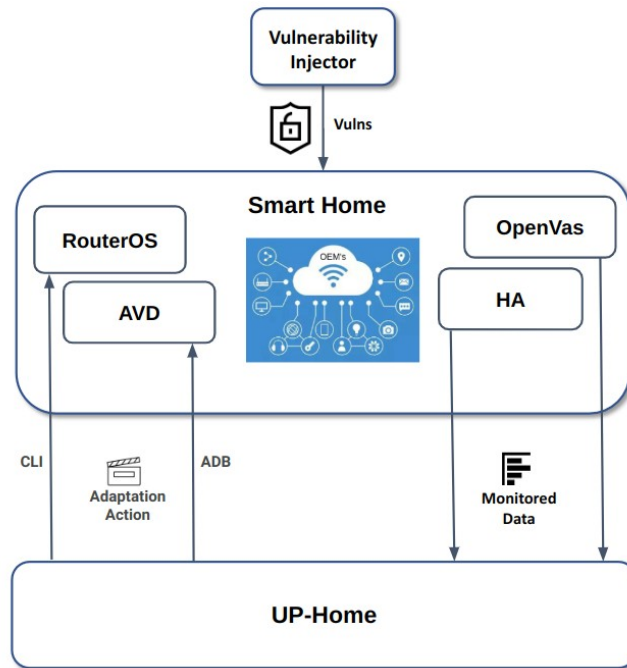


Figure 3: Setup used in the UP-HOME evaluation

In this setup, there is also a vulnerability injector (Vulnerability Injector), which creates a vulnerable condition (vulns) in the smart home. The vulnerability injector intentionally introduces vulnerabilities into IoT devices to test their security. This action can be done for various reasons, such as determining the effectiveness of security measures and identifying and fixing vulnerabilities, e.g., active Telnet or FTP, outdated devices. It is worth noting that penetration tests are widely used [Bacudio et al., 2011] [Shah and Mehtre, 2015] to inject vulnerabilities, and these vulnerabilities can lead to data leakage. All experiments consist of injecting vulnerabilities, observing how UP-Home detects them (through the Monitor), and deciding whether a change is necessary (Analyser). If necessary, the other elements of the solution (Planner and Executor) come into play, sending ADB (Android Debug Bridge) commands to the AVD (Android Virtual Device) or CLI (Command Line Interface) to RouterOS.

4.1 UP-Home in action

UP-Home runs in the scenario shown in Figure 3 to achieve the first objective. Openvas scans device connections, looking for vulnerabilities.

Considering the dynamic smart home environment, different devices can be present during each vulnerability scan. For this reason, 13 scans were carried out on different days with OpenVas to obtain the list of vulnerabilities present in the smart home. At the end of the thirteenth scan, 16 vulnerabilities were identified

in the penetration test. Figure 4 presents the list of vulnerabilities with their respective criticality levels.

Name	Severity
libunnp Multiple Buffer Overflow Vulnerabilities	10.0 (High)
Operating System (OS) End of Life (EOL) Detection	10.0 (High)
MikroTik RouterOS RCE Vulnerability (CVE-2021-41987)	8.1 (High)
MikroTik RouterOS < 6.46.7, <= 6.47.3, 7.x DoS Vulnerability	7.5 (High)
MikroTik RouterOS DoS Vulnerability (CVE-2022-36522)	6.8 (Medium)
MikroTik RouterOS < 6.48.2 Multiple DoS Vulnerabilities	6.5 (Medium)
MikroTik RouterOS <= 6.48.6 DoS Vulnerability	6.5 (Medium)
jQuery < 1.9.0 XSS Vulnerability	6.1 (Medium)
SSL/TLS: Report 'Anonymous' Cipher Suites	5.4 (Medium)
Weak Host Key Algorithm(s) (SSH)	5.3 (Medium)
Weak Key Exchange (KEX) Algorithm(s) Supported (SSH)	5.3 (Medium)
Cleartext Transmission of Sensitive Information via HTTP	4.8 (Medium)
FTP Unencrypted Cleartext Login	4.8 (Medium)
Telnet Unencrypted Cleartext Login	4.8 (Medium)
SSL/TLS: Deprecated TLSv1.0 and TLSv1.1 Protocol Detection	4.3 (Medium)
Weak Encryption Algorithm(s) Supported (SSH)	4.3 (Medium)

Figure 4: Vulnerabilities found

As a result of the scans, reports are generated and accessed by UP-Home. In addition, UP-Home monitors device integration data through the HA (home-assistant). Finally, the reports are analyzed, and if vulnerabilities are detected, actions are planned and executed to block or update the RouterOS gateway through CLI or ADB commands, preventing the vulnerability from persisting. Having created a filter in the RouterOS gateway firewall, the port and host corresponding to the vulnerability are no longer accessible by other devices. The update action comprises sending a command to an outdated host, causing it to update the firmware.

UP-Home acted on each report in the scan, analyzing the results and planning and executing actions to mitigate the vulnerabilities. UP-Home changes a parameter in the RouterOS gateway, preventing network communication with the target and vulnerable port. The results of this action by UP-Home are shown in Figure 5.

At the end of 10 scans, 33 vulnerabilities were found. As shown in Figure 5, the number of vulnerabilities decreased with each UP-Home intervention. The vulnerabilities were resolved with each successive scan, and no new vulnerabil-

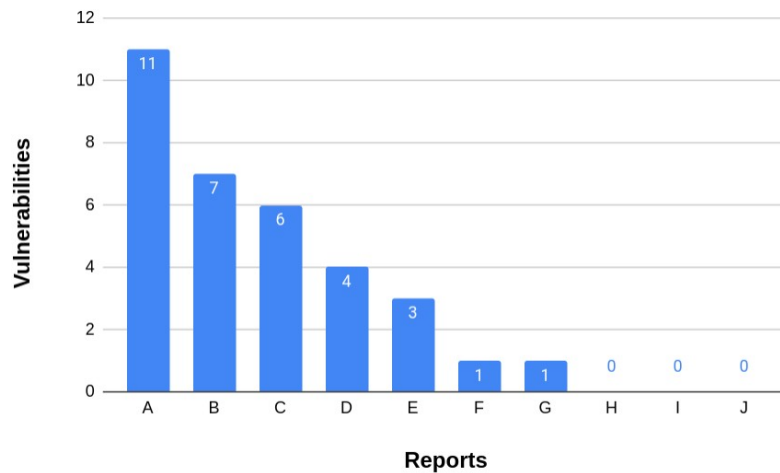


Figure 5: Vulnerabilities resolved by scanning

ities were detected in the last three scans. Hence, it was possible to perceive that UP-Home automatically improved the security of the smart home. The vulnerabilities presented in Figure 4 are common in different devices, increasing the possibility of attacks since more devices have compromised security. Table 2 presents details of how the incidence of vulnerabilities found in the hosts occurred.

The Host column identifies the device, and CVSS (Common Vulnerability Scoring System)¹³ defines the criticality level of the vulnerability. Low, Medium, and High vulnerabilities correspond to CVSS values of 0.1 to 3.9, 4.0 to 6.9, and 7.0 to 10.0, respectively. The Action shows what UP-Home has done to mitigate the vulnerability. Two actions can occur, namely block or update. The block alters a parameter in RouterOS. The block prevents the possibility of an insecure connection to the respective device. In the update, an element of the software stack is swapped on the outdated device, in this case, the firmware. Figure 6 shows the result of UP-Home's actions on the detected vulnerabilities.

Thirty-three vulnerabilities were detected, 26 were blocked, and seven devices were updated. None of the vulnerabilities found remained unresolved, i.e., all were mitigated.

4.2 Evaluation of parametric adaptation

In the scenario described in Figure 7, Vulnerabilities Injector 1 was executed, where one type of vulnerability was replicated.

In this scenario, experiments were carried out to evaluate the parametric adaptation strategy implemented by UP-Home. In practice, adaptation consists

¹³ <https://www.first.org/cvss/>

Host	CVSS	Action	Vulnerability
01	6.1	block	jQuery <1.9.0 XSS Vulnerability
01	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
01	10.0	update	Operating System (OS) End of Life (EOL) Detection
01	10.0	block	libupnp Multiple Buffer Overflow Vulnerabilities
02	8.1	update	MikroTik RouterOS RCE Vulnerability (CVE-2021-41987)
02	4.3	block	Weak Encryption Algorithm(s) Supported (SSH)
03	5.3	block	Weak Host Key Algorithm(s) (SSH)
03	5.3	block	Weak Key Exchange (KEX) Algorithm(s) Supported (SSH)
03	6.5	update	MikroTik RouterOS <6.48.2 Multiple DoS Vulnerabilities
03	6.5	update	MikroTik RouterOS <= 6.48.6 DoS Vulnerability
03	7.5	update	MikroTik RouterOS <6.46.7, <= 6.47.3, 7.x DoS Vulnerability
03	2.6	block	Weak MAC Algorithm(s) Supported (SSH)
03	4.8	block	FTP Unencrypted Cleartext Login
03	4.8	block	Telnet Unencrypted Cleartext Login
03	6.8	update	MikroTik RouterOS DoS Vulnerability (CVE-2022-36522)
03	6.1	block	jQuery <1.9.0 XSS Vulnerability
03	4.3	block	SSL/TLS: Deprecated TLSv1.0 and TLSv1.1 Protocol Detection
03	5.4	block	SSL/TLS: Report 'Anonymous' Cipher Suites
04	7.5	update	MikroTik RouterOS <6.46.7, <= 6.47.3, 7.x DoS Vulnerability
04	4.8	block	FTP Unencrypted Cleartext Login
04	4.8	block	Telnet Unencrypted Cleartext Login
04	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
04	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
04	6.8	update	MikroTik RouterOS DoS Vulnerability (CVE-2022-36522)
04	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
04	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
04	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
04	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
04	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
04	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
05	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
05	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
06	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
06	4.8	block	Cleartext Transmission of Sensitive Information via HTTP
07	4.3	block	Weak Encryption Algorithm(s) Supported (SSH)

Table 2: Vulnerabilities resolved

of executing the block action. After the execution of the injector, the time required for UP-Home to detect and mitigate them was evaluated. It was considered the time when no vulnerabilities were injected as the baseline, and then the number of vulnerabilities was increased in intervals of 20 up to a maximum of 200 vulnerabilities.

In particular, the first experiment injected was the jQuery < 1.9.0 XSS vulnerability, catalogued under CVE-2012-6708, which has a medium criticality level and CVSS grade of 6.1. This vulnerability has the jQuery function (`strInput`) and does not reliably differentiate between HTML selectors. In vulnerable versions, jQuery determined whether the input was HTML by looking for the '<' character anywhere in the string, giving attackers more flexibility when constructing a malicious payload. Figure 8 presents the result of the adaptive actions with different vulnerabilities.

Observing Figure 8, the execution time is linear, corresponding to the increased number of injected vulnerabilities. Mitigating the jQuery < 1.9.0 XSS (CVE-2012-6708) vulnerability in these experiments can serve as a preventive measure for various other types of vulnerabilities as well:

- CVE2012-5958. The Libupnp Multiple Buffer Overflow Vulnerabilities can be

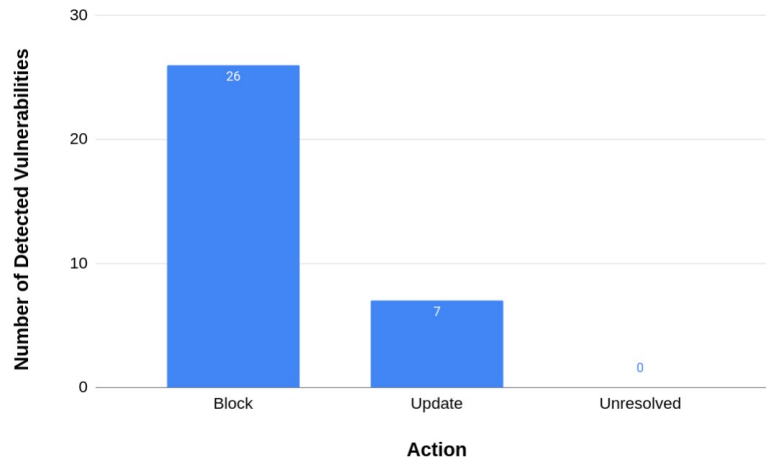


Figure 6: Number of resolved vulnerabilities

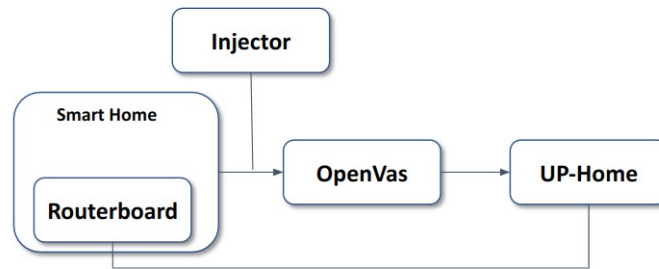


Figure 7: Injector vulnerabilities 1

exploited by attackers to execute arbitrary code within the affected device's context. However, unsuccessful attempts may only result in the application crashing.

- CVE2007-1858, CVE-2014-0351. SSL/TLS: Report' Anonymous' Cipher Suites may allow remote attackers to obtain confidential information or have other unspecified impacts.
- CVE-2007-1858, CVE-2014-0351. With SSL/TLS (Deprecated TLSv1.0 and TLSv1.1 Protocol Detection), an attacker can use known cryptographic flaws to eavesdrop on the connection between clients and the service to gain access to sensitive data transferred over the secure connection.

A second injector (Vulnerability Injector 2) was used to expand the evaluation, as shown in Figure 9. This second injector allowed other types of vulnerabilities to be used in the experiment. The 3 table lists the types of vulnerabilities and their respective criticality levels.

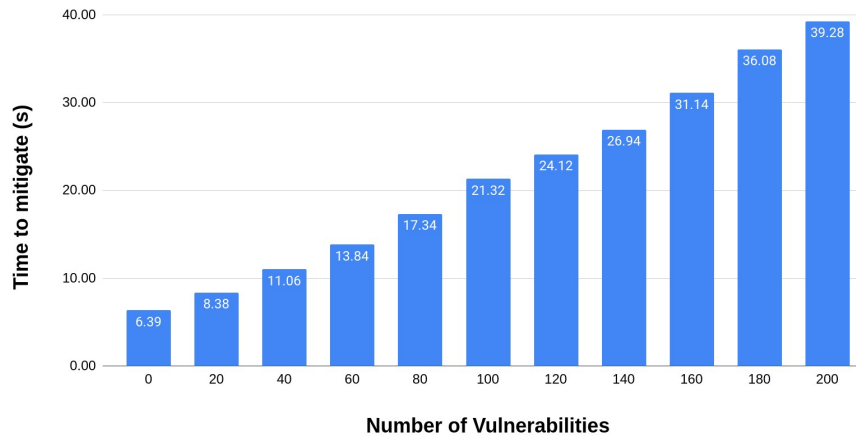


Figure 8: Runtime with different workloads

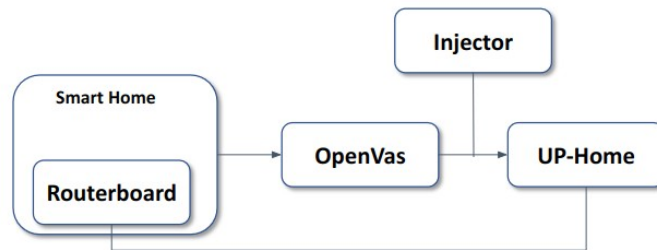


Figure 9: Vulnerability injector 2

The experiments performed next consisted of injecting vulnerabilities and measuring the time spent by UP-Home while the number of discovered vulnerabilities grew. The adaptation action comprises sending CLI (Command Line Interface) commands to the RouterOS gateway through API, blocking the vulnerabilities in the home network. Table 3 lists the types of vulnerabilities and their respective criticality levels.

Some vulnerabilities can be found in more than one device. Hence, it is possible to have more vulnerabilities than the list in Table 3. In this configuration, the injector purposefully expanded the smart home's vulnerabilities. These vulnerabilities are replicated to evaluate the execution time to mitigate them, simulating a possible more adverse condition. In practice, the injector multiplies the results obtained in the OpenVas scans, leading to more actions to reduce the vulnerabilities.

Figure 10 shows the results of this experiment.

The workload named 36 Vulns (baseline) corresponds to 36 vulnerabilities detected. The other workloads are the baseline multiplied by 2 (72 Vulns), 3

CVSS	Name
10.0	libupnp Multiple Buffer Overflow Vulnerabilities
6.1	jQuery < 1.9.0 XSS Vulnerability
5.4	SSL/TLS: Report 'Anonymous' Cipher Suites
5.3	Weak Host Key Algorithm(s) (SSH)
5.3	Weak Key Exchange (KEX) Algorithm(s) Supported (SSH)
4.8	Cleartext Transmission of Sensitive Information via HTTP
4.8	Telnet Unencrypted Cleartext Login
4.8	FTP Unencrypted Cleartext Login
4.8	Telnet Unencrypted Cleartext Login
4.3	Weak Encryption Algorithm(s) Supported (SSH)
4.3	SSL/TLS: Deprecated TLSv1.0 and TLSv1.1 Protocol Detection
4.3	Weak Encryption Algorithm(s) Supported (SSH)

Table 3: Vulnerabilities detected

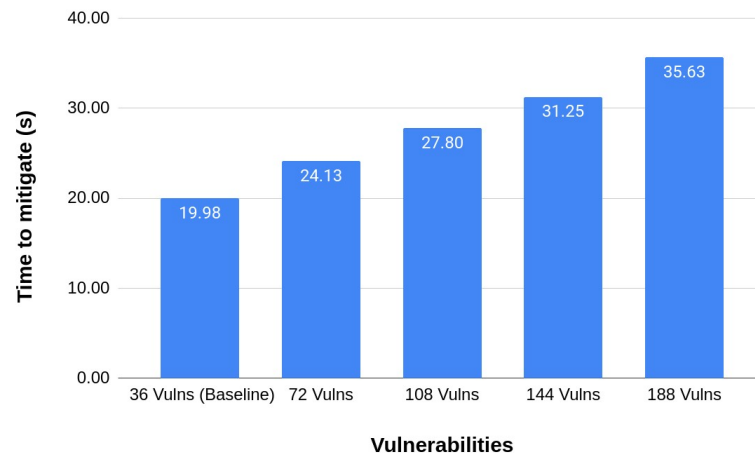


Figure 10: Runtime with different workloads

(108 Vulns), 4 (144 Vulns) and 5 (180 Vulns), thus simulating a more critical condition. Observing Figure 10, it can be seen that the execution time is linear. Figure 11 compares the other workloads with the baseline.

As shown in Figure 11, with twice as many vulnerabilities as the baseline (72 Vulns), UP-Home needed 21% more time. Moreover, in another workload (180 Vulns), which corresponds to 5 times the baseline, UP-Home took 78%.

4.3 Evaluation of compositional adaptation

In this last set of experiments, the UP-Home adaptation time was evaluated when the compositional adaptation strategy was used, i.e., the execution of the update

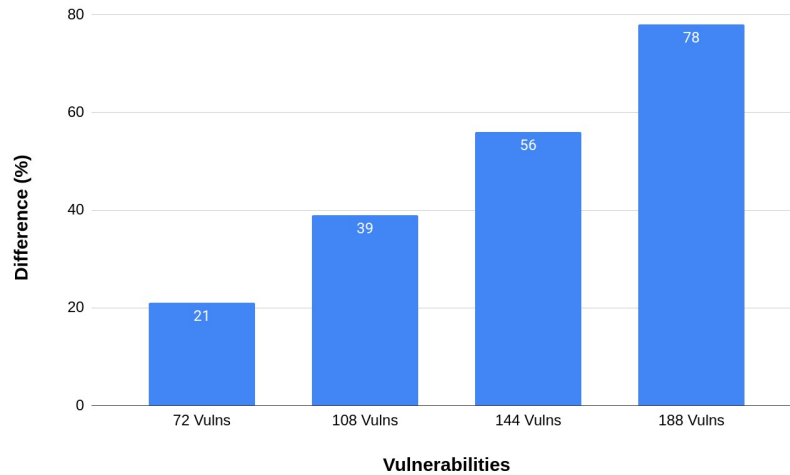


Figure 11: Comparison between baseline and other workloads

action. In these experiments, monitoring was done through the HA and the cloud, where UP-Home reads security parameter information. UP-Home monitors the latest version of the software available from four different manufacturers and the version of the devices present in the smart home. This data was analyzed, and update actions were performed on outdated devices. For this experiment, a smartphone with an Android operating system was utilized, along with two security cameras from different manufacturers, namely Intelbras and V380. Additionally, a Sonoff smart switch and the HA gateway were also included.

Update actions take place via ADB commands sent to an AVD. The update process is carried out by a script that sends commands to update the application. Update download and installation times are outside the scope of the assessment, as during this time, the device typically restarts and loses communication.

Figure 12 presents the results of the experiments.

In this figure, the No Update label refers to the time taken for monitoring, analysis, and planning when UP-Home decides that no device needs to be updated. As noted, Smartphone and Camera B had the shortest and longest update times, respectively. This difference is because each application has different behaviours, e.g., application opening time, application screen access time, and the number of buttons that need to be clicked by the script.

5 Related work

Existing solutions related to UP-Home can be organised into two broad categories: self-adaptive systems for IoT and security improvements through OTA (Over The Air) updates.

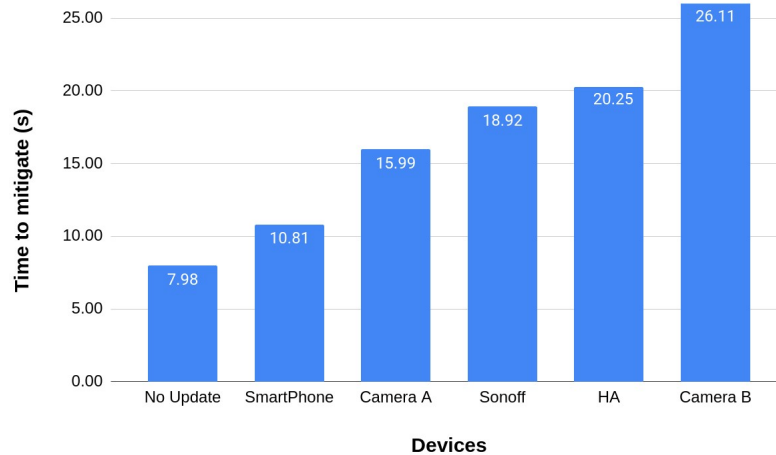


Figure 12: 5 Device update runtime

5.1 Self-adaptive systems for IoT

Hellaoui et al. [Hellaoui et al., 2016] presented a model for adaptive security based on the reliability of IoT devices. In this work, the proposed model is adopted to reduce the authentication overhead so that only the necessary packages are authenticated [Hellaoui et al., 2016]. Each node periodically calculates and decides locally to authenticate the received message, depending on the trust associated with the message's sender (neighbouring node). In addition, authentication upon receiving a message is applied dynamically.

Chen et al. [Chen et al., 2014] proposed a framework for Autonomic Cybersecurity Management (ACSM), utilising an autonomous model designed explicitly for IoT. ACSM can estimate, detect and respond to cyberattacks without human intervention.

Alhafidh [Alhafidh and Allen, 2016] discussed the importance of ensuring the security and privacy of user data in the design and specification of smart home systems. The authors proposed a security monitoring module for a series of typical scenarios based on the behaviour of one or several individuals. Thus, the system seeks to understand different users' behaviour as they interact with the devices and builds a user activity model within the smart home.

Palanca [Palanca et al., 2018] presented a goal-oriented self-adaptive architecture based on the DGOC (Distributed Goal-Oriented Computing) formal model for developing a smart home environment. According to the authors, users express their goals, and the framework is responsible for adjusting the environment, e.g., turning on a smart TV, dimming lighting, and closing windows.

El-Maliki [El-Maliki and Seigne, 2016] presented the SARM (Security Adaptation Reference Monitor) framework, which is an autonomous solution developed for IoT. Conceptually, SARM is divided into a functional unit (managed system) and a monitoring unit (manager system) connected by a loop based on Control Theory. The security parameters are adjusted at each iteration, considering the

risk of environmental threats and the system's performance.

Some works use MAPEK for the adaptation logic.

Utilising blockchain technology can implement security measures for self-adaptive systems designed for IoT. Sedgewick [Sedgewick and de Lemos, 2018] successfully integrated self-adaptation with blockchain to safeguard a network of IoT devices from malicious activities by employing a MAPE-K control loop.

Ribeiro [Ribeiro et al., 2016] designed a self-adaptive architectural pattern in IoT for modelling control loops. The authors adopted the decentralised approach, combining "master/slave" and "regional planning" patterns. In the monitoring phase of MAPE-K (on the slave), the MLPLW (Multi-layer Perceptron with Limited Weights) machine learning model was utilised.

To solve problems associated with security, flexibility, and scalability requirements, Singh [Singh and Lee, 2021] proposed a self-adaptive security model for multimedia services based on IoT. The basis for this self-adaptive model was the categorisation analysis of self-adaptive security systems using the components of the MAPE-K control loop. In this way, the model monitors and detects vulnerabilities and then reacts by determining changes.

Lee [Lee et al., 2019] proposed a framework based on the MAPE-K. This work uses a mathematical model of a finite state machine and a decision-making method based on game theory. The solution is based on adaptive strategy and uses Nash equilibrium to analyse the results of strategic interactions between different decision-makers.

Security Assurance Cases (SACs) are modelled using the GSN (Goal-Structured Notation) notation and reflect the control definitions and arguments that prove their compliance. Jahan [Jahan et al., 2020] presented a framework that uses a GSN model. In this work, the authors presented the interaction between two control loops, MAPE-K and MAPE-SAC (Monitor Analyzer Plan Executor Security Assurance Cases).

Mozzaquatro [Mozzaquatro et al., 2016] introduced a security framework based on ontology for decision-making in an IoT setting. They utilised a MAPE-K control loop and an IoT security reference ontology (IoTSec) to gather contextual information from the environment and feed it into the system.

The MDP (Markov Decision Process) can be used to formally describe an environment and serve as a basis for developing an IRS (Intrusion Response System). Iannucci [Iannucci and Abdelwahed, 2016] designed an autonomous IRS over a centralized MAPE-K control loop.

5.2 Security improvements through OTA updates

Including an update mechanism on IoT devices is a necessary security measure. This section presents works related to OTA. OTA updates a device through the network and allows a wireless device to update software faster and more securely than physically intervening with the device [Tiinside, 2021] [DornerWorks, 2020].

Chandra [Chandra et al., 2016] presents a solution that enables the OTA update of the firmware of IoT devices based on the Lightweight Mesh (LWMesh) network protocol. This protocol provides low-consumption communication between connected devices.

Lee [Lee and Lee, 2017] focuses on secure firmware updates, a security challenge for embedded devices in IoT environments. The authors propose a firmware

update scheme that uses blockchain to securely check the firmware version and validate and download the latest firmware to the devices.

Zandberg [Zandberg et al., 2019] adopts open-source standards and libraries that provide secure means for firmware updates of IoT devices. This work used the SUIT (Software Updates for Internet of Things) back-end architecture, an IETF (Internet Engineering Task Force) standard for authentication and integrity protection of IoT updates. A concern in this work is the maintenance of OTA updates during the lifecycle of the device.

Frisch [Frisch et al., 2017] implemented a system that promotes security in developing and publishing OTA updates. This implementation was applied using encryption for embedded devices based on ESP8266 microcontrollers. The MQTT (Message Queuing Telemetry Transport) protocol was used in the OTA update process. The proposed mechanism was integrated into the HA gateway (home assistant) to check for available updates.

Lin [Lin and Bergmann, 2016] proposed a gateway architecture for firmware updates of IoT devices to ensure a secure environment in smart homes. A gateway manages the update process locally and automatically schedules updates at convenient times.

5.3 Further Solutions

The search for improvements in the safety of commercial, industrial and residential buildings has been a recurring concern, e.g., the Barni project. The Barni project [Barni, 2023] integrates traffic normalization and improves event visualization in building automation environments to promote operator situational awareness. Thus, the Barni project aims to protect the network-level security of building automation systems that run the BACnet (Building Automation and Control Networks) protocol. BACnet is a standardized communication protocol used in building automation and control systems. Kaur et al. (2015) [Kaur et al., 2015] present actions on the network and application layers, showing how to prevent the exploitation of vulnerabilities in the BACnet network.

Iqbal [Iqbal et al., 2023] presented a security architecture for smart home systems based on SDN (Software Defined Network), where processing complexities are transferred from devices with limited resources to a centralised controller. The authors sought to overcome smart device resource and security limitations to promote network-level security based on light cryptography. With an authentication protocol, the initial access is deliberate in the presence of the SDN controller, which follows the subsequent authentications for service requests.

5.4 Summary

The analysis of related works takes the following considerations:

- Cyber attacks have grown daily, and a way adopted in the literature to mitigate vulnerabilities in IoT has been using autonomous systems to deal with the constant changes in the environment [Chen et al., 2014], [El-Maliki and Seigne, 2016].
- Among other security measures, blockchain has been used in self-adaptation to increase protection in IoT. For example, some initiatives use blockchain

to validate integrity in device firmware updates [Sedgewick and de Lemos, 2018], [Lee and Lee, 2017], [Chandra et al., 2016].

- Reliability is a central point for communication between the nodes of a network, marked by the level of trust each node builds. The change of security factors of the CIA triad (Confidentiality, Integrity, and Availability) can serve as triggers to self-adaptation action to protect smart homes [Hellaoui et al., 2016] [Singh and Lee, 2021] [Alhafidh and Allen, 2016].
- The interaction with the smart home utilising objectives with the use of formal methods is possible, and this usually happens without being noticed by the users [Mozzaquatro et al., 2016], [Palanca et al., 2018], [Lee et al., 2019].
- The association of "master/slave" approaches and regional planning builds an architectural pattern for developing control loops. Other options may arise with the association of different control loops, enabling the treatment of security and performance [Ribeiro et al., 2016], [Jahan et al., 2020], [Iannucci and Abdelwahed, 2016].
- Lifecycle issues of IoT devices and transfer of ownership have drawn attention in some works. In this case, gateways (e.g., home-assistant) have been used to monitor firmware versions. Another critical concern has been the firmware's development, validation, transportation, and the need to address any failures after installation. [Zandberg et al., 2019], [Frisch et al., 2017], [Lin and Bergmann, 2016].

Table 4 presents a comparative evaluation of existing works. Columns present criteria associated with self-adaptation and whether the work addresses OTA updates. Firstly, the type of adaptive management is identified: MAPE-K, Control Theory or Undefined. Next, the degree of decentralisation of the solution includes the following options: Centralised, Decentralised or Undefined. The fourth column shows the security aspect being focused on in the work. The last three columns correspond respectively to the condition of the work proposing an autonomous solution (with little or no human intervention), dynamic (acting at runtime), or addressing OTA updates. The ✓ in the last three columns refers to the positive condition for standalone, dynamic or OTA. Likewise, the marking X means the work has no respective condition. The following abbreviations are used in the table 4: TCL (Type of Control Loop), DoD (Degree of Decentralisation), AUT (Autonomous), DYN (Dynamic), OTA (Over The Air).

Through the analysis of the table and the description of the works, it is noticed that different self-adaptation approaches are applied in IoT environments. The adaptation logic can be distributed (decentralised) depending on the case. This problem is common when MAPE-K components, for example, Analyzer and Plan, need to use more computational resources externally since the scarcity of resources prevails in IoT. The central issues of each scenario dictate how the adaptation logic should be done, whether centralised or decentralised, distributing the functionality of MAPE-K [Krupitzer et al., 2015]. Except for works that do not aim at self-adaptation ([Chandra et al., 2016], [Lee and Lee, 2017], [Zandberg et al., 2019], [Frisch et al., 2017], [Lin and Bergmann, 2016]), in some related works it was not possible to identify that loop type of control is used

	TCL	DoD	SA	AUT	DYN	OTA
[Hellaoui et al., 2016]	Undefined	Undefined	Authentication	×	×	×
[Chen et al., 2014]	Control Theory	Centralized	Authentication	✓	✓	×
[Alhafidh and Allen, 2016]	Undefined	Centralized	Undefined	✓	✓	×
[Palanca et al., 2018]	Undefined	Decentralized	Authentication Privacy	✓	✓	×
[El-Maliki and Seigne, 2016]	Control Theory	Centralized	Authentication Integrity	✓	✓	×
[Sedgewick and de Lemos, 2018]	MAPE-K	Decentralized	Access control	✓	✓	×
[Ribeiro et al., 2016]	MAPE-K	Decentralized	Confidentiality	✓	✓	×
[Singh and Lee, 2021]	MAPE-K	Centralized	Confidentiality Integrity Availability	✓	✓	×
[Lee et al., 2019]	MAPE	Centralized	Authentication Authorization	✓	✓	×
[Jahan et al., 2020]	MAPE-K	Decentralized	Authorization	✓	✓	×
[Mozzaquatro et al., 2016]	MAPE-K	Decentralized	Authentication Authorization	✓	×	×
[Iannucci and Abdelwahed, 2016]	Control Theory MAPE-K	Centralized	Authentication Integrity	✓	✓	×
[Chandra et al., 2016]	Undefined	Undefined	Authentication	×	×	✓
[Lee and Lee, 2017]	Undefined	Undefined	Integrity	×	✓	✓
[Zandberg et al., 2019]	Undefined	Undefined	Authentication Integrity	×	✓	✓
[Frisch et al., 2017]	Undefined	Undefined	Integrity	×	✓	✓
[Lin and Bergmann, 2016]	Undefined	Undefined	Authentication Access control Confidentiality	×	×	✓
[Kaur et al., 2015]	Undefined	Undefined	Integrity Availability	×	×	×
[Iqbal et al., 2023]	Undefined	Undefined	Integrity	×	×	✓
UP-Home	MAPE-K	Centralized	Confidentiality Integrity Availability	✓	✓	✓

Table 4: Summary of related works

([Hellaoui et al., 2016], [Alhafidh and Allen, 2016], [Palanca et al., 2018]). However, they are presented as adaptive solutions.

Some works used the decentralized MAPE-K control loop [Jahan et al., 2020], [Sedgewick and de Lemos, 2018], [Mozzaquatro et al., 2016]), [Ribeiro et al., 2016]). Other works ([Singh and Lee, 2021], [Lee et al., 2019] e [Iannucci and Abdelwahed, 2016]) adopted centralized MAPE-K. MAPE-K is an adaptation reference model but not the only one. Another way to implement changes is with the use of Control Theory [Fileri et al., 2017], for example, what was proposed in the works [Chen et al., 2014], [El-Maliki and Seigne, 2016]. Two control loop models were associated in Iannucci and Abdelwahed (2016), joining Control Theory and MAPE-K [Iannucci and Abdelwahed, 2016].

Two papers used blockchain. The first implemented a self-adaptive system in a network of IoT devices to handle access control of devices on the network [Sedgewick and de Lemos, 2018]. In another work [Lee and Lee, 2017], the authors used blockchain to promote security in a firmware update system. Machine learning served to create an architectural pattern, and in the work of Singh and Lee (2021) [Ribeiro et al., 2016], it promoted improvement in the security goals

of the CIA triad. Among the works listed in this section, five works directly deal with self-adaptation approaches aimed at smart homes [Chandra et al., 2016], ([Sedgewick and de Lemos, 2018], [Lee et al., 2019], [Lin and Bergmann, 2016]), citefrisch2017over, the others generically focus on IoT, without dwelling on a particular IoT application domain.

Machine learning served to create an architectural pattern [Ribeiro et al., 2016], and in the work of Singh and Lee (2021) promoted security in terms of security goals of the CIA triad [Singh and Lee, 2021]. Among the works listed in this section, five papers deal directly with self-adaptation approaches aimed at smart homes ([Sedgewick and de Lemos, 2018], [Lee et al., 2019], [Chandra et al., 2016], [Lin and Bergmann, 2016], [Frisch et al., 2017]), the others generically focus on IoT, without dwelling on a particular IoT application domain.

The interaction between the system that manages the changes and the users who express their objectives translates into a set of actions that are transparent to the user in the work of Palanca et al. (2018) [Palanca et al., 2018]. Jahan et al. (2020) addressed Security Assurance Cases (SACs) modelled with the GSN (Goal-Structured Notation) notation for compliance issues, which are used to guide the adaptation process. Monitoring and analysis of both MAPE-K loops operate in parallel; planning and execution are coordinated [Jahan et al., 2020].

Using the traffic normalizer, TCP/IP networks can be more secure by removing suspicious network traffic [Barni, 2023]. This strategy can help eliminate potentially harmful elements, including preventing the exploitation of vulnerabilities [Kaur et al., 2015]. It is important to propose secure and lightweight authentication protocols for smart home environments such as SDN [Iqbal et al., 2023]. However, security threats such as impersonation, session key disclosure, and Man-in-the-Middle (MITM) attacks may occur.

Security solutions such as IDS and IRS have been adopted to detect or respond to attacks from potential intruders ([Ribeiro et al., 2016], [Chen et al., 2014], [Alhafidh and Allen, 2016], [Zandberg et al., 2019], [Palanca et al., 2018]), [Frisch et al., 2017]. The preservation of information security (integrity, authentication, authorisation) is constantly addressed in most related works. Security measures have been applied through self-adaptation (Section 5.1) and OTA updates (Section 5.2). Although some works seek to improve security or apply to the context of self-adaptive systems, none apply to promoting updates of parameters or components autonomously and dynamically, considering security constraints. Given this, UP-Home seeks to operate autonomously (with little or no human intervention) and dynamically (at runtime) to ensure that smart home devices are safely updated.

In general, there is a need to build self-adaptive solutions that address security issues related to smart homes, such as updating OTA. Furthermore, it should be noted that using a self-adaptive reference model such as MAPE-K can help implement security measures in a scenario of increasing vulnerabilities, as in IoT. Finally, there is a need for proposals that seek to promote security within the framework of software stack updates present in IoT devices. Faced with this, UP-Home seeks to operate autonomously (with little or no human intervention) and dynamically (at runtime) to ensure that smart home devices are securely updated. UP-Home seeks to apply OTA updates primarily to ensure integrity, playing a significant role in maintaining the confidentiality and availability of smart home devices.

6 Conclusion and future works

Considering the growth of security-related problems, this paper presented UP-Home, a self-adaptive solution that aims to improve the security of smart homes. The proposed solution strives to ensure that the smart home devices meet the established security requirements. The UP-Home solution uses autonomic computing concepts and acts on a reference model of self-adaptive systems (MAPE-K control loop) to apply security adaptations. UP-Home defines monitoring strategies and a mechanism for analyzing security parameters to identify and execute coordinated actions on devices violating security goals.

The main contribution of this paper is the proposition of a solution that promotes security autonomously and dynamically by managing security parameters. UP-Home is centred on identifying smart home security parameters and goals and acting to achieve them continuously. In addition to this contribution, there are other related contributions. One of these contributions is the implementation of security monitoring strategies, considering security parameters that can be implemented in smart home environments. Another contribution is the definition of an analysis strategy that precisely defines the occurrence of security breaches in smart home devices through compositional and parametric adaptation.

This paper has achieved the following results:

- Security Parameters. Definition of security parameters considered when monitoring and making decisions about necessary security updates of smart homes.
- Conceptual Model. Definition of the conceptual model for the proposed solution. As presented in Figure 1, the model consists of a set of conceptual elements present in adaptive systems and includes goals, the adaptation logic, and the elements that need to be monitored.
- Architecture. The definition of the UP-Home architecture (see Figure 2) with the adaptation logic elements and all the high-level components of the solution.
- Prototype. The UP-Home architecture was fully implemented using the HA gateways (home-assistant), Mikrotik router board, and the OpenVas scanner.

Within the context of this paper, several opportunities for continuity are envisaged. Firstly, evaluate the consumption of computational resources using different workloads. Some examples might be monitoring bandwidths, protocols, and network technologies. Secondly, machine learning techniques should be adopted to detect potential security attacks so that UP-Home adaptation can become proactive, i.e., act before security issues arise. Finally, additional vulnerabilities can also be included in the monitoring mechanism used by UP-Home.

References

[Aksu et al., 2019] Aksu, M. U., Altuncu, E., and Bicakci, K. (2019). A first look at the usability of openvas vulnerability scanner. In Workshop on usable security (USEC).

- [Alhafidh and Allen, 2016] Alhafidh, B. M., and Allen, W. (2016). Design and simulation of a smart home managed by an intelligent self-adaptive system. *International Journal of Engineering Research and Applications*, 6(8), pages. 64–90.
- [Bacudio et al., 2011] Bacudio, A. G., Yuan, X., Chu, B.-T. B., and Jones, M. (2011). An overview of penetration testing. *International Journal of Network Security & Its Applications*, 3(6), 19. Academy & Industry Research Collaboration Center (AIRCC).
- [Barni, 2023] Barni (2023). Barni: Building automation reliable network infrastructure. <https://net.cs.uni-bonn.de/wg/itsec/projects/completed/barni/>.
- [Bauwens et al., 2020] Bauwens, J., Ruckebusch, P., Giannoulis, S., Moerman, I., and De Poorter, E. (2020). Over-the-air software updates in the internet of things: An overview of key principles. *IEEE Communications Magazine*, 58(2), pages. 35–41. IEEE.
- [Chandra et al., 2016] Chandra, H., Anggadajaja, E., Wijaya, P. S., and Gunawan, E. (2016). Internet of things: Over-the-air (OTA) firmware update in lightweight mesh network protocol for smart urban development.. In *2016 22nd Asia-Pacific Conference on Communications (APCC)* (pages. 115–118). IEEE.
- [Chen et al., 2014] Chen, Q., Abdelwahed, S., and Erradi, A. (2014). A model-based validated autonomic approach to self-protect computing systems. *IEEE Internet of things Journal*, 1(5), pages. 446–460.
- [Darby, 2018] Darby, S. J. (2018). Smart technology in the home: time for more clarity. *Building Research & Information*, 46(1), pages. 140–147. <https://doi.org/10.1080/09613218.2017.1301707>
- [DornerWorks, 2020] DornerWorks (2020). The importance of over-the-air (OTA) updates in IoT. <https://www.iotforall.com/over-the-air-ota-updates-reduce-risk-win-customers>.
- [El-Maliki and Seigne, 2016] El-Maliki, T., and Seigne, J.-M. (2016). Efficient security adaptation framework for internet of things. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)* (pages. 206–211). IEEE.
- [Ferreira et al., 2023] Ferreira, L., Oliveira, T., and Neves, C. (2023). Consumer’s intention to use and recommend smart home technologies: The role of environmental awareness. *Energy*, 263, 125814. Elsevier.
- [Filieri et al., 2017] Filieri, A., Maggio, M., Angelopoulos, K., D’ippolito, N., Gerostathopoulos, I., Hempel, A. B., (2017). Control strategies for self-adaptive software systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 11(4), pages. 1–31.
- [Frisch et al., 2017] Frisch, D., Reißmann, S., and Pape, C. (2017). An over the air update mechanism for esp8266 microcontrollers. In *Proceedings of the ICSNC, the Twelfth International Conference on Systems and Networks Communications*, Athens, Greece (pages. 8–12).
- [Gu et al., 2020] Gu, T., Fang, Z., Abhishek, A., Fu, H., Hu, P., and Mohapatra, P. (2020). Iotgaze: IoT security enforcement via wireless context analysis. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications* (pages. 884–893). IEEE.
- [Gubbi et al., 2013] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): a vision, architectural elements, and future directions. *Future generation computer systems*, 29(7), pages. 1645–1660. Elsevier.
- [Hassija et al., 2019] Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., and Sikdar, B. (2019). A survey on IoT security: application areas, security threats, and solution architectures. *IEEE Access*, 7, pages. 82721–82743. IEEE.

- [Hellaoui et al., 2016] Hellaoui, H., Bouabdallah, A., and Koudil, M. (2016). Tas-IoT: trust-based adaptive security in the IoT. In 2016 IEEE 41st Conference on Local Computer Networks (LCN) (pages. 599–602). IEEE.
- [Hussin et al., 2023] Hussin, S. F., Abdollah, M. F., and Ahmad, I. B. (2023). Acceptance of IoT technology for smart homes: a systematic literature review. In International Conference on Information Systems and Intelligent Applications (pages. 187–202). Springer.
- [Iannucci and Abdelwahed, 2016] Iannucci, S., and Abdelwahed, S. (2016). A probabilistic approach to autonomic security management. In 2016 IEEE International Conference on Autonomic Computing (ICAC) (pages. 157–166). IEEE.
- [IoTSEC, 2018] IoTSEC. (2018). IoT security architecture and policy for the enterprise-a hub based approach release 1. Normas de Segurança da Informação/IoT Security Foundation. 37. <https://www.iotsecurityfoundation.org/wp-content/uploads/2018/11/IoT-Security-Architecture-and-Policy-for-the-Enterprise-a-Hub-Based-Approach.pdf>.
- [Iqbal et al., 2023] Iqbal, W., Abbas, H., Deng, P., Wan, J., Rauf, B., Abbas, Y., and Rashid, I. (2023). Alam: Anonymous lightweight authentication mechanism for sdn enabled smart homes. *Journal of Network and Computer Applications*, 103672. Elsevier.
- [Jahan et al., 2020] Jahan, S., Riley, I., Walter, C., Gamble, R. F., Pasco, M., McKinley, P. K., and Cheng, B. H. C. (2020). Mape-k/mape-sac: An interaction framework for adaptive systems with security assurance cases. *Future Generation Computer Systems*, 109, 197–209. Elsevier.
- [Kaur et al., 2015] Kaur, J., Tonejc, J., Wendzel, S., and Meier, M. (2015). Securing bacnet’s pitfalls. In *ICT Systems Security and Privacy Protection: 30th IFIP TC 11 International Conference, SEC 2015, Hamburg, Germany, May 26-28, 2015, Proceedings 30* (pages. 616–629). Springer.
- [Kephart and Chess, 2003] Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1), 41–50. IEEE.
- [Komninos et al., 2014] Komninos, N., Philippou, E., and Pitsillides, A. (2014). Survey in smart grid and smart home security: Issues, challenges and countermeasures. *IEEE Communications Surveys and Tutorials*, 16(4), pages. 1933–1954. IEEE.
- [Krupitzer et al., 2015] Krupitzer, C., Roth, F. M., VanSyckel, S., Schiele, G., and Becker, C. (2015). A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, 17, 184–206. Elsevier.
- [Lee and Lee, 2017] Lee, B., and Lee, J.-H. (2017). Blockchain-based secure firmware update for embedded devices in an internet of things environment. *The Journal of Supercomputing*, 73(3), pages. 1152–1167.
- [Lee et al., 2019] Lee, E., Seo, Y.-D., and Kim, Y.-G. (2019). Self-adaptive framework based on mape loop for Internet of Things. *Sensors*, 19(13), 2996.
- [Lehman, 1996] Lehman, M. M. (1996). Laws of software evolution revisited. In *European Workshop on Software Process Technology* (pages. 108–124). Springer.
- [Lin and Bergmann, 2016] Lin, H. and Bergmann, N. W. (2016). IoT privacy and security challenges for smart home environments. *Information*, 7(3), 44. Multidisciplinary Digital Publishing Institute.
- [Madakam et al., 2015] Madakam, S., Lake, V., and others. (2015). Internet of Things (IoT): A literature review. *Journal of Computer and Communications*, 3(05), 164. Scientific Research Publishing.

- [Marikyan et al., 2019] Marikyan, D., Papagiannidis, S., and Alamanos, E. (2019). A systematic review of the smart home literature: A user perspective. *Technological Forecasting and Social Change*, 138, pages.139–154. Elsevier.
- [McKinley et al., 2004] McKinley, P. K., Sadjadi, S. M., Kasten, E. P., and Cheng, B. H. (2004). Composing adaptive software. *Computer*, 37(7), 56–64. IEEE.
- [Mozzaquatro et al., 2016] Mozzaquatro, B. A., Jardim-Goncalves, R., Melo, R., and Agostinho, C. (2016). The application of security adaptive framework for sensor in industrial systems. In *2016 IEEE Sensors Applications Symposium (SAS)*, (pages. 1–6). IEEE.
- [Nieles et al., 2017] Nieles, M., Dempsey, K., and Pillitteri, V. Y. (2017). An introduction to information security. NIST special publication, 800, 12.
- [Palanca et al., 2018] Palanca, J., Del Val, E., Garcia-Fornes, A., Billhardt, H., Corchado, J. M., and Julián, V. (2018). Designing a goal-oriented smart-home environment. *Information Systems Frontiers*, 20(1), pages. 125–142. Springer.
- [Ribeiro et al., 2016] Ribeiro, A. de R. L., de Almeida, F. M., Moreno, E. D., and Montesco, C. A. E. (2016). A management architectural pattern for adaptation system in internet of things. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)* (pages. 576–581). IEEE.
- [Salehie and Tahvildari, 2009] Salehie, M. and Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)*, 4(2), 1–42. ACM New York, NY, USA.
- [Schiefer, 2015] Schiefer, M. (2015). Smart home definition and security threats. In *2015 ninth international conference on IT security incident management & IT forensics* (pages. 114–118). IEEE.
- [Sedgewick and de Lemos, 2018] Sedgewick, P. E., and de Lemos, R. (2018). Self-adaptation made easy with blockchains. In *2018 IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages. 192–193. IEEE.
- [Shah and Mehtre, 2015] Shah, S. and Mehtre, B. M. (2015). An overview of vulnerability assessment and penetration testing techniques. *Journal of Computer Virology and Hacking Techniques*, 11(1), 27–49. Springer.
- [Singh and Lee, 2021] Singh, I., and Lee, S.-W. (2021). Self-adaptive and secure mechanism for IoT based multimedia services: a survey. *Multimedia Tools and Applications*, pages. 1–36. Springer.
- [Tan and Wang, 2010] Tan, L. and Wang, N. (2010). Future internet: The internet of things. In *2010 3rd international conference on advanced computer theory and engineering (ICACTE)* (Vol. 5, pages. V5–376). IEEE.
- [Tiinside, 2021] Tiinside. (2021). Segurança em IoT é um desafio permanente para o mercado. <https://tiinside.com.br/24/02/2021/seguranca-em-iot-e-um-desafio-permanente-para-mercado/>.
- [Weyns, 2017] Weyns, D. (2017). Software engineering of self-adaptive systems: an organised tour and future challenges. In *Handbook of Software Engineering* (pages. 1–16). Springer.
- [Zandberg et al., 2019] Zandberg, K., Schleiser, K., Acosta, F., Tschofenig, H., and Baccelli, E. (2019). Secure firmware updates for constrained IoT devices using open standards: A reality check. *IEEE Access*, 7, 71907–71920. IEEE.

[Zhang et al., 2014] Zhang, Z.-K., Cho, M. C. Y., Wang, C.-W., Hsu, C.-W., Chen, C. K., and Shieh, S. (2014). Iot security: ongoing challenges and research opportunities. In 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications (pages. 230–234). IEEE.

[Zhou et al., 2019] Zhou, W., Jia, Y., Yao, Y., Zhu, L., Guan, L., Mao, Y., Liu, P., and Zhang, Y. (2019). Discovering and understanding the security hazards in the interactions between IoT devices, mobile apps, and clouds on smart home platforms. In 28th USENIX Security Symposium (USENIX Security 19) (pages. 1133–1150).

[Zikria et al., 2018] Zikria, Y. B., Yu, H., Afzal, M. K., Rehmani, M. H., and Hahm, O. (2018). Internet of things (IoT): Operating system, applications and protocols design, and validation techniques. Elsevier.