Automated standard based security assessment for IoT

André Nuno de Pinho Tavares Gurgo e Cirne
Mestrado em Segurança Informática
Departamento de Ciência de Computadores
2020

Orientador
Luís Filipe Coelho Antunes
Professor Catedrático
Faculdade de Ciências da Universidade do Porto

Coorientador
Patrícia Raquel Vieira Sousa
Professora Assistente Convidada
Faculdade de Ciências da Universidade do Porto

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas
pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ______/______/________

UNIVERSIDADE DO PORTO

MASTERS THESIS

Automated standard based security assessment for IoT

Author:
André CIRNE

Supervisor:
Luís ANTUNES

Co-supervisor:
Patrícia SOUSA

A thesis submitted in fulfilment of the requirements for the degree of MSc. Information Security

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

January 4, 2021

Acknowledgements

I want to thank Professor Luís Antunes and Professor Patrícia Sousa for this opportunity, all the support given and guidance during this work.

For the ideas given and the never-ending interest in this thesis, I want to thank to Professor João Resende.

I want to thank my friends and colleagues on C3P, Ana, Inês, Luís and Simão, for the friendly environment that was created, and many suggestions that were given for my work.

To my family for their support at all times, and the patience in the bad ones. Also, a special thanks to my sister, Joana, for their motivational speeches when the motivation was low.

I also want to thank all my friends that support me during this step of my life.

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

MSc. Information Security

Automated standard based security assessment for IoT

by André CIRNE

The Internet of Things (IoT) is changing the way we interact with the world around us. The number of IoT devices is increasing exponentially, and, as a result, the information exchanged on the network also increases. Much of that information often have personal and confidential content. This new paradigm brings security challenges, mainly due to the dynamic and heterogeneous nature of IoT-enabled environments. There are different types of attacks on different IoT layers, such as unauthorized access, device cloning, Sybil attack, sinkhole attack, denial of service attack, code injection, and man in the middle attacks. Traditional security proposals are not feasible in these scenarios, therefore, it is necessary to promote the adoption of security measures.

IoT certifications have emerged in recent years as they play a key role in adopting uniform security policies on IoT devices and systems. Meanwhile, the European Union approved the Cybersecurity Act to unify and regulate security certifications in the member states. The Cyber Security Act also introduced the European Cyber Security Certification Framework (ECSCF), which establishes a framework to unify the structure of cybersecurity certifications in member states.

Cybersecurity certifications require the use of methods to test compliance with defined security requirements. Test methods have limitations in the IoT environment because they are complex, dynamic, change quickly, have limited resources, and traditional test methods cannot keep up with these changes, delaying the product's release date and increasing the certification's cost. One solution to this problem is to create systems capable of auditing a device automatically and continuously.

Our work collects the requirements that different IoT environments and application scenarios impose on certifications. We analyze the current state of development of automatic IoT security tests, and, finally, we propose a new evaluation system that automatically and continuously evaluates IoT environments. At the same time, our proposal meets the requirements to be eligible as an assessment method for certification following the ECSCF.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

Mestrado em Segurança Informática

Análise de segurança automática para dispositivos IoT

por André CIRNE

Internet of Things (IoT) está a mudar a forma como interagimos com o mundo à nossa volta. O número de dispositivos IoT estão a aumentar exponencialmente e, com isso, a informação trocada na rede também aumenta. Esta informação contém, muitas vezes, conteúdo pessoal e confidencial. Este novo paradigma traz desafios de segurança, principalmente devido à natureza dinâmica e heterogênea dos ambientes IoT. Existem diferentes tipos de ataque em diferentes *layers* do IoT, nomeadamente acessos não autorizados, ataques de *sybil* ou *man-in-the-middle*. As soluções de segurança tradicionais não são viáveis nestes cenários e é necessário adoptar novas medidas de segurança para resolver estes problemas.

As certificações para IoT têm surgido ao longo dos últimos anos, uma vez que desempenham um papel fundamental na adoção de políticas de segurança uniformes em dispositivos e sistemas IoT. A própria União Europeia aprovou a Lei de Segurança Cibernética para unificar e regular as certificações de segurança nos estados membros. A Lei da Segurança Cibernética, da Comissão Europeia também introduziu a ECSCF, que estabelece uma estrutura para unificar a estrutura de certificações de segurança cibernética nos estados membros.

As certificações de segurança exigem o uso de métodos para testar a conformidade com os requisitos de segurança definidos. Os métodos de teste têm limitações no ambiente IoT porque são complexos, dinâmicos, mudam rapidamente, têm recursos limitados e tradicionalmente não conseguem acompanhar essas mudanças, atrasando a data de lançamento do produto e aumentando o custo da certificação. Uma solução para esse problema é criar sistemas capazes de auditar um dispositivo de forma automática e contínua.

O nosso trabalho faz uma recolha dos vários requisitos impostos a ambientes IoT, certificações, e cenários de aplicação. Analisámos o estado atual de desenvolvimento de testes automáticos de segurança de IoT e, por fim, propomos uma nova metodologia que avalia de forma automática e contínua os ambientes de IoT. Ao mesmo tempo, a nossa proposta tem de cumprir os requisitos para ser elegível como um método de avaliação para certificação, de acordo com o ECSCF.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Internet of Things (IoT) is a buzzword of the 21st century, and there is no consensus on its definition. Depending on the services provided, goals, and architecture, the definition differs [1]. This generalized heterogeneity is one of the characteristics of IoT. Currently, there is an effort to overcome this situation with standardization [2]. There are different standards from a variety of organizations, such as the Institute of Electrical and Electronics Engineers (IEEE) [3], National Institute of Standards and Technology (NIST) [4], European Telecommunications Standards Institute (ETSI) [5] to define what is IoT.

Nevertheless, IoT is often considered the concept of interconnecting objects that gather information from the environment and interact with the physical world and take advantage of an Internet connection to reach a common goal [6].

Nowadays, the number of IoT devices is continuously increasing. These systems are taking responsibilities in different domains and becoming increasingly essential for their operation. As a result of the intense adoption of IoT devices, new security matters arise. The demand for more inexpensive systems produces heterogeneity between devices and a lack of standards [7]. Along with this, the typical IoT architecture contributes to constraints for developing a first-class security and privacy system.

The architecture of an IoT system is composed of sensors, aggregators, communication channels, external utilities, and decision triggers [4]. These devices usually have limited resources, so it is impossible to apply standard security measures to them in the same way as a full-size computer [8].

IoT systems have dynamic and temporary communication channels between devices, leading to wireless communications preference [9]. This option introduces new attacks, such as jamming and tampering information [8]. Besides that, due to the choice of protocols only regarding power consumption, bandwidth requirements, and interoperability [10], these protocols usually do not offer great security mechanisms, resulting in more serious threats [11].

Asides from the architecture of these systems, their context also influences its security. The environment where the system operates, the costs that a security failure or privacy breach might induce, the physical location of each system's element, and its topology at a specific instant of time defines the system context [12]. Depending on these variables, the importance of a system and the likelihood of an attack can be assessed, and with this, the necessary security requirements can be defined [13].

With the emergence of new attacks and vulnerabilities, automated monitoring, testing, and mitigation tools are essential to address the risks arising from such threats or configuration failures. Device manufacturers must be increasingly involved in the promotion of automated solutions that guarantee each device's intended operation and the management and secure deployment of IoT devices [14]. Also, not automated solutions results in higher costs and delays in the production time [15]. By itself, most manufacturers are not creating solutions with security and privacy by design, leading to a lack of security measures on the IoT devices. The solutions for this issue can be government grants, awards, or the enforcement through laws and regulations [16].

1.1 Motivation

The certification of IoT systems is an urgent need, as it is the most effective solution to improve the security mechanisms of this type of system [17]. There are already some certifications on the market, and the academic community is actively working on new ones. Additionally, the creation of new certifications is being encouraged and regulated by different countries and international organizations [18].

To follow this trend and as a way to standardize new certifications that may arise, the European Union (EU) established the European Cyber Security Certification Framework (ECSCF) that defines a structure and methodology to be used by certifications in all member states. The Cybersecurity Act, a legislation that aims to strengthen European capabilities in cybersecurity, reinforces this scheme [19].

Existing certification schemes are described by the industry as not being agile or scalable, too expensive, delaying the release date of products, or requiring that everything is perfectly documented to a level that does not add anything further to the security [20]. Moreover, assessment methods used in these certification processes might affect its effectiveness. Classical security assessments are performed periodically, and IoT systems change quickly, so the likelihood of a new device emerging between periodic assessments will be high. Thus, the effectiveness of the assessment and, consequently, the certification might be compromised [12].

Several authors point out one way to mitigate these problems. *Jeffrey Voas* [21] states in his work that one possibility is to constantly audit a device with an auditing system that, ideally, could operate independently of any IoT vendor. *Jason Nurse et al.* [12] also reinforces this statement, saying that it is necessary to create automated and continuous risk assessment approaches for inter-dependent systems.

New emergent IoT certifications should depend on automatic assessment techniques to be able to fulfill the necessities of IoT [12]. Therefore, to support this trend, it is necessary to research existent automatic assessment techniques and methodologies, to identify their limitations and develop new solutions.

Beyond that, certifications aiming the European market should respect ECSCF guidelines. Thus, assessment methodologies need to facilitate the integration with these guidelines.

1.2 Proposed Solution

The work detailed in this thesis aims to provide ways to automate a compliance evaluation of a security standard in an IoT environment, at the same time, we fulfill the requirements imposed by IoT to security assessments. We also expect to identify which characteristics are more difficult to automate in a standard, which would be beneficial for developing new standards focused on IoT. The development of a tool capable of this would be useful for certification scheme owners as a new evaluation method adapted to different security requirements or directly by the system administrator to indicate the level of compliance of an environment regarding a specific standard.

Furthermore, we also intend to fulfill the European Union Agency for Cybersecurity (ENISA) guidelines to ease the integration of our solution in a cybersecurity certification.

Therefore, our work will focus on combining existing open-source technologies to perform testing and continuous monitoring to an IoT environment in an automatic way that can keep up with the constant changes in the environment.

We called our solution Automatic External and Continuous Security Assessment (AECSA).

1.2.1 Objectives

The main goals that we pretend to reach with our proposal are:

- **Goal 1** - Creation of a test procedure that meets the ENISA best practices to be eligible as an evaluation method for a technical specification of security requirements.
- **Goal 2** - An evaluation methodology that targets an agnostic vendor IoT environment with a specific use case.
- **Goal 3** - Provide an automatic and continuous assessment method.

1.2.2 Features

The main features of our proposal are as follows:

- **Modular**: The platform can change the tests being executed and add new ones with minimal effort. This feature allows our solution to be adapted to different standards.
- **Customizable**: Users can customize the evaluation policy for a specific device.
- **Vendor agnostic**: The platform can evaluate a device without requiring a privilege access to it.
- **Open Source**: Do not depend on closed-source solutions.
- **Device detection**: The platform can automatically recognize new devices on the network.
- **Security score**: The platform incorporates a score associated with each device to express the evaluation results better.
- **Automatic assessment**: The tests are performed with minimal human interaction.
- **Continuous assessment**: The evaluation results are updated over time and according to the changes of the IoT environment.

1.3 Contributions

During this work, a survey was submitted to the journal Computers & Security from Elsevier. At the time of this publication, we are waiting for the submission result.

1.4 Outline

This thesis is organized in the following chapters: Chapter 2 describes the current development state of cybersecurity certifications in the European Union (EU) and the requirements that IoT impose to these certifications. In the Chapter 3 explores the current work on security assessments for IoT and analyzes existent technical security requirements specifications. Chapter 4 details the system design of our proposal and Chapter 5 presents the implementation details and approaches used on our system. In the Chapter 6, we evaluate a virtualized IoT environment with our system. In the Chapter 7, we lay down some ideas for future work and we mention some final remarks.

Chapter 2

Background

With the Cybersecurity Act, the EU intends to establish a European cybersecurity certification scheme that defines a structure and methodology used in all member states. This legislation does not limit its application to a specific domain but aims all types of cybersecurity certifications.

Our proposal aims to create a new test procedure that meets the requirements of the Cybersecurity Act. Besides that, our solution must also fulfill the specific requirements for IoT. In this section, we will describe what a security certification scheme is, according to the main guidelines of the European bodies and how it should be structured. Next, we will detail the requirements for a security certification focused on IoT and finally, evaluate existing IoT security certifications regarding the EU and IoT requirements. We will also analyze the current development state of this field.

This background information will be important to understand our decisions in the design process.

2.1 Security certifications

A security certification is an assessment to determine if specific controls are correctly implemented, operating as intended, and producing the desired outcome to meet the security requirements [22]. Moreover, certifications should enforce the necessary controls. So that security risks to which the target is subject are properly mitigated.

These certifications are designed by the scheme owner, who should keep the certification updated according to the industry's needs, and are carried out by Conformity Assessment Body (CAB)s, who are expected to conduct applicable conformity assessments and generate an evaluation report to grant the certification.

2.1.1 Certification characteristics

In any security certification, there are three fundamental characteristics: a context, a target, and an audience (Fig. 2.1). These attributes will define in which cases a given certification can be applied.

This set of attributes was inspired by the ECSCF, the ENISA classification system for security certifications [23] and the characteristics of the EUROSMART certification [24].

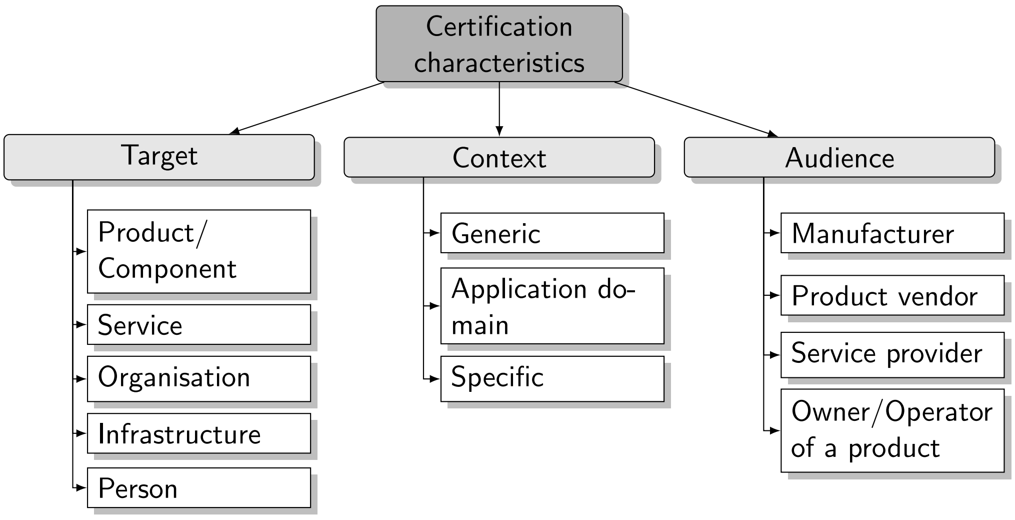


FIGURE 2.1: Certification characteristics

The certification target is the aim of the evaluation. This could be a unique product or component (for example, a smart bulb), a service (for instance, an assessment to a complete system that includes a smart bulbs, an IoT gateways, and Web API) and an organization or its infrastructure [25].

The context is the type of environment used during the certification procedure, which can vary from a general use case, where only the generic characteristics of that environment are known, or a specific application scenario, such as smart cities. The certification effectiveness will vary depending on the type of context, the scope assessed, the threats considered, and the assumptions made. A more precise context definition will result in a more precise listing of security requirements, resulting in a more secure device [12].

These two characteristics will affect the target audience of the certification. The audience of a certification could be divided into two groups: the sponsors and consumers. The sponsor is the organization financing the certification of the target, and the certification consumer is the entity for which the certification intends to prove that it complies with the security requirements. The main actors for these two groups are the manufacturer, product vendor, service provider, and device owner.

This wide range of characteristics causes a great diversity among certifications. This heterogeneous landscape is the one that the Cybersecurity Act wants to regulate and unify.

2.2 European guidelines for certification schemes

With the Cybersecurity Act, the EU intends to create an ECSCF that defines a structure and methodology to harmonize cybersecurity evaluations across the EU.

To achieve this goal, the ENISA and European Cyber Security Organization (ECSO) have published several recommendations. ECSO created the ECSCF [20], which is a meta-scheme for standardization, certification, labeling and supply chain management. Thus, this is not a certification but a structure that new certification schemes should follow.

The creation of this framework focus on the challenges stated by industry members of ECSO. These members realized that existing schemes assess the security of products, services, and systems with excellent results, but are not agile and require a lengthy certification process.

ECSCF meta-scheme allows the combination of different certifications to take advantage of its benefits. This framework is not intended to meet specific domain requirements, but rather to support creating a certification framework independent of the application scenario.

To support the ECSCF, ENISA created documents to clarify the way the Cybersecurity Act can be implemented [23], and a list of existing certifications grouped by application domain [26].

Every three years, the European Commission defines a work program with the priorities for future European cybersecurity certification schemes.

ENISA, according to these orientations, must prepare candidate schemes or review existent schemes to answer the conditions of the European Commission. These schemes must follow the ECSCF structure [27].

In the following subsections, we will summarize the significant guidelines that a certification must comply with.

2.2.1 Scheme owner responsibility

The owner of a cybersecurity certification is responsible for keeping the certification updated according to the industry's needs. To do this, it must maintain a working relationship with relevant stakeholders [23].

Besides, the owner must set up a structure for the operation and management of the scheme, including mechanisms to ensure transparency and trust, such as ways to deal with vulnerabilities discovered after the certification was issued.

Trust in certification must also go through the CABs involved in the process; everyone involved in the process must have the technical skills and competencies required to certify a target [27].

2.2.2 Core components

Article 54 of the Cybersecurity Act [27] lists components that candidates for EU cybersecurity certification must-have. From this list, we highlight three central elements that we consider mandatory for a certification to meet the organization proposed by the ECSCF.

Technical Specification of Security Requirements
The document contains a collection of requirements that describes the desired security features that must be implemented and covered by the certification scheme.

Assessment Methodology
The document precisely defines the validation procedures to assess the target against the technical security requirements, including documentation review, test procedures, and the level of automation that can be taken. It should also indicate the expected results of the assessment, such as a report or score.

Conformity Assessment Specification
The document describes all the policies and processes that govern the certification scheme,

covering the certification scheme's scope, the surveillance of certified products, and laboratories responsible for the evaluation.

Without these components, it is impossible to meet the structure proposed by ECSCF. The Cyber Security Act also states that these elements should be reused based on existing standards when applicable and whenever possible.

2.2.3 ENISA qualification system

The ENISA, based on the Cybersecurity Act's principles, proposes a qualification system to support the owner of the certification scheme in choosing the right components for his needs and to identify the assurance level of existent security certifications. This system introduces criteria to evaluate each one of the core elements (Subsection 2.2.2) [23].

The technical specification of security requirements is evaluated according to the accuracy defining the security requirements, namely if it aims a generic category of devices or a precise type of device.

The classification system defines four parameters to classify an assessment methodology: the assessment style, evaluation formalism, produce comparable results and level of objectivity in the evaluation.

The assessment style distinguishes whether the evaluation is done by a third-party or self-assessment. The evaluation formalism verifies if the assessment methodology uses formal and structured language to specify the technical requirements and evaluation procedures. Beyond that, the qualification system assesses whether the assessment methodology produces comparable results and how much the evaluation depends on the evaluation team's discretional assessment (objectivity level).

The conformity assessment specification is evaluated according to the measures installed to monitor the audited devices, the laboratories that issue the certification, and the capacity to update the security requirements according to the appearance of new threats.

2.2.4 Indicators of Confidence and Security

There are some schemes, like Common Criteria [28], which have an Evaluation Assurance Level, a numerical assessment that describes the depth and rigor of an assessment.

The assurance level dictates the level of access and the amount of information required to evaluate a target. Generally, the assessment is limited by the publicly available information and the final product (black-box approach) on the low assurance level. Whereas at higher levels, the evaluator has access to privileged information about the product, such as source code or design information (white-box approach).

ECSCF borrows the assurance level notion from the Common Criteria with some adaptations. The ECSCF states that with higher assurance levels, the Common Criteria solution does not guarantee that the certification context is extensive enough to identify the target's risks properly. Thus, it proposes an additional condition in the assurance level. It requires that with a higher assurance level, the context is more specific. The evaluation will have more available information about the target, mitigating the issue mentioned before.

The ECSCF introduces a structure for assurance levels divided into two groups: the base group, in which a black-box type evaluation is carried out, and the advanced group in which the target underwent a deep assessment where the context of the product is included.

The base-level includes two levels of assurance, the Entry Level (Level E), which is based on self-assessment, and the Basic level (Level D), which is the same as Entry, but a third party makes the assessment.

The advanced group holds three assurance levels: Enhanced basic (Level C), where the assessment goes beyond the black-box assessment, and the target context requirements are included in the evaluation. This level requires that the target withstands basic attacks; Moderate (Level B), where is given to the evaluator some privileges information, but not full access (grey-box approach) and it requires the resistance against medium complexity attacks; High (Level A), a white-box assessment where the context of the product is fully understood and the products needs to resist to high complexity attacks.

In the moment of designing a security certification, the scheme owner must adapt this structure according to its use case and choose an evaluation methodology suitable for each assurance level[20].

2.2.5 Common language

One of the goals of the ECSCF is to reuse existing parts, such as standards or other certifications, as much as possible. A common structure across different certification schemes is necessary to be able to combine them. This section will describe the structures introduced by the ECSCF to facilitate the combination of different schemes.

The ECSCF sets three structures to ease the utilization of different components in the same certification: the Generalized Protection Profile (GPP), the Generalized Security Target (GST) and the European Cyber Security Certificate (ECSC).

GPP defines a security problem from a previous risk assessment and security objectives expected from the target. It must also describe the features that need to be implemented to meet the security objectives and the assurance levels used to evaluate the device. The notion of GPP was borrowed from the Common Criteria [28].

A group of experts creates each GPP due to the need for a deep technical knowledge in the target's domain and threat landscape.

Based on the GPP, a GST is generated for the certification target. GST has all the requirements defined in the GPP and additional security requirements defined by the certification sponsor.

The certification results must be in a unified format the ECSC. It must include a label with the assurance level and a way to obtain detailed information more efficiently, such as a QR-code or NFC-tag pointing for an online version of the ECSC. The detailed version of the ECSC must contain the main attributes of the evaluation, such as the name of the product, the GPP used, the scope of the certification, and the results of the evaluation process.

2.2.5.1 Certification flow with the ECSCF

ECSCF denotes a certification scheme divided into four phases. First, the expert group designs a GPP for a type of certification target based on a Technical Specification of Security Requirements. Then, a GST is created for the certification target, based on a GPP, with the inclusion of the vendor's security requirements. The target is then assessed by a CAB tester according to GST and an evaluation report is generated. The CAB certifier reviews the report and makes the decision to grant an ECSC [23] (Figure 2.2).

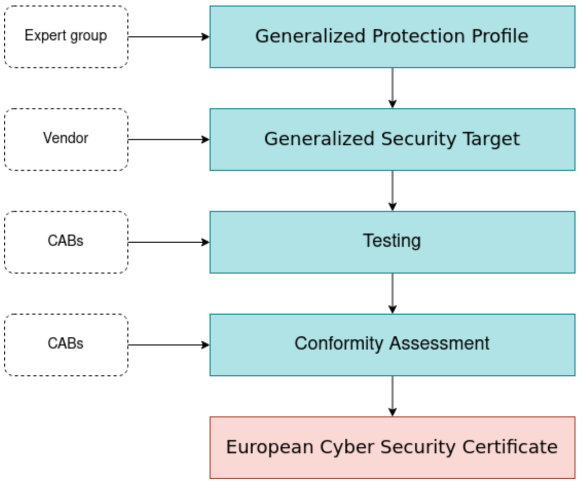

Expert group
Generalized Protection Profile
Vendor
Generalized Security Target
CABs
Testing
CABs
Conformity Assessment
European Cyber Security Certificate

FIGURE 2.2: Certification flow with the ECSCF

2.3 IoT Concept and Application Domains

In the previous sections, we presented the requirements that a certification that wishes to comply with the European guidelines need to fulfill. These requirements are domain agnostic. Next, we will describe different application domains for IoT technologies, and then we will introduce some specific requirements to IoT environments.

2.3.1 IoT Concept and Application Domains

Nowadays, different environments have devices that can communicate with each other and collect data from their surroundings. Its owners are taking advantage of IoT systems' new possibilities, giving them a more intelligent and responsive planet.

Users can have a car, a home refrigerator, and even the vacuum cleaner connected, with the ability to collect and transmit data. The car shows the GPS map in real-time; the refrigerator can show the validity of the products and the shopping list for the next month, and the vacuum cleaner can be programmed to start cleaning the room even if the user is not at home. An irrigation monitor for large plantations can be activated remotely. In public transport, delays and lack of information disturb many people who need to get to work or class. Users can receive on the smartphone or see the exact location on the screens installed at the points.

These devices communicate with each other, giving users the possibility of more comfort, productivity, and information. The use of these devices can include health monitoring, providing real-time information about city traffic or the number of spaces available in a parking lot, what direction they are in, and even recommending activities.

These are just a few of the numerous examples of IoT that already exist and are being created.

To better understand how IoT applications and devices are increasingly present in everyday life, we can divide them into two main areas: day-to-day and market areas, both industrial and infrastructure.

There are some distinct categorization models in the literature. *Atzori et al.* [9] divides the applications of IoT in four domains: transportation, healthcare, smart environments, and personal domains. *Gubbi et al.* [6] also categorizes the applications in four domains. However, they differ from the previous ones, dividing the domains into personal/home, enterprise, utility, and mobile. Another possible classification is presented by *P.P. Ray* [29], in which the environments divide into nine different domains: RFIDs, service-oriented architectures, wireless sensor networks, supply chain management and industry, healthcare, smart societies, cloud service and management, social computing and security.

Considering an IoT security certification, the categorization needs to group different environments with similar security requirements, to maximize the effectiveness of security measures applied to each environment in the application domain. So, the *Atzori et al.*'s approach is restrictive in a sense that does not differentiate environments with very distinctive security requirements. For instance, smart homes and industrial plants are in the same category, while, in terms of privacy concerns, a smart home demands more privacy requirements than an industrial plant. *P.P. Ray*'s categorization scheme is very detailed compared with the others, but it is focused on the technologies used and not on the security requirements. Thus, during our research, we will guide us by *Gubbi et al.* categorization scheme, because it is capable of grouping different applications with similar security requirements. It was designed based on the scale of the environment: individual, community, and national, since they tend to have equivalent security and privacy needs.

Therefore, the categories that we will utilize during this work are: personal and home domain, enterprise domain, utility domain, and mobile domain.

- **Personal and home domain** This domain represents all IoT systems that are used at the scale of an individual or home. It is included smart home systems and also healthcare systems. This technology is presented in the broad concept of automation, ranging from the lighting system to the security system. One of these benefits is the saving on domestic expenses, for example, on the electricity bill. Home automation can greatly improve the quality of life at the management level, but even for older people or mobility impairments.

- **Enterprise domain** The enterprise domain includes IoT systems within a work environment. Usually, these systems provide monitoring capabilities and process automation, allowing owners to reduce their costs. Examples of environments of this domain are enterprises, industrial and agriculture facilities.

- **Utility domain** The utility domain focuses on applying IoT technologies to a wide area, such as a system that is spread by a nation or region. This domain includes all kinds of smart meters, smart grids, and smart cities. Usually, these applications aim to optimize services, so the normal consumer for this type of technology is supply companies like electricity, water and communications, or public institutions.

- **Mobile domain** The significant characteristic of the mobile domain is the constant dynamism of the system over time. Examples of these domains are smart transportation and smart logistics. Transportation facilities are now with sensors and actuators. Besides that, also, the transportation infrastructure is being monitored. All the data that has been gathered allows owners and companies to monitor their assets and help them diagnosing and preventing problems.

2.3.2 General requirements

From the different literature available on the security requirements of IoT devices and systems, we have collected a list of resources that, ideally, a security certification focused on IoT needs to meet, namely security assessment, privacy impact assessment, pattern reuse, short time certification, attention to laws and regulations and application of an updated policy.

Any certification that aims to appraise IoT systems or devices must follow these requirements.

2.3.2.1 Security assessment

A security assessment is the process of determining how effectively a subject meets its security objectives [30]. There are different security assessments, such as risk analysis, vulnerability assessments, penetration testing, and audits, that are relevant for IoT environments.

Risk assessment is a way to identify threats and assess their likelihood of happening [31]. Depending on the type of context, this analysis will more accurately represent the risks to which the system will be subject, and as a result, it is possible to adapt the other certification processes to the specific requirements of a subject [32]. Each security certification must have a risk assessment because it can express a certification target's specific security requirements. Without it, the assessment will not adapt to the context of the device [33].

Vulnerability assessment is essential to minimize the problem of security vulnerabilities on IoT devices. This analysis aims to identify known vulnerabilities present in the system [30]. For the IoT environment, it has the advantage of being a lightweight process that can be automated. However, it only looks for known vulnerabilities. Therefore, when a certification only applies a vulnerability assessment, it cannot discover unknown vulnerabilities. *Penetration testing* is a technical assessment that attempts to simulate an attacker's actual behavior. This assessment differs from a vulnerability assessment, looking for known vulnerabilities and trying to discover unknown vulnerabilities and put the vulnerabilities found into practice to discover their real impact. For IoT environments, this analysis should be improved using automated tests to increase coverage while decreasing the execution time [34].

Audit is an assessment that compares an existing system configuration to a standard. Depending on the pattern, this may include other assessments, such as vulnerability assessments or penetration tests. This type of assessment is commonly used for the application of security guidelines.

Risk assessment is a mandatory analysis for a security certification, without which it will not pay attention to the system's context. Vulnerability assessment is an essential step in reducing IoT vulnerabilities; however, it has a limited scope when applied alone, assessing only known vulnerabilities. The penetration test is a robust assessment, although it must be kept in mind that this process is long and tends to be manual. The efficiency of an audit will always depend on the standard chosen. It is up to the certification to choose the type of security assessment that best suits its needs.

2.3.2.2 Privacy Impact Assessment

Currently, IoT certifications cannot be limited to security assessments and must also have a Privacy Impact Analysis (PIA). PIA is an analysis of how information is handled, to determine the risks of personal processing data and to assess safeguards to mitigate potential privacy risks [35].

Due to the proximity of these systems to human activities, mainly in healthcare and smart home systems, these systems can be a single point of failure for the environment's privacy. The privacy requirements will be different depending on the type of system we are dealing with. However, basic privacy requirements must be guaranteed [33].

2.3.2.3 Standard re-utilization

By itself, the IoT world is a heterogeneity source, with many different devices, protocols, and a lack of standards in general. Therefore, IoT certifications must combine different existing standards. Besides, this combination of existing standards increases confidence and respect for a new certification, as it inherits part of it from reused standards [20].

2.3.2.4 Certification time

The certification process's duration is often a factor in the non-certification of a product. Long certification processes increase production time, increasing costs and making the product financially unprofitable [16]. This situation must be avoided because the certification process must be as quick and light as possible to reduce certification time and minimize costs.

2.3.2.5 Laws and regulations context

Ideally, certification needs to include different laws and regulations that a given system must comply with. These obligations vary depending on the country in which the system will be installed, requiring certification to be modular and adapt to the system's context. The laws and regulations must be inserted in the certification as a way to promote it. For example, if the security certification evaluates compliance with applicable laws and regulations, the certification owner will know that he will not be subject to possible fines, thereby encouraging the use of the certification.

2.3.2.6 Update policy

One of the problems with IoT systems is the lack of security updates. When a vulnerability is discovered on a device, it is likely never to be fixed, even if it is publicly known [36]. Therefore, the certification authority needs to be aware of this situation and encourage the manufacturer to support their devices longer. If the certification authority does not implement an update policy, security vulnerabilities can be found on a certified device, discrediting certification.

2.3.2.7 IoT context aware

As we mentioned earlier, an IoT system is marked by its context. The system context is defined by different variables, of which we highlight the physical location of each element of the system, the topology of the system network, and the type of elements at a given time. These variables, in an IoT environment, generally vary over time.

This dynamic context is why many traditional security methodologies fail because classic security assessments are carried out periodically, and IoT systems change quickly, so the likelihood of a new device emerging between periodic assessments will be high. The security assessment will not predict and consider the possible changes that may occur before the next periodic assessment [12]. Thus, the effectiveness of certification can be compromised. Any security certification for IoT needs to be aware of this evolution in the system's context and assess the risks arising from this dynamism.

This requirement is only valid when a certification has a specific context, because, without it, we cannot monitor any change in an IoT environment.

2.3.2.8 Access to Guidelines

In addition to all the requirements mentioned above, we also propose a new requirement called *Access to guidelines* that should not exclude the possibility of systems that are not certified, but comply with certification. We mean that it is preferable that access to the certification guidelines is free and that only the certification process has costs. That way, we can minimize the problem of cheap IoT products that are not profitable for certification.

2.3.3 Context aware requirements

The application domain where an IoT system is inserted affects the importance given to each of the requirements mentioned in the subsection 2.3.2. To select the most suitable certification for each domain, it is necessary to detail the crucial aspects of a certification to evaluate a target in a specific domain effectively.

2.3.3.1 Personal and home domain

The focus of these systems is always the personal environment. The key requirement for personal and home domain is a data-driven security assessment due to being the most capable of endangering their users' privacy [37, 38].

In this type of domain, the cost of certification is essential. Context-specific certifications are expensive, so owners in a small-environments are unwilling to pay for this type of service. Therefore, these certifications are typically supported by manufacturers.

This domain also includes healthcare devices. Certifications for these devices must cover privacy requirements and specific regulations (for instance, the HIPAA regulation [39]), but besides that, it is also necessary to ensure that devices are fault-tolerant as failure could put in risk a patient's life [40].

2.3.3.2 Enterprise domain

The enterprise domain includes IoT systems within a work environment. Usually, these systems provide monitoring capabilities and process automation, allowing owners to reduce their costs. Examples of environments of this domain are enterprises, industrial and agriculture facilities.

Similar to the personal and home domain, the certification cost will be essential for this domain. The budget for an IoT certification in an enterprise domain will not be as low as the home domain but will not be the main focus. Depending on the importance of the IoT system for the company's primary revenue, it is advisable to evaluate the system with a specific context.

2.3.3.3 Utility domain

The utility focuses on the application of IoT technologies in a wide area. In this domain, all types of smart meters, smart grids, and smart cities are included. Typically, these applications are intended to optimize services, so the average consumer of these technologies are electricity, water, communications companies, or public institutions.

This domain contains the most critical IoT systems and is, therefore, the main target for attackers. The security assessment should be as complete as possible. Also, this domain certification should ensure that these systems are fault-tolerant, as essential services, such as water and electricity, depend on them [41].

2.3.3.4 Mobile domain

This domain has a continuous creation and destruction of communication channels and a constant movement of devices. This characteristic contributes to one of the most dynamic context. Thus, a certification for this domain needs to be prepared for a frequently changing system.

Finally, to sum up, the special requirements imposed by each application scenarios are summarized in the Table 2.1.

Domain	Important Requirements
Personal	• Privacy Concerns • Low Certification Cost • Healthcare Devices Regulations • Fault Tolerance in healthcare
Enterprise	• Low/medium certification cost • Specific context for critical systems
Utility	• Most complete security assessment • Ensure fault tolerance
Mobile	• Adapt to a dynamic environment

TABLE 2.1: Requirements for each application domain

2.3.4 ECSCF and IoT

ECSCF does not try to meet a specific area's requirements, but creates a framework that mitigates common problems in security certifications.

Some of the requirements mentioned in the subsection 2.3.2 are already present in the ECSCF, as they are not particular problems of IoT environments.

The concern about reuse existing parts is present throughout the ECSCF. This framework tries to unify security certifications. So, it also wants to reduce heterogeneity in security certifications and its heterogeneity marks IoT. Therefore, the "re-utilization of standards" requirement is ensured by the ECSCF.

IoT systems lack security updates, and this is one of the problems that must be minimized with security certifications. In the subsection 2.3.2, we stated that security certifications for IoT must enforce the implementation of updated policies. The ECSCF shares this concern and creates mechanisms at the certification level to monitor certified systems. Thus, ensuring that security vulnerabilities found in certified systems do not discredit the certification.

We also described the problem with cheap IoT products that are not profitable for certification, but it can be minimized by the ECSCF with different assurance levels and allowing self-assessment in some of these levels. The cost and duration of lower certification levels will be lower than higher levels.

In brief, even though the ECSCF is a generic approach, some problems affect any security certification. The concern about reuse existing parts, the time and monetary impact that a certification process produces in the production costs, and the enforcement of update policy is shared by both ECSCF and IoT literature. Thus, the legislative vision is not far from the IoT reality.

2.4 Security Certifications

Certifications for IoT systems and devices are beginning to emerge. This section will describe five certifications focused on IoT. From these certifications, two were designed by academic researchers (DSPSMA [42], ARMOUR certification [43]) and the other five were created by the industry (ICSA Labs certification [44], UL-2900 [45], BSI Kitemark for Internet of Things devices [46], IoTAA Security and Privacy Trustmark [47] and Eurosmart IoT Security Certification [48]).

2.4.1 DSPSMA

Dynamic Security and Privacy Seal Model Analysis (DSPSMA) [42] is a certification scheme that combines a real-time monitoring system and the existing certification model, where an audit is performed at a specific point in time. This certification is materialized in a dynamic seal applied to the system that can be updated over time. Currently, this certification does not have implementation yet.

Due to the attempt to combine a standard certification and a monitoring system, the DSPSMA is structured in two phases: the *initial certification phase* and the *monitoring phase*.

In the *initial certification phase*, although it is mentioned the possibility to adapt to other normative environments, the system is assessed according to the normative environment of the European Union, such as GDPR and e-privacy directive. Different technical standards and recommendations are also assessed. For instance, the ISO/IEC Information Security Management Systems standards will assure that security, privacy, and risk analysis will be performed to the system. This application of different standards in the same certification process minimizes the possibility of blind spots in the certification, as each standard has its limitations and, together, will cover each other's limitations. Due to the diversity of standards and regulations against which the system will be assessed, auditors and technical experts will conduct this part of the certification.

In the *monitoring phase*, the certified system will be continuously monitored. From this process, if a potential breach of system privacy and security is detected, the owner and the DSPS organization will be notified. The notification triggers an evaluation process by the customer for the system and an obligation to report the evaluation results to the DSPS organization. Depending on the severity of the breach, a system re-certification may be required. This mechanism guarantees the validity of the certification over time.

The monitoring part of this certification fulfills different functions, depending on the different types of users. For a generic end-user, DSPS acts as a user-friendly tool for checking the certified system's overall status. Advanced users, such as system owners and administrators, will have access to more advanced tools to view and analyze problems. The DSPS organization will serve as a surveillance mechanism to monitor the certified system, dynamically update the certification, and as probes to detect new threats from the IoT world.

Finally, an interesting point of the proposed architecture for this certification is applying a blockchain as a solution for log storage. The blockchain maintains the historical records of each certification and guarantees the authenticity of the data.

Existing standards support this certification. However, a crucial point is missing. It fails to provide quick unified guidance for the audition phase. In the technical certification report, this problem is recognized and identified as future work. This certification also uses general security standards, such as ISO 27001, which may not be prepared for specific requirements of IoT environments [12].

2.4.2 ARMOUR certification

The European Union's ARMOR project aims to address security and trust issues in the Internet of Things, providing automated and simplified testing, benchmark, and certification processes for many devices. One of the project results is the proposal for a cybersecurity certification (ARMOR Certification) [43].

This certification's design follows the main guidelines of ECSO, including ECSCF, but it also includes some additional components to accommodate the dynamic context where IoT devices are deployed.

This certification is divided into an initial security risk assessment and a continuous security testing phase. These two phases are based on the main European Telecommunications Standards Institute guidelines for risk-based security assessment and test methodologies (ETSI EG 203 251) [49].

The certification is divided into six phases, the last one being continuous over time. Initially, a context is established, where the objective is to understand the device's business and regulatory environment. A risk analysis is then performed, resulting in identifying the different vulnerabilities that can affect the subject. For each vulnerability, if the associated test does not exist, it needs to be created. Finally, the device is configured in the monitoring, and test structure and tests are performed on it.

The device is monitored continuously, and the tests are repeated in two situations: when a new vulnerability is found or when a new firmware update is released. If a new vulnerability is found, the tests are repeated after adding a test that assesses whether a particular device is vulnerable or not. Thus, the certification is updated when the device is unsafe, but the manufacturer has the opportunity to resolve it and recover the initial certification results.

One of the fundamental characteristics of this certification scheme is the labeling system, through which the result of the certification is transmitted to users. A QR code tag is attached to each device containing a link to a page. This page displays the updated certification status of the device. It can also contain the certification result for more specific contexts, such as when a device is used in a domestic environment [50].

This certification process aims to ensure that a device is not vulnerable to known vulnerabilities. However, it has no process for discovering new vulnerabilities that are specific to the device being scanned. Besides, IoT vulnerability databases are missing, which can limit the efficiency of this certification. Another problem is the scarcity of an analysis aimed at user privacy. Standard approaches tend to bind to GDPR requirements only if appropriate for the device and not as a general rule.

2.4.3 ICSA Labs certification

ICSA Labs is an independent company that provides product certification and testing to end-users and business entities. Among the different certifications offered, ICSA Labs offers one for IoT devices.

The certification is based on its IoT Security Testing Framework, a lightweight security guideline for IoT devices and their communications. The framework defines several security requirements grouped into seven domains, more precisely: encryption, communications, authentication, physical security, platform security, and alerting. In the case of encryption requirements, it is also important to note that they require a device to be compliant with the NIST and FIPS recommendations [44].

A device is certified if it meets the different security framework requirements mentioned above [51].

When certification is carried out in its guidelines, the device cannot be affected by any known vulnerability. However, it has no mechanisms to enforce security updates after the certification is issued. Besides, this certification is issued without an expiration date. Therefore, it is possible to have a device certified and affected by a vulnerability.

2.4.4 UL-2900

We will now cover the UL-2900 [45]. We did not have access to the UL-2900 certification guideline due to paid access. Thus, the information presented was based on the public information available.

The UL-2900 family is a group of standards developed by Underwriters Laboratories (UL) [52] for the security of network-connectable products and systems. This standard and related certification are intended to provide manufacturers with testable and measurable criteria for assessing product weaknesses and vulnerabilities and implementing risk controls. The American National Standards Institute has already recognized some of these standards [53].

This family of standards comprises three different groups: general requirements, industry requirements, and, finally, general process requirements.

In the industry requirements group, guidelines for a specific application domain are defined. For example, for healthcare products, UL 2900-2-1 details the basic principles used in healthcare devices.

In terms of product evaluation, certification is divided into different phases. Initially, documentation is gathered about its production process and functional requirements. After that, the product moves to the risk control and management phase, where different risks are assessed. A product security assessment is then carried out, confirming whether the necessary measures have been taken to minimize previously identified risks. The product is checked for known vulnerabilities, diffusion tests, and penetration tests. Finally, the certification authority issues a certification and a report with different tests, assumptions, and results [54].

As we were limited to public information about this certification and standard, we could not discover some characteristics. These include whether the certification has taken any steps to ensure that the manufacturer maintains secure updates or for how long the certification remains valid. However, from what we have recovered, it is essential to highlight that the security assessment is complete, as it includes fuzzing tests and penetration tests, allowing the assessment of more intrinsic characteristics of the subject. Although, because of this range of tests, it is also a very heavy certification. As a way to alleviate this problem, the certification authority allows the subject to be partially certified.

2.4.5 BSI Kitemark for Internet of Things devices

The British Standards Institution (BSI) is the UK's national standardization organization. It provides testing and certification services for several products, including IoT devices.

When it comes to IoT, it has a certification for IoT devices that offers three levels of guarantee: *Residential* (for residential environments), *Commercial* (for commercial environments) and *Advanced* (for high risk residential and commercial environments).

The certification will include ISO 9001 compliance tests, vulnerability testing, and penetration testing. The penetration test is designed to adapt to the attack's complexity according to the level of assurance. For example, at the *Residential* level, the techniques used to test the target will be less complicated than those used to test at the *Advanced* level. The device is also tested against the requirements of the ETSI technical specification for consumer IoT security (ETSI TS 103 645) [46].

BSI regularly monitors certified devices, repeating security tests to ensure that the devices remain secure.

It is also important to mention that ETSI TS 103 645 enforces the device measures to ensure user data privacy, regardless of the regulatory context. These measures are under the GDPR.

2.4.6 IoTAA Security and Privacy Trustmark

The IoT Alliance Australia (IoTAA) [47] is an Australian organization made up of different IoT companies and individuals, intending to promote good security practices.

One of their plans is to create an IoT certification for devices. Currently, certification is not formulated yet. However, its guidelines are now available under the name "IoTAA Security Guideline".

These guidelines consist of thirty mandatory requirements and seven recommendations. Manufacturers can choose not to implement, if it is not suitable, or to adapt the mandatory requirements concerning application.

The requirements address the security needs of IoT devices and privacy requirements, concerning alignment with Australia's Privacy Principles, an Australian regulation that guarantees protection.

This guideline has the particularity of enumerating standard protocols and advising on the appropriate configurations to guarantee the security of devices [55]. This fact presents a possible problem for this guideline, as the configuration will change over time, and the owner of the guideline needs to continually monitor these changes. Typically, other certifications delegate this type of recommendation to other specific guidelines for that matter. For example, most certifications delegate the recommended Transport Layer Security (TLS) configuration for the National Institute of Standards and Technology (NIST) guidelines.

2.4.7 Eurosmart IoT Security Certification

The Eurosmart IoT Security Certification Scheme (e-IoT-SCS) was created with the ECSCF requirements in mind. This certification focused on IoT devices as part of a typical IoT infrastructure, evaluating them with a *Basic* or *Substantial* assurance levels (Subsection 2.2.4).

In terms of target audience, this certification is designed to be sponsored by the IoT product vendor and answer to the necessities of an IoT service provider or device owner. Because of these design options, the evaluation procedure is dependent on privileged access to the device [24].

This certification scheme has Security Profiles for each type of IoT product, defining the security requirements and assurance activities for each specific security problem. The Security Profile considers the asset's sensitivity and the environment where it is operating, allowing to scale the necessary security controls following the identified risks. It also tries to cover the full attack surface from physical attacks up to cloud infrastructure attacks [48]. The evaluation procedure is also driven by a risk approach and includes a vulnerability assessment and a testing phase to demonstrate if the device implements the necessary security features. Depending on the assurance level, this could include a penetration test and fuzzing [56].

The e-IoT-SCS includes a phase of surveillance after the certification is issued. In this phase, the CABs involved in the certification process need to frequently re-inspect product samples. The Security Profile defines the frequency of these activities. Moreover, the CABs must monitor EU CSIRT sources for security alerts impacting the evaluated product [48].

Its security profile includes a wide range of stakeholders, including the device owner and a possible security operator/administrator, which may or may not have the necessary skills depending on the type of environment [57]. This type of premise is important to IoT environments because there are environments like smart homes where the user is a non-expert and expects a simple setup processes. On the other hand, in an environment like a smart city, there could be fully capable security operators to respond to threats.

This certification accepts self-assessment in the supply-chain evaluation and checks that security goals and assumptions are correct.

2.5 Fulfillment of IoT and EU requirements

We created three tables to make it easier to compare the certifications described in the Section 2.4. The Table 2.2 has the characteristics of each certification, the Table 2.3 includes the different security assessments employed by each certification and finally, Table 2.4 describes the fulfillment of the requirements that we defined in the Section 2.3.2.

2.5.1 Reflection on certifications

In the subsection 2.1.1, we mentioned that certifications have three fundamental characteristics: context type, scope, and audience (the audience is divided into sponsor and target). Table 2.2 exhibits the characteristics of the certifications mentioned throughout this work.

	Context type	**Scope**	**Sponsor**	**Target**
DSPSMA	Specific	System	System owner	System owner
ARMOUR	General	Device	Product Vendor	User
ICSA Labs	General	Device	Product Vendor	User
UL-2900	Application domain	Device & system	Product Vendor	User
BSI Kitemark	Application domain	Device	Product Vendor	User
IoTAA	Specific	System	*Unknown*	*Unknown*
ECSCF	Application domain	Device	Product Vendor	User

TABLE 2.2: Certification caracteristics

Most certifications are carried out concerning an application domain or general context, as they often perform during the production process, making it impossible to define a precise context for the system. For this reason, the DSPSMA is the only one with a specific context, as the target is certified when it is already deployed and not during production time.

Most certifications are sponsored by the product supplier and target the User/Consumer. The purpose of these certifications is to build trust among consumers of IoT devices.

DSPSMA is the only certification that sees the system owner as the sponsor and target of the certification. This certification targets critical IoT systems that require security evaluations for the actual deployment of IoT devices in a specific environment.

The sponsor and the certification target influence the scope of certification. Certifications sponsored by a product vendor will evaluate devices, while certifications sponsored by an end-user will target a complete IoT system.

Throughout the subsection 2.3.2, we identified the requirements for an IoT certification. Table 2.3 lists the different security assessments that are applied to the subject during the certification process, and in Table 2.4, we summarize how certifications implement the remaining requirements.

	Risk assessment	**Vulnerability assessment**	**Penetration testing**
DSPSMA	Yes	Yes	No
ARMOUR	Yes	Yes	No
ICSA Labs	No	Yes	No
UL-2900	Yes	Yes	Yes
BSI Kitemark	No	Yes	No
IoTAA	No	Yes	No
ECSCF	Yes	Yes	Yes

TABLE 2.3: Security assessment comparation

Security assessments based on an audit is a common way to determine the security of a subject. These audits include other types of analysis. ICSA Labs and IoTAA certifications use a list of security requirements created especially for the certification, and in addition to that, they require a vulnerability assessment. BSI kitemark also applies a list of security requirements, but this list is a European technical specification. As part of the audit, the BSI kitemark also conducts a vulnerability assessment and penetration testing. DSPMA employs a risk assessment and vulnerability assessment in its audit. ARMOUR certification does a risk assessment to find the device's right assurance level and a vulnerability assessment. UL-2900 and e-IoT-SCS are complete security audits, performing a risk assessment, vulnerability assessment, and penetration testing.

All certifications implement vulnerability assessments. On the other hand, penetration testing is the less common, with only three certifications featuring this method. This preference for vulnerability assessments is due to automating this type of assessment, making the certification process lighter than one with a penetration testing.

Relying only on a vulnerability assessment can put the device's security at risk, as this analysis looks only for known vulnerabilities. If the device has customized software that has never been analyzed, this method will not find them even if it has vulnerabilities. Penetration testing is the only way to discover unknown vulnerabilities. However, it has the disadvantage that it is mainly a manual process.

The number of tests and whether they are manual or not are the main factors for the certification process duration. Certifications attempt to overcome this problem with different assurance levels that will include additional testing with an increase in assurance levels.

	PIA	**Standard re-utilisation**	**Access to guidelines**	**Laws and regulations**	**Update Policy**	**IoT context aware**
DSPSMA	Yes	ISO 27001 ISO 29190 ISO 15408 NIST SP 800-122	Free	Depends on the country	Constant monitoring for threats	-
ARMOUR	No	ETSI EG 203 251 ISO 15408 ECSCF	Free	Depends on the country	Constant monitoring for threats	*N.A.*
ICSA Labs	No	ICSA Labs Security Testing Framework	Free	-	-	*N.A.*
UL-2900	No	UL-2900 Standard	Payed	Depends on the country	*Unknown*	*Unknown*
BSI Kitemark	Yes	ETSI TS 103 645 ISO 9001	Free	-	Vulnerability report program and regular surveillance	*N.A.*
IoTAA	Yes	IoTAA Internet of Things Security Guideline	Free	Australian laws	Vulnerability report program	*N.A.*
e-IoT-SCS	Yes	ECSCF	Free	Depends on the country	Active surveillance	*N.A.*

TABLE 2.4: Certification requirements

In the subsection 2.3.2, we mentioned the importance of a PIA, which was placed as a requirement to improve privacy concerns regarding user data, even when this is not mandatory by regulation. Therefore, during our analysis of the certifications, we only considered those that perform a PIA independently of the regulatory context. Certifications that carry out privacy analysis only if the subject is forced to do it by the law will be considered as not having a PIA. By this principle, the DPSMA, BSI Kitemark, IoTAA, and e-IoT-SCS have PIA's. The others have no privacy concerns or only have them when it is required by regulation. This lack of PIA's may be due to these certifications aim to make an automatic assessment of the certification target, but PIA's are done manually and interviewing the manufacturers, which would imply a delay in the certification process.

Generally, access to guidelines is free, except the UL-2900. DPSMA is a particular case where the certification guideline is free but refers to paid standards, such as ISO 27001.

Most certifications include the assessment of applicable laws and regulations. The only one that does not do this is the ICSA Labs certification. BSI Kitemark exclusively includes the GDPR.

From the certifications analyzed, we could identify two approaches to address the IoT update issue. One is the constant monitoring of the subject after the certification phase. If the certification authority detects a vulnerability, it will automatically update the certification report (DSPSMA, ARMOUR, BSI Kitemark and e-IoT-SCS). The other solution is creating a vulnerability report program, which includes public disclosure and patch of vulnerabilities found (BSI Kitemark and IoTAA). BSI Kitemark is the only certification that enforces these two types of control.

The requirement of awareness regarding the dynamic context in which the system is inserted is only applicable to the DSPSMA. Unfortunately, this certification does not have any attention to the constant change of the system's environment. The dynamic environment associated with IoT technologies is a known characteristic of IoT and is pointed out as one of the open research opportunities.

2.5.2 Reflection on domains

This section will analyze the certifications we have collected throughout this work, analyzing their adaptation for each application domain. We select those that meet the domain requirements and describe their advantages and disadvantages.

Personal and home domain

For the personal and home domain, the certification that most closely matches the requirements is e-IoT-SCS. It aims to certify devices in an application domain, charging the manufacturer with the cost of certification. The user receives a label indicating the level of certification with which the product was generated. In terms of privacy issues, e-IoT-SCS has well-established requirements to ensure the privacy of its users.

Another certification that can be used for this domain is the BSI Kitemark. However, this certification has the disadvantage that it is not compliant with ECSCF.

Due to its regulations and possible effects in case of failure, the healthcare devices sub-domain needs a certification as complete as possible based on healthcare regulations. Accordingly, the UL-2900 family is advised to this domain, especially as it has a specific standard for this type of devices (UL 2900-2-1).

Enterprise domain

The enterprise domain shares the low-cost certification requirement with the personal and home domain, so the e-IoT-SCS and BSI Kitemark can be an option for this domain.

In this case, the BSI Kitemark assumes a privileged position because it has an assurance level dedicated to the enterprise domain.

For IoT systems that perform a critical function on a company, there is no perfect certification. The DSPSMA is the closest candidate because it is the only one that evaluates a specific context. However, it is excessively heavy for small environments.

Utility domain

DSPSMA meets the utility domain requirements. This certification aims to evaluate a system applied to a specific domain. The possibility of supporting the systems' owners by monitoring their assets is an asset, especially when these systems are considered primary attackers' primary targets. Besides, during the audit phase, the system is inspected to verify that it is fault-tolerant, an essential feature for critical environments.

Mobile domain

There is still no security certification prepared for a fully dynamic system to meet the mobile domain requirements. For system assessment at the device level, the best option is to get involved in a certification, such as Armor. This certification is applied to each device, enabling the detection of vulnerabilities that can affect the system. If the owner opts for system-level certification, DSPSMA offers the possibility to assess specific subject requirements. However, it is unpredictable how certification monitoring will react to an extremely dynamic system. As we mentioned earlier, there is not any certification preparation for the IoT system with a dynamic context.

2.5.3 ECSCF and existent certifications

Only two of all certifications we reviewed are compliant with the ECSCF, the ARMOUR certification, and e-IoT-SCS.

Naturally, certifications that do not target the European market will not be interested in implementing the ECSCF. Four certifications target the European market, and only two are compliant.

e-IoT-SCS is the most recent certification and is compliant with ECSCF. ARMOUR certification was being developed when ECSCF was published and also implements ECSCF. DSPSMA is not compliant with ECSCF, because it was released before the ECSCF. BSI Kitemark is the only certification released after ECSCF that is not compliant with it. However, this certification was launched just six months after the release of ECSCF, so probably this is why it does not follow the ECSCF recommendations.

To sum up, new certifications are trying to be compliant with ECSCF. DSPSMA and BSI Kitemark are not compliant with ECSCF, probably because they were released around the same time as ECSCF and, to be compliant, they would need to be redesigned.

2.6 Future research directions for certifications

Throughout this research, we analyzed the existing certifications and identified some areas that are not sufficiently developed to meet the requirements imposed by IoT and, therefore, there are still open research opportunities in this field.

Security certifications that assess risks regarding user privacy are increasingly common. However, half of the certifications assessed here consider only the subject's privacy when required by law and not as a general rule. Therefore, certifications must continue to encourage privacy by design on their subjects, regardless of the regulatory context. Another area that needs to be developed in this field is how we can automate PIA's so that this type of evaluation is more easily integrated with automatic assessments.

There is currently a notable lack of attention to IoT systems that have a dynamic context in security certifications. Most of the certifications evaluate devices only, not the entire system. Therefore, there is a need for a new certification that focuses on IoT systems with a specific context to assess the IoT dynamic context better.

Another missing point is that most security certifications evaluate devices based only on security checklists and vulnerability scans, sufficient to know if a specific security measure is being implemented, but fails to check for unknown vulnerabilities software. It is necessary to find a way to test for unknown software vulnerabilities without compromising the certification duration.

We can also detect a trend that certifications are aimed at manufacturers and not entirely at users. The certifications we analyzed (except DSPSMA) see the user as a pure consumer of the final report and not as the certification process's initiator. Most certifications are requested by the manufacturer, probably due to the costs involved. DSPSMA is user-focused, but too heavy for small environments. There is still no light certification for small system owners who need assurance that their IoT system is secure.

Chapter 3

Related work

As we mentioned in the section 1.2, one of our goals is to develop an automatic assessment tool capable of evaluating an IoT environment without being dependent on privileged access to the device and capable of adapting to different security requirements.

Security assessments in IoT require new tools and techniques due to their heterogeneity, many devices, and resource constraints. The majority of these processes are manual, which is a barrier to enhancing the IoT security [12].

Security assessment tools should apply principles of modularization to provide flexibility to test various devices. The security assessments can be divided into three categories: interface testing (test interfaces used to communicate with the exterior, such as a web interface), transportation testing (test the network infrastructure, associated cryptographic schemes and communication protocols), and system testing (test the operating system that runs on a IoT device). We can list specific problems for each type of assessment and possible solutions for those problems.

Interfaces are commonly affected by physical, authorization, and account attacks due to weak security practices [58]. *Interface testing* tries to check if the target is vulnerable to these attacks. These tests are conceptually similar. They all crawl the entry points and submit a test payload. So, modular tests would facilitate creating frameworks that tests for different vulnerabilities and enables faster tests by removing setup delays. Another solution is fuzzing techniques, not only with random generated payloads, but also by creating payloads from the mutation of real interactions with the target [34].

The great challenge that IoT imposes to *transportation testing* is the existence of heterogeneous networks in the same system. It is common for the coexistence of multiple communication stacks, such as using TCP/IP and Zigbee at the same time. These different technologies make security testing difficult and increase the attack surface of the system. As each technology suffers from different network attacks and to access to these networks, it is necessary for different hardware. The evaluation methods must be prepared to deal with these different technologies.

In the *system testing*, the manual analysis of binaries is infeasible, given the large number of IoT devices to test and a big diversity of system attacks. Therefore, static analysis technologies and symbolic execution of binaries allow the automation of simpler tasks and assist the tester's work. Equivalent to these solutions, the virtualization of the whole system can also assist the system's analysis, allowing us to perform dynamic analysis of a system, even when we do not have privileged access to the device. In all these techniques, it is important to design them being agnostic in terms of CPU architecture because of the variety of architectures existing in IoT devices [59].

In addition to the mentioned attacks, there are two common attacks to multiple types of testing. Cryptanalysis attacks can be prevented through transport and system testing. Side-channel attacks are a more broadly type of attack, as they do not depend on attacking a specific implementation error, but rather leverage the information gained from a device to impact a component on the target. Examples of this type of attack are power analysis and timing attacks.

Each type of assessment checks the device for specific security controls. The goal of these controls is to mitigate a threat. Thus, the Table 3.1 describes the relationship between the different threats and the tests that checks the controls that are able to mitigate these attacks.

	Interface testing	**Network testing**	**System testing**
Physical attacks	●		
Accounts/authorization attacks	●		
Network attacks		●	
Crypto attacks		●	●
Software attacks			●
Sidechannel attacks	●	●	●

TABLE 3.1: Relations between tests and attacks that they try to prevent

In the Section 3.1, we review the literature on IoT security testing and next, in the Section 3.2, we examine existent technical specification of security requirements for IoT.

3.1 IoT security testing

The challenges we mentioned at the beginning of this chapter are beginning to be addressed by the academy. This section will review the existing work on IoT security testing and compare them with our proposal.

V. Sachidananda et al. [60] proposes a security testbed fully automated and designed for IoT devices. It intends to simulate real-world conditions to test the IoT devices in different contexts, and defines an architecture that enables the easy addition of new tests and automatic generation of reports. The security analysis phase uses open-source tools to port scan, fingerprint, and scan for vulnerabilities. However, it depends on a closed-source application for the orchestration of the tests (NI TestStand), which runs only on the Windows operating system. Moreover, another disadvantage of this approach is that it only focuses on the device's network attack surface. So, it does not detect problems like hardcoded credentials or other problems intrinsic to the device's operating system or hardware side.

O. A. Waraga et al. [61] also developed a security testbed for IoT devices. However, this security assessment is fully constructed with open-source tools. This testbed's peculiar feature is that it monitors the device's outgoing connections to detect connections related to known IoT malware. This can be an interesting feature to analyze devices that have been in a production environment because they may already be compromised.

G. Chu et al. [62] developed a process for penetration testing, based on the OWASP research's necessities, that facilitates the integration with automatic assessments. The process is divided into three stages: information gathering, analysis, and exploitation. Each stage is organized in tasks for each one of the IoT attack surface. These tasks involve running common penetration testing tools, such as *nmap*, *nikto* and *openvas*.

The automation of this process is done by modeling it in a belief–desire–intention (BDI) software model. The BDI model is characterized by three logic components: belief, desire, and intention. The model describes the actions that an agent should take, guided by the information collected in the previous assessment phases. This model tries to mimic the behavior of a penetration tester. Given the information known about the device, the automation process can perform a specific set of actions, which would be the actions that a penetration tester would do. This solution was not tested in a real environment.

O. Alrawi et al. [63] focused on the evaluation of IoT devices for a home environment and proposed a methodology that researches can employ to analyze this type of devices.

The methodology was based on a threat model that identifies the attack vectors in four different areas: device, mobile, cloud, and communication. Each of these areas was analyzed with open-source and closed-source security assessment tools. The evaluation results were systematized in a group of security scores, each one for the attack vectors considered. With this type of score, it is possible to identify the strong and weakest attack vectors in IoT product. During their research, they analyzed forty-five devices. The results from these tests and the respective scores can be found on an online portal [*]. These analyses were made without any automation process.

Another assessment procedure was proposed by *J. Chen et al.* [64], which created an automatic black-box fuzzing framework. Usually, IoT devices have a mobile application to control it. This assessment technique's unique property is that it uses the information extracted from the control application to guide the fuzzing of the device. This process is done by performing a dynamic analysis of the application, identifying the messages delivered to the device, and modifying them before they are sent. This was proven to be effective even in the presence of encryption. During the test phase of a prototype, they discovered eight new vulnerabilities with this technique by testing nine devices. These solutions have the advantage of being completely vendor-agnostic and do not require any privileged access to the IoT device.

Other authors, due to the need to be proactive in detecting new devices and monitoring changes in IoT environments, have created a more active system that, in addition to detecting security issues, also automatically takes actions to minimize their impact on security.

One example is the *IoT Sentinel* [65], which is an automated system capable of identifying the types of IoT devices connected to a network and automatically enforcing rules that minimize the impact of vulnerable devices on a network. The identification of the device-type is made using network traffic profiling and extracting device-specific fingerprints mapped to device-types with machine-learning. The vulnerability assessment is done by searching Common Vulnerabilities and Exposures (CVE) databases for vulnerability reports related to the device-type previously identified. According to the results of the vulnerability assessment, the level of network isolation is assigned. A device-type without reported vulnerabilities will be assigned to a trusted level, and, on the other hand, a vulnerable device is segregated from the network. This solution has two fundamental

[*]The evaluation portal is available online at: `https://yourthings.info`.

limitations: there is no mention of how often the vulnerability assessment is done, and there was no evaluation of the measures' effectiveness.

	V. Sachi -dananda et al.[60]	**O. A. Waraga et al.[61]**	**G. Chu et al.[62]**	**O. Alrawi et al.[63]**	**J Chen et al.[64]**	**M. Miettinen et al.[65]**	**Our solution**
Modular	✓	✓	✓	✓	✗	✓	✓
Customizable	✓	✓	✓	✓	✓	✓	✓
Vendor agnostic	✗	✓	-	✓	✓	✗	✓
Open Source	✗	✓	✓	✓	✓	✓	✓
Device detection	✓	✗	✗	✗	✗	✓	✓
Security score	✗	✗	✗	✓	✗	✓	✓
Automatic assessment	✓	✓	✓	✗	✓	✓	✓
Continuous assessment	✗	✗	✗	✗	✗	✗	✓

✓ = Implemented;

✗ = Not implemented;

- = Not applicable

TABLE 3.2: Security assessments for IoT

The Table 3.2 summarizes the findings of our research on IoT security assessments and compares them with our assessment solution. The requirements utilized during this comparison are the features that we intend to implement on our solution.

3.2 Technical Specification of Security Requirements

Our testing methodology will be based on a technical specification of security requirements. The effectiveness of our tests will depend on the requirements set out in the technical specifications.

A technical specification of security requirements is a document composed of multiple requirements. Each security requirement mitigates a type of attack.

The certifications analyzed throughout Section 2.4 are based on several technical specifications for the definition of security requirements. Some of them use general-purpose standards (such as *ISO27001* or *Common Criteria* [28]) and others use specific standards created for IoT.

In this section, we will focus on the technical specifications that target IoT devices, and we will evaluate their effectiveness according to the types of attacks mentioned in the Table 3.1.

There are four standards that are currently being used by security certifications to assess IoT devices, *ICSAlabs Security Testing Framework*, *ETSI TS 103 645*, *IoTAA Security Guideline* and *UL-2900*. Unfortunately, it was not possible to evaluate the *UL-2900* standard, due to the lack of public information available.

In the Table 3.3, we organized the security requirements of each technical specification by the type of attack they are trying to mitigate. From this, we retrieved the following conclusions.

ICSA Labs Security Testing Framework has principles covering all categories. However, they are too broad and, therefore, can lead to ambiguities during a certification process. The strengths of this standard are its principles regarding physical security and access controls.

ETSI TS 103 645 lacks in the prevention of physical attacks, but has many principles for minimizing the risk of software attacks. Besides, this standard addresses each principle in a very detailed manner, to eliminate possible uncertainties that may arise from more simplified descriptions.

IoTAA Security Guideline fails to prevent side-channel attacks, as it has no requirement to avert this kind of attack. This standard also addresses the different technologies and their configurations, such as which cryptographic algorithms should be used. This detail in a security standard requires the standard author to update the technical specifications to comply with best practices constantly.

Security assessment	**Physical attacks**	**Software attacks**	**Side channel attacks**	**Crypt- -analysis attacks**	**Network Attacks**	**Authorisation and accounts attacks**
ICSAlabs Security Testing Framework	B.6, D.1, D.2, D.3	E.1, E.2, E.3, E.4	A.3, A.5	A.1, A.2	B.1, B.2, B.3, B.5	A.4, B.4, C.1, C.2, C.3, C.4, C.5, C.6
ETSI TS 103 645	4.6-2	4.2-1, 4.2-2, 4.2-3, 4.3-1, 4.3-2, 4.3-3, 4.3-4, 4.3-5, 4.3-6, 4.3-7, 4.3-8, 4.3-9, 4.6-1, 4.6-3, 4.6-4, 4.7-1, 4.7-2, 4.13-1	4.4-1, 4.5-2	4.5-1	4.5-1, 4.9-1, 4.9-2, 4.9-3	4.1-1, 4.6-5
IoTAA Security Guideline	9, 34	4, 5, 6, 9		1	2, 31, 32, 33	10, 11, 12, 13

TABLE 3.3: Security controls organised by type of attack

To summarize, there is no perfect technical security specification; each one has its strengths and weaknesses. Even though, *ICSA Labs Security Testing Framework* has security requirements that mitigate each type of attack, they are too generalized and may create doubts during the implementation of the security requirements. *ETSI TS 103 645* and *IoTAA Security Guideline* maintain a balance between general application security requirements and clear instructions for its implementation.

Chapter 4

System design

During our analysis of existent certifications (Subsection 2.6), we realize that the majority of the certifications aim the manufacturer to be its sponsor and the user to be the consumer of the certification. This interaction type is the most common because, for a non-interconnected system, the vendor is liable when the system fails. Thus, he certifies his system as a way to reduce possible risks [21]. However, when different IoT systems are connected, a common reality in critical environments, the vendor's evaluation does not consider this additional attack surface. To rectify this problem, it is necessary to create new certifications that aim to certify complex systems in their specific context and are sponsored by the system owner. Our proposal is an assessment methodology that aims to support the creation of these new certifications.

As we stated in the Section 1.2, we have three main goals for our proposal:

1. **Creation of a test procedure that meets the ENISA best practices to be eligible as an evaluation method for a technical specification of security requirements.**
2. **An evaluation methodology that targets an agnostic vendor IoT environment with a specific use case.**
3. **Provide an automatic and continuous assessment method.**

4.1 Architecture

Our assessment system has two main tasks to perform. One is to run the security tests and the other is to analyze the results and generate the report and security score. Each one of these tasks, can be seen as an independent component, so we will refer to them as functional units.

The Figure 4.1 is the blueprint for the architecture of our proposal. In it, we can observe the two functional units, the *Security Testing*, which is responsible for executing the security tests, and the *Security Evaluation* that generates the report and security score. In the ECSCF certification flow (subsection 2.2.5.1), this graph can be interpreted as the certification test phase.

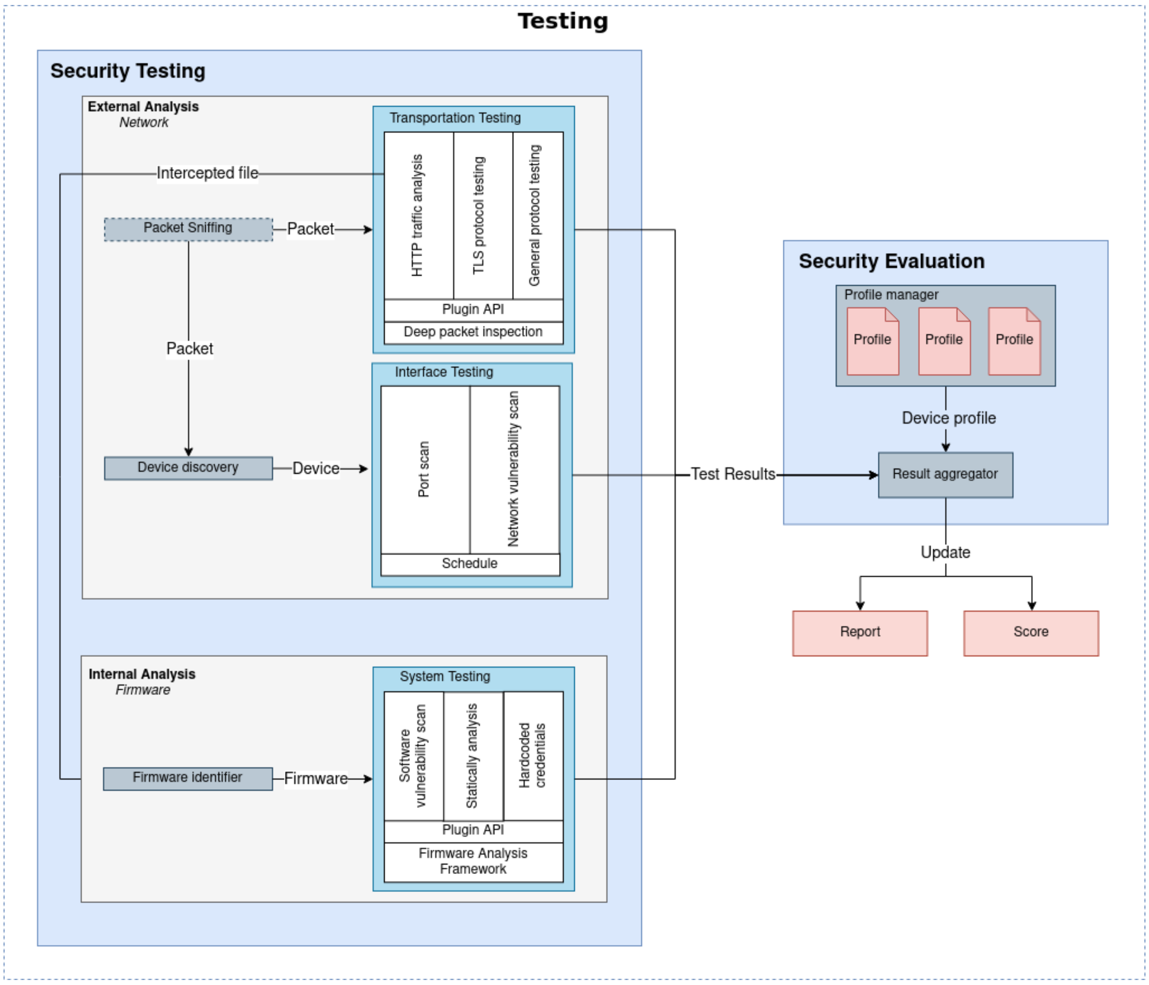


FIGURE 4.1: Architecture of our proposal

In the rest of this chapter, we will describe the components of each functional unit. Then, we will translate our goals into requirements and consider how we will meet each requirement, and finally, we will explore the limitation of our architecture.

4.1.1 Security Testing

The security test unit is responsible for testing and monitoring devices on a specific network. It also updates the test results according to changes in the environment, such as the appearance of a new device or changes in a device's firmware. The core element in this functional unit is the packet sniffer. The packet sniffer runs continuously, sniffing network packets and feeding the rest of the testing process. This component must be configured on a separate network and receives a copy of the packets with mechanisms, such as SPAN *, as it does not need to interact directly with the traffic. This is a great advantage, as it does not impact the network performance. As we analyze a copy of the traffic, any delay introduced by the analysis process will not affect the real traffic.

The packet sniffer will feed the *Transportation Testing* module. This module performs Deep Packet Inspection (DPI) to each network flow and identifies the underlying protocol. With the protocol identified, the packet is forwarded to the plugin associated with this protocol, which will run the necessary tests.

The sniffed packets are also used to discover new devices on the network. This information is shared with the *Interface Testing* module, which schedules periodic network vulnerabilities scans to each device.

The *Transportation Testing* module will identify possible firmware files from unencrypted traffic and send them to the *System Testing* module. The *System Testing* module is based on a framework that extracts the different components of a firmware image. This framework exposes a plugin API that can be used to implement different tests to the firmware.

Each of these test modules will produce results stored in a database for later use by the security evaluation unit. This also facilitates our system's adaption to different network topologies, as we can have several security testing units spread over the network that will send their results to a centralized database.

4.1.2 Security Evaluation

The security evaluation unit is responsible for translating the different tests' results into a unified report and score. The score may vary depending on the profile selected for the

*SPAN is a feature present in some network switches that allows us to mirror two network ports or *vlans* [66] device. These profiles represent the security requirements that the device must meet and are created in the previous certification phase.

When a new device is discovered, it is associated with a default security profile. The user can later change this association for the most appropriate profile.

The security profile is a structured document defining the relationship between tests and security requirements. Beyond that, the profile also defines the weight of each requirement in the security score.

The result aggregator has constant access to the most up-to-date test results. When the user requests a report and score, the result aggregator processes the latest information (security events with less than a day old and the latest results from a firmware analysis) and generates an updated security score and detailed report. This will serve as evidence for the next certification phase, conformity assessment.

Depending on the certification scheme's approach, this result aggregator can be adapted to generate a report and score every time there is a change in the security score of a device, at the same time, it notifies other systems of this change. This feature was not implemented.

4.2 Requirements

Given the architecture described above (Section 4.1) and the goals listed in the Section 1.2, it is important to explain how we are going to fulfill these goals. Therefore, in this section we will introduce the necessary requirements to meet our goals and then we will explain how these requirements are implemented by our architecture.

During this section, we will follow the approach of *Nicole Viola et. al.* [67] to define system requirements from the system objectives. The following enumeration will describe our goals and the requirements that are necessary to reach them.

1. **Goal** - *Creation of a test procedure that meets the ENISA best practices to be eligible as an evaluation method for a technical specification of security requirements.*

   As we mentioned in the Section 2.2, the European Union established a European cybersecurity certification scheme, which defines some characteristics for new security certifications. We are not proposing a new certification scheme, but rather an assessment methodology. The following requirements are extracted from documentation supporting the ECSCF.

(a) **Allow different assurance levels [20].** The ECSCF introduces the idea of different assurance levels depending on the context of the device. A customizable security profile for each device enables the implementation of different assurance levels, as it can define different weights for each requirement and different requirements according to the assurance level.

(b) **Be modular [20].** The architecture of our system allows several of our components to be adapted according to certification needs. All tests are designed with an API plugin that allows the addition of new tests according to the environment's needs. Another advantage is that the security profile is not linked to a single set of technical requirements or assessment tests. Therefore, tests can be reused by various security profiles.

(c) **Structured representation of the technical requirements and evaluation procedure [23].** The security policy, where security requirements and tests are declared, is represented as an XML file with a defined structure.

(d) **Produce comparable results [23].** The detailed report generated allows an easy comparison between devices regardless of the chosen security profile, when they share the same set of tests. Scoring can also be used to compare devices. However, we can only compare scores with the same assurance level, as the score varies with the security profile used.

(e) **Objectivity in the evaluation [23].** The ECSCF mandates that the evaluation should be non-dependent on discretional assessments and always produce repeatable results. As our evaluation is an automatic process, it is objective and reproducible.

2. **Goal** - *An evaluation methodology that targets an agnostic vendor IoT environment with a specific use case.* During our analysis of existent certifications (Subsection 2.6), we realize that the majority of the certifications aim to be sponsored by the manufacturer and therefore do not assess the specific context of the IoT system. This security assessment procedure intends to promote new certifications that can evaluate the specific context of a IoT system.

   (a) **Vendor agnostic [Subsection 2.3.2].** During a vendor-agnostic evaluation, there is no privilege access to the device. Our assessment does not require any privileged access, as the majority of the tests are network-based and the firmware's evaluation is done outside of the real system, with intercepted Over-the-air (OTA) updates.

(b) **Context aware [Subsection 2.3.2].** Many times, certifications use generalized criteria to evaluate each device. This type of evaluation is not able to meet completely the specific requirements imposed by its actual context. With the owner's security profiles, we can reach the security necessities of the device according to its actual function and environment where it is inserted.

3. **Goal** - *Provide an automatic and continuous assessment method.*

   (a) **Automatic.** The assessment procedure does not require user interaction.

   (b) **Continuous.** One of the challenges raised during our research on security assessments in IoT was that, with frequent changes in these environments, the effectiveness of the assessment is compromised. As our system will be continuously monitoring the network, a new device that appears on the network will be automatically considered in the evaluation results.

4.3 Components

The architecture introduced in the Figure 4.1 represents the high-level architecture of our system. It does not mean that it translates directly into the actual layout of the components. The Figure 4.2 describes the layout of the components of our system.

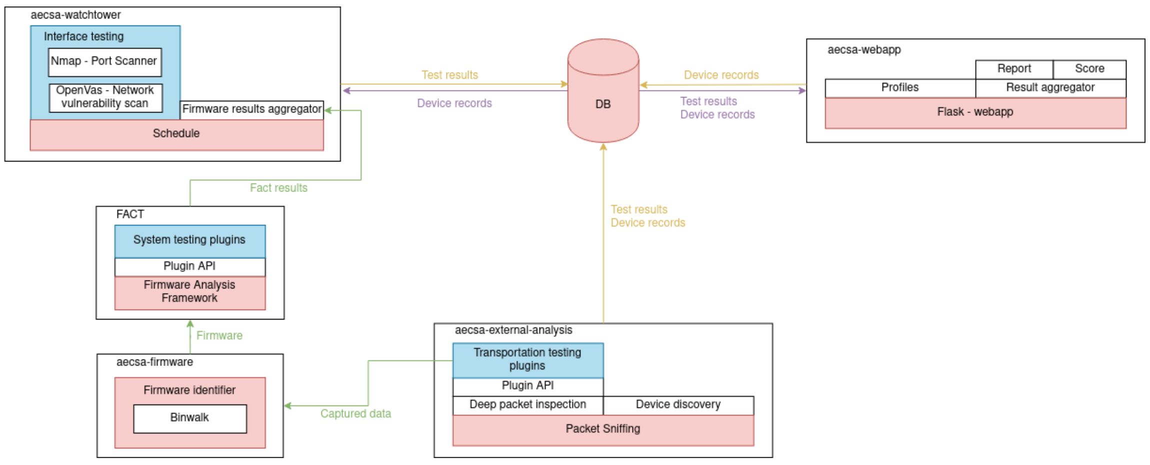


FIGURE 4.2: System components

In total, we have five components that communicate through a shared database. *aecsa-webapp* corresponds to the Security Assessment functional unit. The Security Test functional unit is divided into four different software. This division was made based on the existing software that we chose to do the tests and facilitate our solution's scalability.

Each of these components is designed to be deployed as a Docker container. This allows us to deploy and share our software [68] easily. We provide a docker-compose file to create and configure all components of our system. However, these containers can be run on any platform compatible with OCI images [69].

aecsa-external-analysis is responsible for transportation testing, packet sniffing, and device discovery. As it is self-contained software, it is possible to have multiple replicas of this component on the same network to facilitate analysis in a complex network topology.

aecsa-firmware receives, from a plugin in *aecsa-external-analysis*, objects that were extracted from the Hypertext Transfer Protocol (HTTP) communications. The *aecsa-firmware* analyses these objects and identifies if there is any OTA update and if there is, sends it to *FACT*.

The *FACT* is an automated tool for analyzing IoT firmware, developed by *Fraunhofer FKIE* [70].

aecsa-watchtower performs interface tests periodically and is also responsible for retrieving the test results from *FACT* and storing them in the same format as the other test results.

aecsa-webapp is how the user can interact with the analysis procedure. On this platform, the appropriate profile can be selected for each device found and where the updated results for the analysis can be accessed.

A fundamental part of our systems is that all data is stored in a database, and the database is used to share information between different components. For this role in our system, we will use a *MariaDB SQL Server* [71].

4.3.1 Assumptions

During the development of our project, we decided that we will only analyze Ethernet networks. This option was chosen to facilitate the implementation of a prototype.

Adapting this configuration to other networks based on the IP stack, such as Wi-Fi [72] or 6LoWPAN [73], should be relatively simple with the right hardware to capture network traffic [74]. However, networks using Z-Wave [75], ZigBee [76], Bluetooth Low

Energy [77] and other technologies that are not based on the IP protocol stack will be more difficult to adapt. The tests need to be adapted to the reality of the transport stack. For example, Bluetooth Low Energy uses the GATT protocol, which has a totally different structure from the IP protocol stack [78]. Therefore, testing Bluetooth communications requires important changes to our network packet processing system.

4.4 Summary

To sum up, our security assessment system performs two main tasks, continuously testing each device on a network and analyzing the results of those tests, generating reports and security scores.

Each one of these tasks is divided into smaller tasks, performed by independent components. This type of architecture division facilitates the adaption of our system to different network topologies and, at the same time, eases its scalability. We have four components for testing and one component for evaluating the results and generate reports. Each one of these components is deployed as an independent container and communicate with each other through a shared database.

Chapter 5

Implementation

In the Section 4.3, we described the components that make up our system. Many of these components needed to be implemented from scratch. This chapter describes the implementation process and the problems that emerged during the implementation.

This chapter follows the organization presented in the Subsection 4.1. First, we will describe the components belonging to the Security Testing functional unit, and then the implementation of the Security Evaluation components.

5.1 Security Testing

The components of the Security Testing are responsible for testing the devices and monitoring the network for changes in its topology. In this section, we will describe the implementation process of each one of these components.

5.1.1 aecsa-external-analysis

aecsa-external-analysis is responsible for sniffing packets, discovering new devices, identifying the protocol for each network flow, and performing the appropriate tests.

The *aecsa-external-analysis* needs to handle much traffic. As mentioned in Section 4.1.1, our system takes advantage of port mirroring to avoid introducing network delays. Even so, if we are slow-processing packets, the network interface will begin to drop packets.

Therefore, a balance is needed between performance and ease of implementation. The choice of the programming language used for this component is based on these conditions. The language must also provide a way to capture network data and run DPI, either natively or with a library.

Given these conditions, we had three possibilities for the programming language: *Python* [79], *Go* [80] and *C* [81].

Python

Python is an interpreted language with garbage collection and can call native libraries using *CPython*. This language is not focused on performance, but it provides a simple prototyping environment that is not affected by low-level security vulnerabilities (buffer overflow and heap overflow).

The *Python* language has two modules that are capable of decoding packages and performing DPI: the *scapy* [82] and *nfstream* [83]. *scapy* is written in *Python* and allows the developers to sniff and dissect network packets (its only external dependency is the *libpcap* to sniff network packets). *nfstream* is a library designed to work with online and offline network data. This library is based on the *nDPI* library, which is a *C* library for deep-packet inspection.

Using *Python* for this component would allow us to develop on a high-level programming language. Therefore, the development of the component would be facilitated. In terms of performance, the *Python* language is known as slow [84], but the *nfstream* is designed to be fast and with a small CPU and memory footprint. From previous experiences that we had with *scapy*, we know that it does not perform well with live network traffic.

C

C is a low level compiled language with manually managed memory. *C* is known for its performance, but it is also susceptible to various memory vulnerabilities due to its memory management.

The *C* language has two libraries able to capture network packets and perform deep-packet inspection: the *libpcap* and *nDPI*. The overhead of using these libraries is minimal because they are native to *C*, and in terms of performance, the *C* language offers the best performance compared to our two other options. Despite that, *C* is a complex programming language, requiring a high level of knowledge to avoid introducing vulnerabilities in the code.

Go

Go is a compiled language with a garbage collector, and it is known for its intuitive syntax and is optimized for performance and concurrency operations. The *Go* has two libraries that do what we need: the *gopacket* [85] and *go-dpi* [86]. The *gopacket* is a library implemented in *Go*, capable of sniffing and decoding network packets. The *go-dpi* is a wrapper library for the *nDPI* library.

In terms of security, as this language is a high-level language, it abstracts many of the memory management features, so it is not affected by memory security vulnerabilities introduced by the programmer.

Due to these facts, we decided to use *Go* in the development of *aecsa-external-analysis*, as it provides us with the balance between ease of development and performance.

5.1.1.1 Packet Sniffing

The base of *aecsa-external-analysis* is the packet sniffer. The *gopacket* library offers multiple ways to capture packets from the wire *libpcap*, *af_packet* socket and *libpfring*.

Initially, due to the amount of documentation available, we implemented the packet sniffer using *libpcap* on *gopacket*.

libpcap [87] is a library independent of the operating system and allows the capture of packets at the user level. The *libpcap* is used in most of the well known network tools, such as *tcpdump* and *wireshark*. Using this library to capture network traffic with *gopacket* is very well documented in the *gopacket's* documentation.

Our prototype was able to sniff packets. However, during the tests, we discovered an issue. We downloaded a file of 20MB while the program was sniffing and writing to a *pcap* file. Then, with *wireshark*, we tried to extract the original file and we were unable to extract the original file without it being corrupted. So, we concluded that we were suffering from packet drops somewhere in our setup.

First, we try to use *tcpdump* to redo our tests, because it also uses *libpcap* for sniffing and we want to understand if the error was on the implementation that we done or something in the host configuration. From this, we deduce that the error was in our implementation and not in a misconfiguration of the host machine.

We tried to find out if someone already had this error and discovered several mentions of this type of issue on Github, in which users complain about packet losses with *libpcap* [88]. This problem is not directly related to *libpcap*, but due to the way *Go* interacts with native *C* libraries. To call a function in a *C* library, *Go* uses *cgo* calls. This type of calls is a major performance overhead, due to memory duplication operations.

Therefore, this issue leave us two options, using *libpfring* or *af_packet*.

libpfring [89] and its associated kernel module *pf_ring* are a framework developed to process packets at high-rates, while providing a consistent Application Programming Interface (API) for packet processing applications. The *libpfring* introduces the concept of a circular buffer of packets (ring), allowing to buffer a fixed number of packets while the application does not process them. This implementation also allows multiple applications to read packets simultaneously and ensure that packets from the same flow will always end up being read by the same application. These are the features that make *libpfring* used by applications that need to process high traffic rates [90]. Unfortunelly, *libpfring* is also affected by the overhead of *cgo* calls [91].

So, we opted to use *AF_PACKET*, as it only calls *cgo* during its setup and not with each packet it receives [92]. *af_packet* sockets are used to receive or send raw packets in the device driver, so this is the native way on a Linux system to capture network data [93].

When we change the *libpcap* sniffer to an *AF_PACKET* based one, we were able to retrieve our original file from the *pcap* file.

In the packet sniffing, to decrease the amount of traffic that we need to analyze, we used a capture filter to capture only packets from the target subnet and filter packets without a network layer, because go-dpi analyses only packets that have a network layer.

The capture filters are applied directly to incoming traffic, at the kernel level, just after the raw data is received from the network card driver [94]. This way, we will increase our performance by parsing only the packets that will be used by the application.

5.1.1.2 Deep packet inspection

The DPI is performed by the *go-dpi* library in the *aecsa-external-analysis*. This library identifies the application layer protocols and also aggregates traffic to network flows.

In terms of use, this library is elementary. The library maintains a set of current network flows. When it receives a new packet, it identifies the flow to which the packet belongs and whether it is new. Then, with the packet flow, we can use this library to classify the flow protocol.

This library is based on the *gopacket* library for packet structure and definition of network flows. Before implementing this part, we analyzed *go-dpi* library to understand the underlying architecture, and we found a bug in the identification of network flows.

gopacket introduces two types of flows at two different levels: the network level flow (source and destination IP) and transport-level flow (source and destination network port). The *go-dpi* notion of flow is the conjunction of the two (IP's and network ports). However, due to an implementation error, the *go-dpi* was using only the network ports to identify the network flow. This issue has been reported to the *go-dpi* development team and has been fixed [95].

Each application protocol will have associated protocol handlers. These handlers are configured in the application as plugins. When the classification algorithm identifies a protocol, our application associates the network flow with a packet handler, and the packet handler analyzes every packet in that flow. Further packets of an already identified flow will be sent directly to the packet handler.

In these protocol handlers, we can implement security requirements tests.

5.1.1.3 Device discovery

Throughout this project, each device is identified by its MAC address. This design decision was made because IP addresses change over time, and the MAC address is unique and does not change.

Device discovery is also based on MAC addresses. This process happens once for each network flow. When a new network stream is identified, *aecsa-external-analysis* checks whether the source or destination IP address belongs to the network being analyzed and then retrieves its MAC address to verify that the MAC address was already seen. If the MAC address has never been seen, the device discovery will create a new entry device in the database.

For known devices, *aecsa-external-analysis* will update if the IP address changes.

The device discovery process is implemented with a hash table, which provides quick value searches and caches the database information not to query the database each time a new flow is detected. The program's hash table and database are synchronized periodically.

5.1.1.4 Plugins

The goal of having plugins is that we can develop new tests using our system. These plugins are developed with the native *Go* plugin API [96] and must have two methods that are called by the *aecsa-external-analysis*: *ApplicableProtocols* and *PacketHandler*. *ApplicableProtocols* returns a list of strings with the protocols to which this handler applies, while *PacketHandler* is the method that executes the security test.

The first limitation is based on the plugins' restricted access to the *aecsa-external-analysis* global variables. The arguments and return of functions limit the communication between the plugin and the parent program. Thus, to a plugin communicate with the database, it would need to connect himself to the database and re-implement many of the already available functions in the main program. So, to maintain lightweight and self-contained plugins, our plugins will log the security events using a REST API hosted by the *aeca-webapp*.

The other limitation of *Go* plugins is more a development limitation than a feature limitation. The versions used by the plugin libraries must be the same as those used by the main program.

5.1.1.5 HTTP traffic analysis

To connect the *aecsa-external-analysis* to the *aecsa-firmware*, we developed a plugin that analyzes HTTP traffic and sends the results to the *aecsa-firmware*.

This plugin analyses network flows that were identified as HTTP flows. It reassembles the Transmission Control Protocol (TCP) stream and extracts the HTTP bodies. These bodies are sent to the *aecsa-firmware* to be analyzed. The reassemble of the TCP stream uses the *tcpassembly* and *http* package to decode the HTTP protocol.

The communication between the *aecsa-external-analysis* and the *aecsa-firmware* is made using *gRPC* [97]. *gRPC* is a high performance remote procedure call framework and works across multiples languages, including *Go* and *Python*.

For this plugin, we choose to use *gRPC*, because the implementation of communications across multiple languages is simple. We only need to define a function that we pretend to share, and *gRPC* automatically generates the client and a server code in the necessary languages. Also, as we need to share binary data between the two services, the *gRPC* is ideal because it uses Protocol Buffers [98], which is a serialization method for structured data that allows serialize binary data.

5.1.1.6 Packets workflow

To summarize, the packet sniffer filters out all unnecessary network traffic and sends the remaining packets to the DPI phase. In the DPI phase, the packet flow will be identified. If the protocol flow was previously identified, the packet would be passed to the packet handler; otherwise, the protocol identification algorithm will analyze all packets belonging to the flow. If the protocol identification algorithm successfully identifies the protocol flow, the result is stored to decrease future packets' processing time in this flow (Figure 5.1).

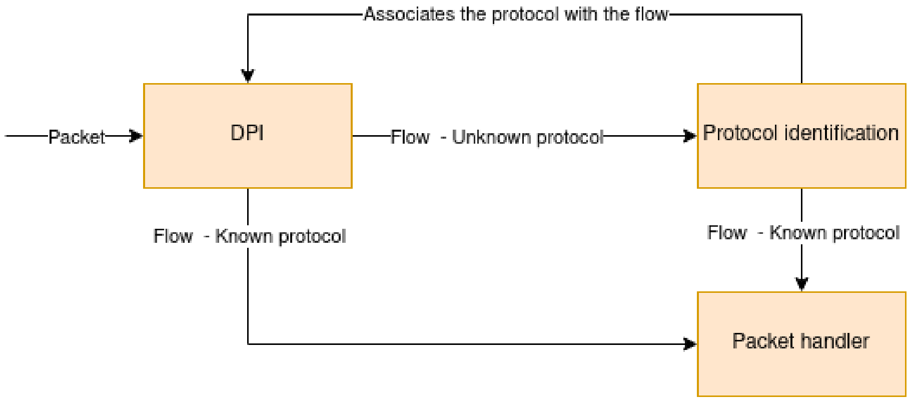


FIGURE 5.1: Packets workflow

5.1.2 aecsa-watchtower

aecsa-watchtower is responsible for scheduling interface tests periodically for each device. Its base component is an implementation of *cron* in *Go* [99]. The *aecsa-watchtower* checks the database for new devices every 5 minutes. If a new device is found, a new *cron* entry is created to run interface tests once a day. The *aecsa-watchtower* has three types of interface testing: port scanning, default credential scanning and vulnerability scanning. Port scanning is implemented using the *nmap* scanner [100], while the default credential and vulnerability scanning uses the *Open Vulnerability Assessment Scanner (OpenVas)* [101].

Nmap is a utility for network discovery and security auditing. This tool allows tasks, such as network inventory and monitoring hosts. In our case, we will use this tool to scan the open ports of a device and identify which operating system the device is running. To integrate *nmap* with our tool, we used the *Go* library called *nmap* [102] that allows to have bindings for the *nmap* scanner.

OpenVas is an open source vulnerability scanner, developed by the *Greenbone* company. Normally, the *OpenVas* is available inside the *Greenbone's* complete solution for vulnerability assessment and asset management.

The integration of *OpenVas* with *Go* was more difficult than the integration of *nmap*. The *OpenVas* has an API that enables remote management of the scanner, but there is only libraries to communicate with this API in *Python*. Moreover, the *OpenVas* is usually provided within the *Greenbone* stack (the integration of *OpenVas* scanner with a complete set of tools for management). Thus, to include *OpenVas* in our solution, we need to develop a *Go* library that is able to communicate with the *OpenVas* API and create an independent *OpenVas* installation to maintain a small footprint.

The API that is available to manage the *OpenVas* is the Open Scanner Protocol (OSP) [103]. OSP is a protocol developed by *Greenbone* that allows to manage a great diversity of scanners in a unified way. The OSP is based on *XML* objects exchanged through a TLS socket. The TLS connection uses client and server certificates for authentication on both sides.

Go offers a native TLS client and *XML* parser implementation. To implement a library capable of managing *OpenVas*, we created a representation of the different OSP *XML* objects in *Go* and developed functions to abstract the interactions with the server. The developed library is available on *Github* [*].

To create a small footprint *OpenVas*, we built a minimal container with an *OpenVas* installation, an OSP server and a set of scripts to download and update vulnerabilities metadata. This container is available on Github and Dockerhub [†].

Thus, the *aecsa-watchtower* uses *go-osp* to manage a minimal *OpenVas* docker. The use of *osp* to create new scans in *OpenVas* was not straight forward. The *OpenVas* uses Nessus Attack Scripting Language (NASL) scripts to test vulnerabilities. These scripts are organized by their function or the type of vulnerability they are trying to test. For instance, if we want to scan for default credentials on a service, we need to select the "Default Credentials" family. However, these scripts have dependencies; for instance, it will only check the default credentials for a telnet service if it knows that the device is running this service. *OpenVas* does not automatically select these dependencies, so to create any scan for a device, we first need to select the "Port Scanner" family and then the type of vulnerability we want to test.

[*] `https://github.com/MrSuicideParrot/go-osp`

[†] `https://hub.docker.com/r/cirne/openvas-light`

In addition to the interface tests, the *aecsa-watchtower* is also responsible for aggregating the results of the firmware testing. As we mentioned earlier, the FACT performs the firmware testing, and its output needs to be imported for our system.

As the main functionality of the *aecsa-watchtower* is to run periodic tasks, we used this component to regularly check the FACT for new reports and import them to our system.

5.1.3 aecsa-firmware

Firmware analysis relies on OTA updates and assumes that these updates are not encrypted. Most manufacturers use custom ways to update their devices. It would be impossible to support all this diversity. So, we will focus on a standard for OTA updates. For our system, we choose the Software Updates for Internet of Things (SUIT) specifications [104], because the Internet Engineering Task Force (IETF) is developing it, and usually, their standards have a great acceptability by the community [105].

The SUIT working group specifies a manifest which provides information about the firmware and security mechanisms to protect the integrity of the manifest and firmware [106].

There are many custom ways to update IoT devices, but if a system does not follow this guideline, we will consider the OTA update as insecure.

To verify if the OTA update is secure according to the SUIT specification, we implemented a parser for the SUIT manifest and stored the information until we receive the firmware. We cannot verify the cryptographic signature of the manifest because we do not know which key was used to sign it, but we are checking if the firmware hash available in the manifest matches any of the received firmware.

The development of the parser for the SUIT manifest was made using the SUIT specification [106] and the previous work made in this topic by *Koen Zandberg et al.* [107], which created a demo update server to test the SUIT specification.

Thus, when the *aecsa-firmware* receives bodies of HTTP requests, it verifies whether it is a CBOR payload [108]. If it is a CBOR payload, the application will try to decode it as a SUIT manifest and then stores the hash algorithm and firmware's hash. If it is not a CBOR payload, the application will check if the payload is a firmware image using *binwalk*. If receiving a firmware image, it will calculate its hash and then check if it matches any previous hash extracted from manifests.

aecsa-firmware, after identifying a firmware, will submit it to the FACT and register in the database the ID assigned by the FACT to the firmware. This ID will later serve to retrieve the results of the analysis.

5.1.4 FACT

FACT is a firmware analysis and comparison tool intended to automate the firmware analysis process. It can unpack many of the common formats of firmware and analyze the resulting content.

This system provides a plugin API to facilitate the development of new unpacking and analysis methods. It also offers a REST API that allows us to submit new analysis and retrieve its results. These two characteristics were the key to the integration of this system on our automated testing system.

By default, FACT includes several plugins. The majority of these plugins can be adapted to evaluate security requirements.

The main disadvantage of FACT is its minimum requirements. It is necessary to have at least 4 cores and 8 GB of RAM. These specs make it unprofitable to have a machine dedicated to this task, when it is used only to analyze firmware updates, and these updates are not that frequent. These high requirements are due to the security static analysis done to each binary found in the firmware.

It is possible to disable this plugin, but we lost the ability to analyze unknown binaries' security. A solution for this problem would be to put this machine in a VPS and turn it on when necessary or share it among different deployments.

5.2 Security Evaluation

The security evaluation was developed as part of a web app. This web app allows the user to manage the existent security profiles and obtain the most updated evaluation results.

In this section, we will first describe how a security profile will be represented and evaluated. Then, we will explain the features that have been developed to control our evaluation system.

5.2.1 Representation and evaluation of a profile

A security profile is defined by the ECSCF as a document that states the security problems that we are trying to solve, the objectives that we intend to reach, and the security features that need to be implemented to fulfill these objectives [20].

To automatize the evaluation of a profile is necessary to translate these security features into security requirements. These requirements are then organized in a security policy, represented as a *XML* file.

Therefore, to evaluate a security profile in our system, it is necessary to create a security policy on a *XML* file. This file has two parts: a header and a body.

The header is identified by the *metadata* tag and it has two fields: the name of the profile (*name* tag) and the profile's unique identifier (*uuid* tag).

The body has tests organized by their category (*category* tag). Each test has a standard id, a reference for the standard used to define the profile, and the points that this test can contribute to the overall security score (*check* tag). A test is a query to a database where events are stored from the evaluation point of view. Each *check* tag can have other tags inside, which will influence how the score is calculated.

If the *check* tag does not have any sub-tag, the evaluator will check if the database has any event of non-compliance with this requirement, and if not, the points associated with the test will be added to the score of the device.

There are three possible sub-tags: *vuln*, *range* and *check*. If a *check* has sub-tags, they must all be of the same type.

The sub-tag *vuln* is used when it is necessary to evaluate the results of a vulnerability scan. This type of test will decrease the device's points according to the severity of the vulnerabilities found. This is done using the field *multiplier*, which is present in each *vuln* tag. The multiplier field can vary from 0 to 1, and for each vulnerability found, we multiply the current points of the test by the multiplier value (Equation 5.1). For instance, in a test that can give 1 point, if we find two vulnerabilities classified as low and the *multiplier* associated with low severity vulnerabilities is 0.75, the result of this test will be 0.5625, because, for the first vulnerability, we multiply 1 by 0.75, and then we multiply the result of this operation again by 0.75.

$$SRq(x) = points(x). \prod_{s=low}^{critical} multiplier(s)^{n*} \qquad \text{(Equation 5.1)}$$

*SR: Score of the requirement

The sub-tag *range* is used to assign points according to the number of events registered. To each *range* tag, there is a multiplier value and a range of numbers. To evaluate *range* tags, the number of events related to a security requirement is counted, then the points of the security requirement are multiplied by the multiplier value assigned to the matched range. This type of tag could be used to evaluate the number of network ports open on a device.

A *check* can also have other *check*s as sub-tags. In this case, each test result will contribute to the overall score of the parent test. So, if the parent test has 3 points and three sub-checks, each sub-checks will count 1 point (Equation 5.2).

$$SRq(x) = \sum_{i=0}^{n} \left(SRq(i) . \frac{n}{points(x)} \right) \qquad \text{(Equation 5.2)}$$

In addition to these tags, there is also the *if* tag, which allows us not to evaluate part of the security profile if it is not applicable. For instance, if we have TLS configuration test, and the device does not have TLS traffic, the points referring to this test will not enter the device's score.

The overall score (S) is generated from evaluating each of the tags and counting the number of points obtained. The final result is shown as a percentage between the points counted and the maximum number of points that a device can get. The *if* tag influences this maximum number of points. Points that are within an *if* tag are counted only if they are applicable (Equation 5.3).

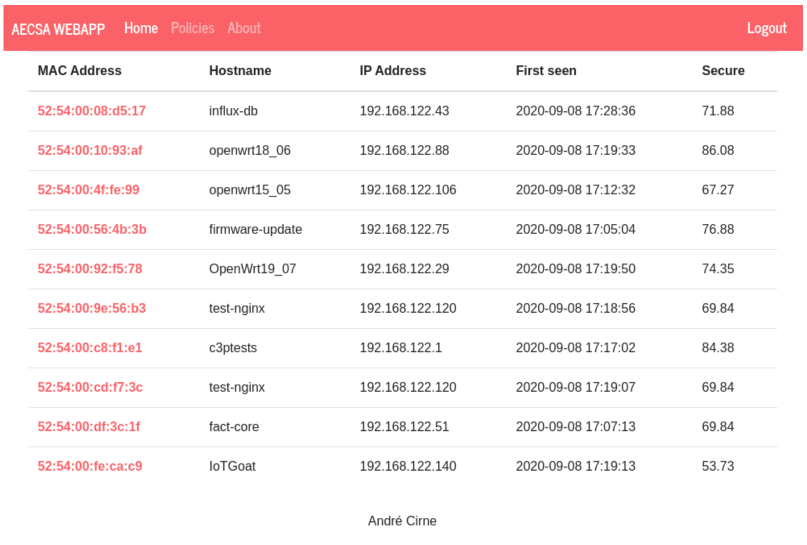
$$S = \frac{\sum_{i=0}^{x} SRq(i)}{\sum_{i=0}^{x} points(i)} .100 \qquad \text{(Equation 5.3)}$$

5.2.2 Web application

The web app was developed using Flask [109], which is a lightweight Web Server Gateway Interface (WSGI) web application framework. Flask supports the creation of webpages using *Jinja* [110] templates. When the user requests a webpage, the template is rendered with the most updated results. Thus, whenever a user intends to access a device's results, the score and report are generated with the most recent information.

The web application has three main pages: the home page, the device summary, and the device report.

The web app's home page features a summary of all the devices detected and their security scores (Figure 5.2).

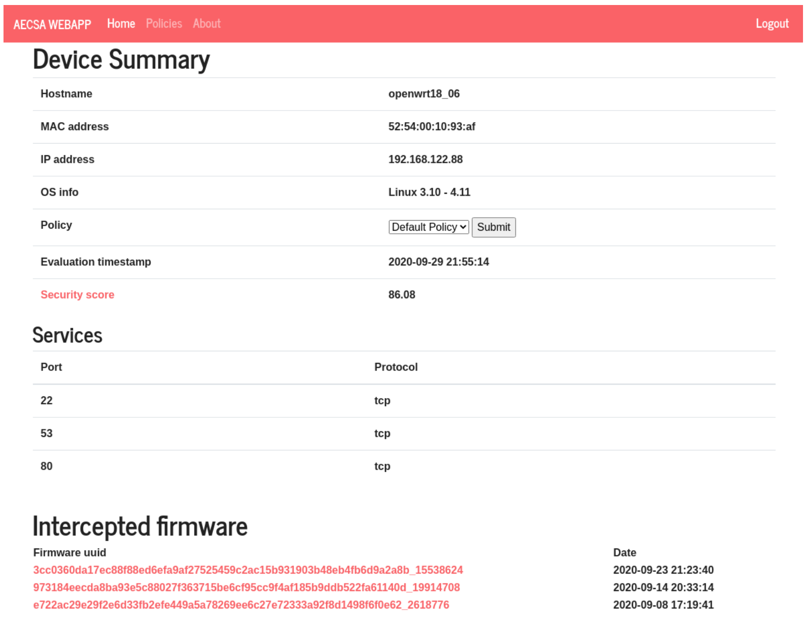


MAC Address	Hostname	IP Address	First seen	Secure
52:54:00:08:d5:17	influx-db	192.168.122.43	2020-09-08 17:28:36	71.88
52:54:00:10:93:af	openwrt18_06	192.168.122.88	2020-09-08 17:19:33	86.08
52:54:00:4f:fe:99	openwrt15_05	192.168.122.106	2020-09-08 17:12:32	67.27
52:54:00:56:4b:3b	firmware-update	192.168.122.75	2020-09-08 17:05:04	76.88
52:54:00:92:f5:78	OpenWrt19_07	192.168.122.29	2020-09-08 17:19:50	74.35
52:54:00:9e:56:b3	test-nginx	192.168.122.120	2020-09-08 17:18:56	69.84
52:54:00:c8:f1:e1	c3ptests	192.168.122.1	2020-09-08 17:17:02	84.38
52:54:00:cd:f7:3c	test-nginx	192.168.122.120	2020-09-08 17:19:07	69.84
52:54:00:df:3c:1f	fact-core	192.168.122.51	2020-09-08 17:07:13	69.84
52:54:00:fe:ca:c9	IoTGoat	192.168.122.140	2020-09-08 17:19:13	53.73

André Cirne

FIGURE 5.2: Web app homepage

The device summary displays a summary of the device's information, including their security score identified operating system, and captured firmware. This page also gives access to the device report (Figure 5.3).



Device Summary

Hostname	openwrt18_06
MAC address	52:54:00:10:93:af
IP address	192.168.122.88
OS info	Linux 3.10 - 4.11
Policy	Default Policy Submit
Evaluation timestamp	2020-09-29 21:55:14
Security score	86.08

Services

Port	Protocol
22	tcp
53	tcp
80	tcp

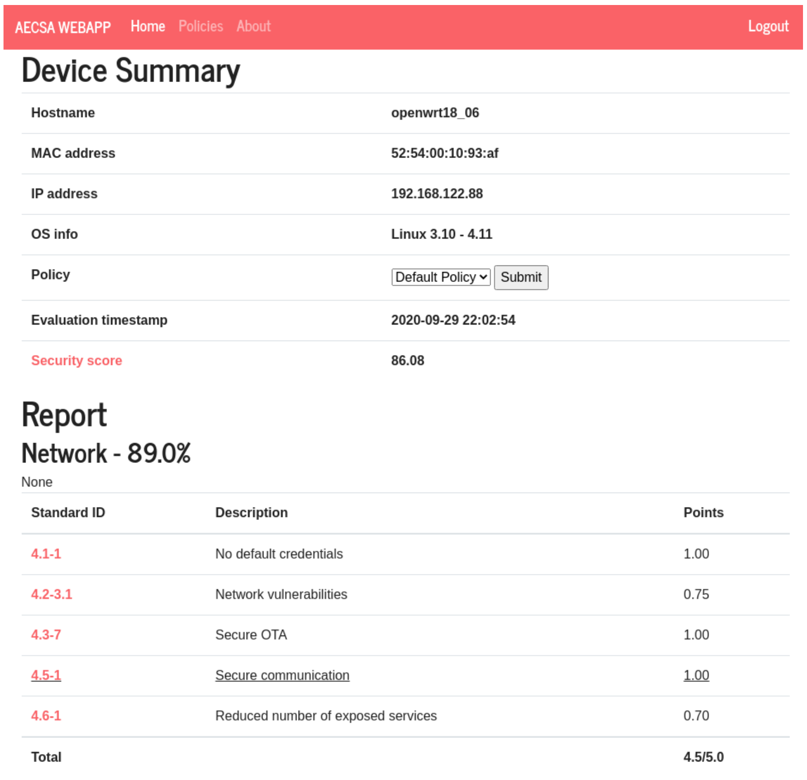
Intercepted firmware

Firmware uuid	Date
3cc0360da17ec88f88ed6efa9af27525459c2ac15b931903b48eb4fb6d9a2a8b_15538624	2020-09-23 21:23:40
973184eecda8ba93e5c88027f363715be6cf95cc9f4af185b9ddb522fa61140d_19914708	2020-09-14 20:33:14
e722ac29e29f2e6d33fb2efe449a5a78269ee6c27e72333a92f8d1498f6f0e62_2618776	2020-09-08 17:19:41

FIGURE 5.3: Device summary

In the device report, the user can explore each security requirement's results, with partial scores for each security requirement category and access to the detailed information about each requirement, namely the security events that influenced this score (Figure 5.4).

Besides, the web application has a page that allows the user to upload security profiles. By default, the application uses a default profile for each new device that discovers.



Device Summary

Hostname	openwrt18_06
MAC address	52:54:00:10:93:af
IP address	192.168.122.88
OS info	Linux 3.10 - 4.11
Policy	Default Policy Submit
Evaluation timestamp	2020-09-29 22:02:54
Security score	86.08

Report

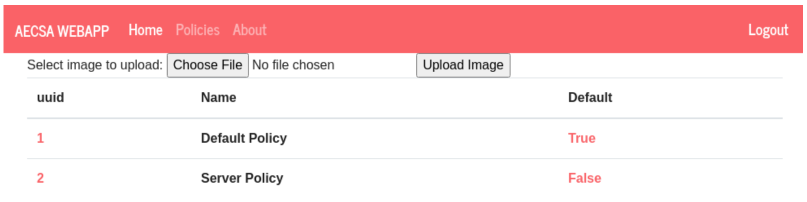
Network - 89.0%

None

Standard ID	Description	Points
4.1-1	No default credentials	1.00
4.2-3.1	Network vulnerabilities	0.75
4.3-7	Secure OTA	1.00
4.5-1	Secure communication	1.00
4.6-1	Reduced number of exposed services	0.70
Total		4.5/5.0

FIGURE 5.4: Device report

However, the user can assign a specific profile to a device or change the default profile (Figure 5.5).



Select image to upload: No file chosen

uuid	Name	Default
1	Default Policy	True
2	Server Policy	False

FIGURE 5.5: Policies manager

The *Flask* application also has a REST API to register security events. This API is used by the plugins that have no connection to the database to register their events.

The web application is currently not hardened in terms of security, all the users have the same permissions, and the REST API for plugins does not have any authentication.

5.3 Summary

During the implementation of our system, we implemented multiple components that can be reused by other projects. These components were made available on Github as independent projects (Table 5.1).

Name	Description
go-osp	A client implementation for the Open Scanner Protocol in Go lang
openvas-light	A dockerized version of openvas, independent of the greenbone stack
FACT-search-secrets	A FACT plugin to search secrets in firmware
Yara-Secrets	Yara rules to detect secrets like passwords, api keys, and tokens.

TABLE 5.1: Independent components that were developed during this thesis

Moreover, our main project is also available on Github (`https://bit.ly/3ecgGzZ`).

Chapter 6

Evaluation

Our proposal goal is to create a test procedure that respects best practices from ENISA, targets IoT environments, and can test them automatically and continuously.

To evaluate if our goals have been met, we tested with a virtualized IoT environment. The tests were divided into three phases.

In the **Phase 1**, we create different security profiles from a security requirements technical specification. With this, we can check if we can represent a technical security specification as an automatic security assessment (Requirement 1.c, 1.e, and 3.a).

Then, we perform the **Phase 2** that focus on evaluating our IoT environment against the security profiles created before. In addition to assessing whether the devices are being correctly evaluated, we will also check if the results can be compared (Requirement 1.d) and does not require privileged access to evaluate the device (Requirement 2.a). In this phase, we also tested if the assessment methodology can adapt to changes in the IoT environment (Requirement 2.c and 3.b).

Finally, in the **Phase 3**, we evaluate the difficulty of adapting to an existing security profile for a new version of a technical security specification (Requirement 1.b).

This chapter describes the IoT environment used to test our system, all the phases mentioned above, and a reflection on our tests' results.

6.1 Test environment

The tests were performed on a fully virtualized network with eight virtual machines. The hypervisor used was *QEMU 4.2.0* on a server with *Ubuntu 20.04 LTS*. In terms of specs, this machine has a *i7-9700K CPU*, which supports VT-x extension for virtualization, 32GB of RAM and 1Tb of *NVMe* storage.

QEMU can virtualize machines with architectures beyond the hypervisor CPU architecture [111]. However, to facilitate the test environment's setup, we decided to have only x86-64 architecture machines.

Our network will simulate an enterprise network with four IoT devices, a server that is responsible for monitoring the other components of the system (*InfluxDB* sever) and a centralized server to deliver OTA updates (firmware-update server).

The IoT devices will run different versions *OpenWrt* [112], inclusive an *OpenWrt* version built specifically to be deliberately insecure, the *IoTGoat* [113].

OpenWrt is a *Linux* operating system targeting embedded devices where its main focus is consumer-grade routers. However, there is a large amount of IoT devices that use this firmware because it is commonly distributed as a Software Development Kit (SDK) from the System on a Chip (SoC) vendor to the product developer. The *Philips Hue Bridge 2.0*[*] is an example of this, which is a *ZigBee* gateway that uses *OpenWrt* as an operating system [114].

Each IoT device has a *telegraph* [115] agent that collects metrics about the device and sends them to a server that stores them. The data is stored in a *InfluxDB* database and can be accessed in the same server using *Grafana*.

Beyond that, IoT devices regularly connect to a server to check if there is any security update. If an update is available, the server will provide a SUIT manifest containing a link for the HTTP server, where the image can be downloaded.

Additionally, this network will have two more servers to host our testing system — one server dedicated to the FACT-core and the other for the rest of the components. As we mentioned earlier, the FACT-core does not need to be in the network that it is being analyzed. However, as our *hypervisor* server has available space to host this server, we opted to host the server on the same network. The other server will have two network interfaces, one for management and the other to sniff the network traffic, configured to be in a port mirroring. This server will run our services using docker.

To test our TLS plugin, we also put in the network a web server. This server will have a *nginx* service with several virtual hosts configured with different TLS configurations.

The information presented above is systematized in the Table 6.1.

[*]Philips Hue Bridge 2.0 is a smart switch that allows users to control all of their *Philips Hue* products through an app.

Name	OS	CPU	RAM	Purpose
AECSA	Ubuntu 20.04	4	4GB	This machine hosts the containers of our test system, using docker. It has two network interfaces, one for management and other to sniff the network packets.
FACT Core	Ubuntu 20.04	4	8GB	An installation of FACT core with the minimal requirements. As we mentioned earlier, this component could be in other network or in the cloud.
firmware-update	Alpine linux	1	768 MB	Server providing OTA updates to devices
InfluxDB	Ubuntu 20.04	2	2GB	This server collects metrics from the devices in the network
IoTGoat	OpenWRT	1	1GB	IoTGoat is a deliberately insecure OpenWRT machine created by OWASP, to exemplify common vulnerabilities of IoT devices.
openwrt-15.05	OpenWRT 15.05.0	1	256MB	Generic openwrt installation
openwrt-18.06	OpenWRT 18.06.0	1	256MB	Generic openwrt installation
openwrt-19.07	OpenWRT 19.07.2	1	256MB	Generic openwrt installation

TABLE 6.1: Devices on the network

6.2 Phase 1

The security policies influence the evaluation of the environment that we are assessing. Normally, during the certification process, a GST was already created when evaluating a device, and the evaluation will follow this security profile. To evaluate a device with our system, the GST should be converted to a common language accepted by our system and enable us to evaluate a device according to any GST.

In our case, we are only developing and testing an evaluation methodology, so we did not develop a GST. Thus, to test our evaluation methodology, we will need to choose a Technical Specification of Security Requirements and create our own profiles based on these requirements. Since we are only developing these profiles to evaluate our system, we will not follow all the certification phases of the ECSCF, but we will only develop the profile directly from the Technical Specification of Security Requirements and in the format accepted by our system.

In the following subsections, we will describe the Technical Specification of Security Requirements we choose and then develop the security profile and the associated compliance tests to each security requirement.

6.2.1 Technical Specification of Security Requirements

Before creating a security profile, we need a set of security requirements. According to the ECSCF, these requirements should be based on existing standards.

As a foundation for our security profile, we choose the *ETSI TS 103 645* because, when we plan this thesis, it was the only European standard for IoT, and it was already in use by a security certifications (BSI kitemark).

The *ETSI TS 103 645* is an Technical Specification (TS) aiming for the security of consumer IoT devices. The standard specifies high-level requirements for the actual security of a IoT device and the services associated with it. This document is divided into provisions. In the provisions, there are mandatory and non-mandatory security requirements. If any of the requirements is considered not applicable to the device, the justification must be recorded.

In total, there are 37 provisions divided into 13 groups. These provisions focus on the different IoT attack vectors, as hardware attacks maintain the data's privacy. The groups of provisions are written below:

- No universal default passwords
- Implement a means to manage reports of vulnerabilities
- Keep software updated
- Securely store credentials and security-sensitive data
- Communicate securely
- Minimize exposed attack surfaces
- Ensure software integrity
- Ensure the protection of personal data

- Make systems resilient to outages
- Examine system telemetry data
- Make it easy for consumers to delete personal data
- Make installation and maintenance of devices easy and Validate input data.

6.2.2 Profile development

To develop a security profile that could feed our system, we analyzed the different security requirements and designed the necessary test to verify if the requirement is fulfilled.

In the Appendix A, it is detailed our initial analyses, where we survey which requirements were mandatory to be implemented and if they can be tested with an automatic test.

We concluded that it was possible to automate 11 from the 37 proposed security requirements by the standard from this process. This means that only 30 percent of the requirements were possible to automate. However, these results are due to the number of requirements that are only possible to test with the device manufacturer's human interaction or inquiry. The requirements that were not possible to automate include all tests related to General Data Protection Regulation (GDPR), processes involving company policies (such as the device update policy), and the relation between the vendor and the user. These types of requirements will always require some user input to test.

To better explain this limitation, we can analyze one of these requirements. The requirement *4.12-2*, which was considered impossible to automate, states that "Consumers should also be provided with guidance on how to set up their device securely.". Without a manual analysis of the user's documentation, it is impossible to attest to this correct implementation requirement.

Based on the requirements that we considered to be able to automate, we develop 15 tests. The number of tests is superior to the number of security requirements because some were divided into multiple tests to evaluate their fulfillment properly. For instance, the requirement 4.2-3, which establishes that all software components in IoT devices should be updated, is divided into three tests: network vulnerabilities, software vulnerabilities, and vulnerabilities detected by static analysis of the firmware.

We will now describe the requirements that we can test with our system and the associated tests. We implemented all the tests using the external analysis plugin architecture.

On the other hand, the FACT core plugins, except the test number 10, were developed by the developers of FACT and reused in our system.

Provision 4.5-1 dictates that any security-sensitive information should be encrypted in transit. This security requirement was divided into two tests: the general protocol testing and TLS protocol testing.

The idea behind this division is to evaluate how secure is the communication, and with the TLS protocol testing, we can evaluate the settings used in the secure channel. These two tests were developed as an External analysis plugin.

- **Test 1 - General protocol testing**

  This test is executed with each network flow (flow means the source and destination IPs and network ports). It has a list of protocols considered insecure, and if the protocol identified in the flow matches any of this list, the plugin will log the situation.

  With this test, we can identify if the device is not secure and uses protocols that may transport security-sensitive information.

- **Test 2 - TLS protocol testing**

  This external analysis plugin is responsible for analyzing each TLS stream.

  The library that we used to develop our packet analysis does not support decoding protocols above the transport layer. So, as TLS is above the transport layer, the *gopacket* does not decode TLS packets [116]. To overcome this problem, we developed a solution using the existent *Go* lang native TLS implementation [117] and the *gopacket*'s *tcpassembly*. The *tcpassembly* will assemble the TCP stream and then, using the *Go* TLS implementation, it decodes TLS packets.

  Go is able to decode TLS packets, however this is done with internal components (*Go*'s unexported functions, variables and structs), so we are not able to use them. We tried to find ways to access these features, but the methods we found only allowed us to access unexported functions and not structs or variables [118, 119]. Thus, we were forced to copy the code of the TLS implementation into the plugin.

  This approach has a limitation. The *Go* TLS implementation only supports TLS 1.2 and 1.3, so if a device communicates with an older version of the protocol, the decoding of the protocol may fail.

The majority of the technical specifications do not define what is considered a secure connection. To do that, we will utilize the *NISTSP800-52REV.2* [120], which is the NIST guidelines for the configuration of TLS communications. All these best practices were represented in a *XML* file [*], which is parsed by the plugin and enable us to switch its settings easily. This policy's representation considers that if the observed setting does not appear on the list, it means that this setting is insecure. Additionally, NIST also defines some configurations as acceptable and others as recommended. To include this in our policy, there is a field representing if the configuration is recommended or not, and this will influence the moment of assigning points for this security requirement. If the configuration is acceptable, it will be assigned a partial score of this requirement's total score.

To test the implementation of these guidelines, we will test four things in the TLS stream: the security of the version of the TLS, the security of the agreement protocol, if the certificate being used is for the correct server, and finally, if the certificate is valid.

TLS 1.3 has a restriction for this test. During TLS 1.3 handshake, the TLS certificates are encrypted, so we cannot check its validity. However, we can still analyze the rest of the TLS specifications on the secure channel [121]. This problem affects every application that does this type of testing and forces applications to become more intrusive to test this type of information, such as transparent proxies to monitor TLS connections. With our goals for this project, this is not an option because to configure a TLS proxy, the device must trust our certificate authority, and to do this, we need privileged access to the device (which does not respect the *Requirement 2.a*).

Provision 4.6-5 mandates that software should run with the least necessary privileges. This requirement could be evaluated with two tests: check if the software is running with few privileged-users and exploit mitigations.

- **Test 3 - Init analysis**

  The Init analysis plugin detects files of auto-start services.

  To transform this plugin into a test, it can check if the services are running with few privileged-users. According to the auto-start system in use, we need to analyze the plugin results and identify if there is any form of user-defined in the service.

[*]`https://github.com/MrSuicideParrot/AECSA-analysis-plugins/blob/master/SSL/tlspolicy.xml`

This plugin identifies five types of auto-start services system: *SystemD*, *rc*, *initscript*, *UpStart* and *SysVInit*. From these systems, only *rc* does not have a standard way to run a program with a specific user.

Thus, when we analyze this plugin's results, and according to the type of auto-start service system, we consider that the security requirement is met if there is a service running with other users than *root*.

This approach has a limitation, as it does not detect if the software changes the *user id* when it is running. To overcome this limitation, there are two possibilities: having privileged access to the device and observing the *user id* with which the programs are running or analyzing software in search of *setuid* operations. Due to the *Requirement 2.a*, it is impossible to analyze the programs when they are running, so they can only develop a new plugin to search for *setuid* operations.

- **Test 4 - Exploit mitigations**

  Exploit mitigations plugin analyses each ELF binary and reports the state of each exploit mitigation technique, namely PIE, NX, RELRO, and Canary stack. Ideally, binaries should have all these exploit mitigations enable.

  To evaluate devices in this test, we assign a score according to the number of exploit mitigations implemented by the device. So, if the device implements all techniques, it will have the maximum score, and if the device only implements half of the techniques, it will have half of the maximum score.

  It is possible that, depending on the binary, there are different exploit mitigations implemented. So, we consider that a mitigation technique is implemented when the majority of binaries implement it.

Provision 4.5-2 defines that all cryptographic keys should be securely managed. This includes its operation and storage.

- **Test 5 - Crypto material - 4.5.2**

  The crypto material is a FACT plugin that searches the firmware for SSH, PGP, and SSL private keys. If we find any of these secret keys, we will consider that the security requirement 4.5.2 is not met.

Provision 4.3-1 dictates that all software components in consumer IoT devices should be updated. We divided the evaluation of this requirement into three separate tests: a network vulnerability scan, a software vulnerability scan, and a software vulnerability scan by static analysis.

- **Test 6 - Network Vulnerability Scan - 4.2-3.1**
  This test uses the builtin *OpenVas* scanner of our architecture to scan devices for known network vulnerabilities.

  The score of this requirement is assigned according to the number of vulnerabilities and their severity.

- **Test 7 - Software Vulnerability Scan - 4.2-3.2**
  Software Vulnerability Scan uses the *cve_lookup* plugin from FACT, which identifies software components in the firmware and searches for known CVE's.

  The score of this requirement is assigned according to the number of vulnerabilities and their severity.

- **Test 8 - Software Vulnerability Scan by static analysis - 4.2-3.3**
  Software Vulnerability Scan by static analysis uses the Common Weakness Enumeration (CWE) checker plugin from the FACT. This plugin performs to ELF binaries statically analysis and searches for weakness in the code (CWE). Internally, this plugin uses the Binary Analysis Platform [122], which currently supports ARM, x86/x64, PPC, and MIPS.

  CWE's do not have a score of severity associated with it because a weakness may not translate to a vulnerability, and beyond that, it is necessary to know the assets that may be compromised to evaluate its severity. This process needs human interaction.

  Therefore, we use this test to evaluate a device for unknown vulnerabilities, the existence of a CWE is a sign of poor code quality and possible vulnerabilities, and the requirement will be considered not met.

Provision 4.6-1 states that unused network ports should be closed to minimize exposed attack surface.

- **Test 9 - Nmap port scanner**
  Nmap port scanner is built-in in our architecture. To test this security requirement, we will scan the device for open ports and assign a score according to the number of exposed services. A device with more open ports will have the worst score in this test.

Provision 4.6-3 states that software should only be available if it is in use.

This type of statement is difficult to automate without knowledge about the behavior of the device. However, we can detect the software present in the device and evaluate the number of installed software.

- **Test 10 - Software components**

  The software components plugin identifies software that is installed in the device. For instance, if a device has debug tools in its firmware, the number of installed software will be high, and this requirement's score will be lower.

Provision 4.1-1 states that IoT devices passwords must be unique and without universal default passwords. We created two tests for this requirement: one that tries default password against exposed services and others that tries to extract credentials from captured firmware.

- **Test 11 - User and passwords - 4.1-1**

  FACT user and passwords plugin searches the firmware by Unix, and *httpd* password files and tries to crack them with known credentials. If we can crack any of the credentials found, this requirement will be considered not fulfilled.

  Test 12 - Default credentials scanner - 4.1-1

  The default credentials scanner is the scanner mode of the *OpenVas* vulnerability scanner that it is built-in in our system. This scan attempts to authenticate itself with default credentials in the exposed services. If this process is successful, we will consider the requirement 4.1-1 is not fulfilled.

Provision 4.4-1 defines that security-sensitive data should be stored securely.

- **Test 13 - Search secrets** Search secrets is a plugin that we develop to search the firmware files for API secrets. FACT core offers an analysis plugin sub-type called *YaraPlugin*. This type of plugin uses *YARA* pattern matching [123] to identify patterns in the firmware files.

  The development plan for these plugins was to search for existent collections of API tokens regex rules, rewrite them into *YARA* rules, and then develop our plugin using these rules. The project that we use to extract the regex rules of API tokens was the *gitleaks* [124], which is a program used to scan git repositories for leaks of

API tokens. We also analyze other projects, but we concluded that everyone was using the same set of regex expressions [125].

The *YARA* rules and the plugin that we developed are available on Github[126, 127].

Provision 4.3-7 states that when a device is updated, it must be delivered over a secure channel.

- **Test 14 - Firmware Suit**

  Firmware Suit is a test built-in in our system that checks if OTA updates follow the SUIT standard.

Provision 4.13-1 defines that user input must be proper validated. A way to automate these tests is fuzzing the different services of the device.

There are different types of fuzzing, depending on the access to the device. As we do not have privileged access to the device, we can infer some information from the port scanner and DPI.

This information allows us to approach this problem with two techniques, grammar-based fuzzing and mutation fuzzing [128].

We can use the port scanner results to identify the ports open in the device and fuzzing this ports with grammar-based fuzzing according to the identified protocol. The other option is to capture real traffic from the device and perform mutation fuzzing based on this network traffic.

Regardless, fuzzing in a production environment is a risky operation. It can compromise the reliability of the system and also consume bandwidth necessary for other services. Because of this, the execution of a test like this could not be totally automated. This needs to be controlled or configured by an operator not to compromise the IoT system's reliability.

Unfortunately, due to the time constraint of this thesis, we did not develop this test.

To summarize, the Table 6.2 lists each test, the security requirement that it aims, and each type of plugin it was implemented.

We develop two security profiles with all these requirements and security tests: the default profile and the server profile (Appendix B). The default profile has all the requirements and tests mentioned above. The server profile aims servers on the same network of the IoT devices and has no requirements regarding firmware and OTA updates.

Test number	Name	Requirements	Plugin type
1	General protocol testing	4.5-1.1	External analysis
2	TLS protocol testing	4.5-1.2	External analysis
3	Init analysis	4.6-5.1	FACT core
4	Crypto material	4.5-2	FACT core
5	CVE lookup	4.2-3.2	FACT core
6	CWE checker	4.2-3.3	FACT core
7	Exploit mitigations	4.6-5.2	FACT core
8	Software components	4.6-3	FACT core
9	User and passwords	4.1-1	FACT core
10	Search secrets	4.4-1	FACT core
11	Firmware suit	4.3-7	Built-in test
12	Network vulnerability scanner	4.2-3.1	Built-in test
13	Default credentials scanner	4.1-1	Built-in test
14	Nmap port scanner	4.6-1	Built-in test
15	Fuzzing	4.13-1	-

TABLE 6.2: Relation between tests and security requirements

6.3 Phase 2

The goal of this phase is to verify the effectiveness of each test and the overall system. To achieve this, we evaluate each test (Unit testing) and the system's overall effectiveness (System testing) individually.

6.3.1 Unit testing

Unit testing ensures that our tests have been implemented correctly. To assess these tests, IoT devices on the network must create specific traffic to verify that the test detects non-compliant behavior. Thus, our tests will need to remotely control IoT devices to initiate vulnerable behaviors and then verify that they have been detected.

We created a script that uses *Ansible* [129] to control each IoT device in the test environment and makes SQL queries to the database to check if the tests were detected correctly.

Ansible is a software of provisioning, configuration management, and application-deployment that uses existing remote administration protocols, such as SSH, to run a script programmatically on one or more machines. This software was used to execute commands remotely more easily, as it allows us to abstract the configuration of a connection with a simple alias and execute several commands simultaneously on different machines. *Ansible* requires an SSH server and a Python installation on each machine. By default, each *OpenWrt* machine has an SSH server, and it is possible to install Python from the *OpenWrt* repositories. Therefore, the process of installing and configuring *Ansible* was simple.

The testing procedure is as follows: First, the database is cleared of all previous events, then a command is executed on the device IoT, which causes the device to behave in an insecure manner. Finally, we query the database to verify that this event was correctly identified and recorded.

The Unit test script is available on the project repository. In total, there are 16 unit tests, at least one for each security requirement test (test 15 does not have a Unit Test because it has not been implemented). Tests 2 and 11 were evaluated with more than one test because it was necessary to ensure that they did not identify secure communications as unsafe.

Our system passed all tests successfully. However, during the results analysis, we discovered a limitation mentioned in other certification schemes, namely the ARMOUR certification.

The quality of the CVE report limits the effectiveness of our automated vulnerability scan. If the CVE details were filled incorrectly, it would compromise our scan.

An example that we observe in our tests was CVE-2017-3209. This vulnerability affects a specific drone model with an insecure FTP server, allowing anonymous files as root. Our vulnerability scan identifies this on all IoT devices in our test environment, because those devices have *busybox* in their firmware, and the vulnerability reporting states that affects all the installations of *busybox*, regardless of the version installed.

This type of mistake can be detected with a human review of the test results. However, for systems that intend to be fully automated, this is not a solution.

6.3.2 System testing

System testing has the goal of testing the overall effectiveness of our system. This test lasted a week, in which our system constantly monitored the test environment.

During this test, we turned off the *telegraph* agent, as we intended to evaluate the IoT devices with their default configurations.

We use two security profiles: the *default profile* and the *server profile*. The virtual machines running different versions of *OpenWrt* were evaluated with the *default profile*. The remaining ones used the *server profile*. During these tests, we simulate a secure OTA update to each *OpenWrt* device to analyze their firmware.

The four *OpenWrt* devices have different versions of the operating system, and without any security patch, it is expected that more recent systems have a better score than older ones. Moreover, *IoT Goat* is based on *OpenWrt* 18, but was created to be insecure. Thus, it should have the worst security score.

Our system assigns a score of 74.35 to *openwrt-19.07*, 71.53 to *openwrt-18.06*, 67.27 to *openwrt-15.05*, and 61.12 to *IoTGoat*.

The results were what we were expecting; more recent installations have a better score than older ones.

By analyzing the reports, we noticed that *openwrt-18.06* and *openwrt-19.07* have similar security reports. There are different vulnerabilities registered in each operating system (Test 5), due to different software versions (Test 8). Additionally, the *openwrt-18.06* registered a CWE in one of their binaries. The rest of the report was equivalent in the two machines.

The default credentials on *openwrt-15.05* (Test 13) were detected. This was predictable because before the version *openwrt-18.06*, all devices have default credentials, and after that version, the user is required to define a unique password to set up the device (Test 13).

IoTGoat was the device with the worst score. This device has the biggest attack surface with 5 exposed services (Test 14), multiple vulnerabilities (Test 5 and 6), and its firmware had sensible information (Test 10).

We also confirmed that it was possible to compare the different devices (Requirement 1.d).

As we mentioned above, the devices that were not based on IoT devices were evaluated with a security policy that does not have firmware related tests. The system was able to evaluate the devices according to the assigned security policy. Moreover, it was possible to change the security policy on the fly (Requirement 1.a).

During this phase, we tested the effectiveness of the continuous assessment. Initially, the servers that we were running did not have any security vulnerability. During the tests, a new vulnerability for the SSH server was discovered (CVE-2020-15778), the Test 12 discovered multiple machines affected by this vulnerability, and the score of these servers was updated (Requirement 3.b).

6.4 Phase 3

During the development of our initial policy, a new standard emerged, the *ETSI EN 303 645* [130]. The *ETSI EN 303 645* was created at the request of the European Commission. This new European Standard (EN) is based on the *ETSI TS 103 645* and shares most of its security requirements but in greater detail.

The biggest difference between these two documents is their impact on the governance of the EU countries. An EN needs to be transposed to a national standard, and the *ETSI TS 103 645* does not require this [131, 132].

The *ETSI EN 303 645* was considered by the ENISA as the standard that fulfills the requirements of the Cybersecurity act for the application on IoT environments [26]. Due to this, we tested the modular capability of our framework by adapting our existing security policy to a new Technical Specification of Security Requirements, the *ETSI EN 303 645.*

ETSI EN 303 645 shares the same provision groups of the *ETSI TS 103 645*. However, it has in total 60 provision, more 23 than the *ETSI TS 103 645*. From these 60 provisions, we can automate 18, which means 30 percent of all requirements (Appendix C).

To evaluate a device according to the requirements of the *ETSI EN 303 645*, it would be necessary to adapt 3 of our existent tests and create a new test for one of the requirements.

Test 8 needs to check the number of software installed to verify if there is any installation of a known implementation of cryptographic operations (Requirement 4.5.2).

Test 13 checks for available default credentials. To verify the compliance with the Requirement 4.1-5, it needs to also check for protection against brute force attacks.

Test 14 that performs a port scan of the device, needs to evaluate the amount of security-relevant information that can be retrieved from this type of scan. The requirement 4.6-2 states that a "device should minimize the unauthenticated exposure of security-relevant information".

Finally, it is necessary to create a new test to evaluate the technologies used in the authentication mechanisms to ensure they use the best practice cryptography according to the technology, risk, and usage (Requirement 4.1-3).

The *ETSI EN 303 645* is an extension of the E*TSI TS 103 645*. It approaches the same topics of the *ETSI TS 103 645*, but now in a more detailed way, with descriptions and examples. This way, there are fewer doubts about the implementation of the requirements.

We were able to plan a test methodology capable of partially evaluating compliance with the *ETSI EN 303 645*. This task was made easier because *ETSI EN 303 645* shares much of its essence with *ETSI TS 103 645*. Nevertheless, if we needed to adapt our framework to another set of security requirements, many of the tests developed could be adapted to other standards.

6.5 Reflection on the results

Throughout this section, we evaluated the testing framework that we developed against the initial goals that we set. We were able to accomplish all the goals set for this project. However, we also identify some challenges that limit this type of assessment.

As already mentioned in the previous work, an automatic vulnerability assessment's success depends on the vulnerability database's quality. Existent databases are not prepared to be run with fully automated systems. Sometimes, information is missing or incorrect about the vulnerabilities, which compromises an automatic scan's effectiveness. Moreover, there are no databases focused on IoT vulnerabilities.

The most restrictive requirement that we set on our project was to evaluate the device without the product vendor's support. This condition did not allow us to evaluate some of the security requirements with precision that we would like to have (Test 3). This is the biggest disadvantage that a vendor-agnostic evaluation has when compared with other types of assessments.

Finally, the academic community is currently working on automated security assessments, and automated security assessments during a certification process are a desirable possibility for certifications [12]. However, due to the way current technical security specifications are constructed, there is no possibility to fully evaluate a device according to a standard without some manual assessment.

Chapter 7

Conclusion

Currently, there is a need for automatic certifications and assessments for IoT environments. Existing certifications are described as not being agile or scalable, and classic security assessments are carried out periodically. Therefore, the dynamic nature of IoT undermines the effectiveness of these methods. Besides, these limitations lead to a lack of certifications that meet the IoT environment needs.

Our work addresses these challenges by analyzing how to create an automated compliance assessment of a security standard and mitigate these issues.

7.1 Research Summary

Before creating our proposal, we started by analyzing the needs of IoT certifications and the current work on automatic security testing. With that, we defined the requirements to develop an automated assessment that could be used for a certification process. Besides, we also realize that there are still IoT application domains without a certification that meets all their needs. Thus, we decided to focus our approach on one of these domains: environments that require evaluating an IoT environment in its specific context.

With these requirements in mind, we developed our system with three types of tests: interface testing, network testing, and system testing. The system was designed with existing open source security tools and adapted to create a modular automatic assessment system that assesses different security requirements.

To test our system's effectiveness, we set up a virtualized test environment and assessed its compliance with *ETSI TS 103 645*. From that test run, we were able to automate the assessment of thirty percent of security requirements.

7.2 Current Limitations

This percentage of compliance is due to the goals we set for our project. We expected that our system would not require any privileged access to the device and be automated. However, due to the way current technical security specifications are built, there is no possibility to fully evaluate a device against a standard without any manual evaluation.

Besides, during testing, we also noticed that many vulnerability databases have missing or incorrect information, compromising a fully automated vulnerability assessment's effectiveness.

Therefore, we were able to automate a conformity assessment of a technical security specification successfully. However, our results were restricted due to the current way in which these specifications are built. The academic community is working on automated security assessments, but there is a need to connect automated security assessments to technical security specifications. Technical security specifications are not prepared to be evaluated by automatic assessments. Thus, the applicability of these new assessment techniques is limited until this gap is filled.

7.3 Future Work

During this thesis's development, some tasks were not possible to accomplish due to time constraints, and we were also unable to explore some of the ideas that emerged. Thus, in this chapter, we describe the tasks, improvements, and ideas that have been left for the future due to lack of time.

In evaluating our automatic assessment system, we state that fuzzing the device's interfaces allows us to test whether the user's input is appropriately validated. However, we were not able to develop this test. Although this test only applies to a single security requirement, fuzzing tests, as we mentioned in Chapter 3, are considered one of the solutions to automate tests in IoT. Fuzzing allows us to discover unknown vulnerabilities on a device without the need for manual work; simultaneously, the device's resilience is evaluated. Therefore, developing a fuzzing test component in our system would improve the Technical Security Requirement Specification coverage.

We noticed that many technical security requirement specifications require that any security-sensitive information should be encrypted in transit. Currently, our implementation has a list of secure and insecure protocols to transport information. However, a device can use custom protocols, and it is necessary to evaluate them, namely if they carry any information and whether these communications are encrypted. One way to analyze these protocols is to use compression and entropy analysis to verify any information retrieved from these communications [133]. The implementation of this type of test would allow us to evaluate unknown protocols more correctly.

The firmware analysis of our system depends on the detection and occurrence of OTA updates. Many device vendors make firmware available on their websites to update devices manually. Our system's firmware analysis component can be expanded to allow manual submission of firmware files for analysis.

The web application developed to control this evaluation system does not have any security hardening, there is no authentication in the report event endpoint, and there is no granular access control in the web application. The web application needs to be improved with authentication between external-analysis plugins and web applications and multiple users' permissions.

In addition to the improvements proposed above, which are related to our current goals, we also want to explore this assessment system with less restrictive objectives to increase the number of security requirements that we can test. Instead of having a fully automated testing system, we can better evaluate the devices if we create a hybrid assessment technique with human-in-the-loop.

7.4 Conclusions

During our research, we addressed the main limitations of existing certifications. To bridge the gap between the heterogeneity of IoT environments and the automation of security testing, we contributed to a different security testing approach with an automated compliance assessment of a standard of security.

Our proposed solution called AECSA is an automatic assessment system that updates the assessment results over time, according to changes in the IoT environment, and can be adapted to multiple technical specifications of security requirements.

This approach mitigates the problems identified as limitations for standard security assessment methods when applied to IoT environments. The mitigated issues were also the reason why some IoT environments have a certification that inadequately meets their requirements.

In brief, AECSA bridges the gap by providing a security assessment approach that helps certifications assess evolving IoT environments and require constant monitoring.

Appendix A

ETSI TS 103 645 - Analysis for automation

ETSI TS 103 645	Type of test	Mandatory	Automatic test
4.1-1	Interface testing	Yes	Yes
4.2-1	Procedure testing	Yes	No
4.2-2	Procedure testing	Yes	No
4.2-3	Procedure testing	No	Yes
4.3-1	Interface testing	No	Yes
4.3-2	Procedure testing	No	No
4.3-3	Procedure testing	Yes	No
4.3-4	Procedure testing	Yes	No
4.3-5	Procedure testing	No	No
4.3-6	Procedure testing	No	No
4.3-7	Network/System testing	No	Yes
4.3-8	Procedure testing	No	No
4.3-9	Procedure testing	No	No
4.4-1	System testing	Yes	Yes
4.5-1	Network testing	No	Yes
4.5-2	System testing	No	Yes
4.6-1	Interface testing	No	Yes

4.6-2	Interface testing	No	No
4.6-3	System testing	No	Yes
4.6-4	System testing	No	No
4.6-5	System testing	No	Yes
4.7-1	Procedure testing	No	No
4.7-2	Procedure testing	No	No
4.8-1	Procedure testing	Yes	No
4.8-2	Procedure testing	Yes	No
4.8-3	Procedure testing	Yes	No
4.9-1	Procedure testing	No	No
4.9-2	Procedure testing	No	No
4.9-3	Procedure testing	No	No
4.10-1	Procedure testing	No	No
4.10-2	Procedure testing	No	No
4.10-3	Procedure testing	Yes	No
4.11-1	Procedure testing	No	No
4.11-2	Procedure testing	No	No
4.11-3	Procedure testing	No	No
4.12-1	Procedure testing	No	No
4.13-1	Interface testing	Yes	Yes

Appendix B

Security profiles

In this appendix, we will present the security profiles that were developed during this thesis.

B.1 Default profile

```
<?xml version="1.0" encoding="utf-8" ?>

<policy>
    <metadata>
        <name>Default Policy</name>
        <uuid>1</uuid>
    </metadata>
    <category>
        Network
        <check standard_id="4.1-1" points="1">
            No default credentials
        </check>
        <check standard_id="4.2-3.1" points="1">
            Network vulnerabilities
            <vuln multiplier="0">Critical</vuln>
            <vuln multiplier="0.25">High</vuln>
            <vuln multiplier="0.50">Medium</vuln>
            <vuln multiplier="0.75">Low</vuln>

        </check>
        <check standard_id="4.3-7" points="1">
            Secure OTA
        </check>
        <check standard_id="4.5-1" points="1">Secure communication
```

```
            <check standard_id="4.5-1.1" points="1">Use of secure protocols</check>
            <if standard_id="4.5-1.2">
                TLS checks
                <check standard_id="4.5-1.2.1" points="1">TLS version</check>
                <check standard_id="4.5-1.2.2" points="1">Agreement algorithm</check>
                <check standard_id="4.5-1.2.3" points="1">Valid domain</check>
                <check standard_id="4.5-1.2.4" points="1">Valid certificate</check>
            </if>
        </check>
        <check standard_id="4.6-1" points="1">
            Reduced number of exposed services
            <range multiplier="1" >0,2</range>
            <range multiplier="0.70" >3,4</range>
            <range multiplier="0.30" >5,6</range>
            <range multiplier="0" >6+</range>
        </check>
    </category>
    <category>
        Software
        <check standard_id="4.2-3.2"  points="1">
            Software vulnerabilities firmware
            <vuln multiplier="0">Critical</vuln>
            <vuln multiplier="0.25">High</vuln>
            <vuln multiplier="0.50">Medium</vuln>
            <vuln multiplier="0.75">Low</vuln>
        </check>
        <check standard_id="4.2-3.3"  points="1">
            Software vulnerabilities by static analysis
            <range multiplier="1" >0=</range>
            <range multiplier="0.70" >1=</range>
            <range multiplier="0" >2+</range>
        </check>
        <check standard_id="4.6-3"  points="1">
            Reduce installed software
            <range multiplier="1" >5-</range>
            <range multiplier="0.70" >6,8</range>
            <range multiplier="0" >9+</range>
        </check>
        <check standard_id="4.6-5" points="1">
            Running software with reduced privileges
            <check standard_id="4.6-5.1" points="1">Low privilige user on init processes</check>
            <check standard_id="4.6-5.2" points="1">Exploit mitigations in place</check>
        </check>
    </category>
    <category>
        Firmware
        <check standard_id="4.4-1"  points="1">
```

```
            Sensible information
        </check>
        <check standard_id="4.5-2"  points="1">
            Cryptographic keys
        </check>
    </category>
</policy>
```

B.2 Server profile

```
<?xml version="1.0" encoding="utf-8" ?>

<policy>
    <metadata>
        <name>Server Policy</name>
        <uuid>2</uuid>
    </metadata>
    <category>
        Network
        <check standard_id="4.1-1" points="1">
            No default credentials
        </check>
        <check standard_id="4.2-3.1" points="1">
            Network vulnerabilities
            <vuln multiplier="0">Critical</vuln>
            <vuln multiplier="0.25">High</vuln>
            <vuln multiplier="0.50">Medium</vuln>
            <vuln multiplier="0.75">Low</vuln>
        </check>
        <check standard_id="4.5-1" points="1">Secure communication
            <check standard_id="4.5-1.1" points="1">Use of secure protocols</check>
            <if standard_id="4.5-1.2">
                TLS checks
                <check standard_id="4.5-1.2.1" points="1">TLS version</check>
                <check standard_id="4.5-1.2.2" points="1">Agreement algorithm</check>
                <check standard_id="4.5-1.2.3" points="1">Valid domain</check>
                <check standard_id="4.5-1.2.4" points="1">Valid certificate</check>
            </if>
        </check>
        <check standard_id="4.6-1" points="1">
            Reduced number of exposed services
            <range multiplier="1" >0,2</range>
            <range multiplier="0.70" >3,4</range>
            <range multiplier="0.30" >5,6</range>
            <range multiplier="0" >6+</range>
        </check>
    </category>
</policy>
```

Appendix C

ETSI EN 303 645 - Analysis for automation

EN 303 645	Type of test	Mandatory	Automatic test
4.1-1	Interface testing	Yes	Yes
4.1-2	Procedure testing	Yes	No
4.1-3	Interface testing/Network testing	Yes	Yes
4.1-4	Procedure testing	No	No
4.1-5	Interface testing	Yes	Yes
4.2-1	Procedure testing	Yes	No
4.2-2	Procedure testing	No	No
4.2-3	Procedure testing	No	No
4.3-1	Procedure testing	No	Yes
4.3-2	Network/System testing	Yes	Yes
4.3-3	Procedure testing	Yes	No
4.3-4	Procedure testing	Yes	No
4.3-5	Procedure testing	Yes	No
4.3-6	Procedure testing	No	No
4.3-7	Procedure testing	No	No
4.3-8	Procedure testing	Yes	No
4.3-9	Procedure testing	No	No

4.3-10	Procedure testing	No	No
4.3-11	Procedure testing	No	No
4.3-12	Procedure testing	No	No
4.3-13	Procedure testing	No	No
4.3-14	Procedure testing	No	No
4.4-1	System testing	Yes	Yes
4.4.2	System testing	Yes	No
4.4-3	System testing	Yes	Yes
4.4-4	System testing	Yes	No
4.5-1	Network testing	Yes	Yes
4.5-2	Network testing	No	Yes
4.5-3	Procedure testing	No	No
4.5-4	Interface testing	No	Yes
4.5-5	Procedure testing	Yes	No
4.5-6	Network testing	No	Yes
4.5-7	Network testing	Yes	Yes
4.5-8	Procedure testing	No	No
4.6-1	Interface testing	Yes	Yes
4.6-2	Interface testing	No	Yes
4.6-3	Interface testing	Yes	No
4.6-4	System testing	Yes	Yes
4.6-5	System testing	Yes	No
4.6-6	System testing	Yes	Yes
4.7-1	System testing	Yes	No
4.7-2	Procedure testing	Yes	No
4.8-1	Procedure testing	Yes	No
4.8-2	Procedure testing	Yes	No
4.8-3	Procedure testing	Yes	No
4.8-4	Network testing	No	No
4.8-5	Network testing	Yes	No
4.9-1	Procedure testing	No	No
4.9-2	Procedure testing	No	No

4.9-3	Procedure testing	No	No
4.10-1	Procedure testing	No	No
4.10-2	Procedure testing	No	No
4.10-3	Procedure testing	Yes	No
4.11-1	Procedure testing	Yes	No
4.11-2	Procedure testing	No	No
4.11-3	Procedure testing	No	No
4.11-4	Procedure testing	No	No
4.12-1	Procedure testing	No	No
4.12-2	Procedure testing	No	No
4.12-3	Procedure testing	No	No
4.13-1	Interface testing	Yes	Yes

Acronyms

API Application Programming Interface 53, 55, 57

BDI belief–desire–intention 37

CAB Conformity Assessment Body 7, 9, 12, 27

CVE Common Vulnerabilities and Exposures 38, 77

CWE Common Weakness Enumeration 73, 78

DPI Deep Packet Inspection 44, 50, 51, 53, 75

e-IoT-SCS Eurosmart IoT Security Certification Scheme 27, 29–33

ECSC European Cyber Security Certificate 12

ECSCF European Cyber Security Certification Framework 2, 3, 7–12, 20, 21, 23, 27, 30, 32, 33, 43, 45, 46, 60, 67, 68

ECSO European Cyber Security Organization 8, 23

EN European Standard 79

ENISA European Union Agency for Cybersecurity 3, 4, 7, 8, 10, 42, 45, 65, 79

EU European Union 5, 6, 8, 79

GDPR General Data Protection Regulation 69

GPP Generalized Protection Profile 12

GST Generalized Security Target 12, 67

HTTP Hypertext Transfer Protocol 48, 55

IETF Internet Engineering Task Force 58

IoT Internet of Things 1–7, 13–21, 23, 26–30, 33–40, 42, 46–48, 58, 65, 66, 68, 69, 72, 74–84

NASL Nessus Attack Scripting Language 57

NIST National Institute of Standards and Technology 26, 71

OSP Open Scanner Protocol 57

OTA Over-the-air 47, 48, 58, 66, 75, 77, 83

PIA Privacy Impact Analysis 17, 30, 31, 33

SDK Software Development Kit 66

SoC System on a Chip 66

SUIT Software Updates for Internet of Things 58, 66, 75

TCP Transmission Control Protocol 55, 70

TLS Transport Layer Security 26, 57, 61, 66, 70, 71

TS Technical Specification 68

WSGI Web Server Gateway Interface 61

Bibliography

[1] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the Internet of Things (IoT)," *IEEE Internet Initiative*, vol. 1, pp. 1–86, 2015. [Cited on page 1.]

[2] P. Kess, H. Kropsu-Vehkaperä *et al.*, "Standardization with IoT (Internet-of-Things)," in *Managing Innovation and Diversity in Knowledge Society Through Turbulent Time: Proceedings of the MakeLearn and TIIM Joint International Conference 2016*. ToKnowPress, 2016, pp. 1069–1076. [Cited on page 1.]

[3] O. Logvinov, B. Kraemer, C. Adams, J. Heiles, G. Stuebing, M. Nielsen, and B. Mancuso, "Standard for an architectural framework for the Internet of Things (IoT) ieee p2413," 2016. [Cited on page 1.]

[4] J. Voas, "Networks of 'things'," *NIST Special Publication*, vol. 800, no. 183, pp. 800–183, 2016. [Cited on page 1.]

[5] T. ETSI, "102 689 v1. 1.1,"," *Machine-to-Machine communications (M2M)*, pp. 1–34, 2010. [Cited on page 1.]

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013. [Cited on pages 1 and 14.]

[7] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim, and S. R. Chaudhry, "IoT architecture challenges and issues: Lack of standardization," in *2016 Future Technologies Conference (FTC)*. IEEE, 2016, pp. 731–738. [Cited on page 1.]

[8] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Internet of Things (IoT): Taxonomy of security attacks," in *2016 3rd International Conference on Electronic Design (ICED)*. IEEE, 2016, pp. 321–326. [Cited on pages 1 and 2.]

[9] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010. [Cited on pages 1 and 14.]

[10] S. S. I. Samuel, "A review of connectivity challenges in IoT-smart home," in *2016 3rd MEC International conference on big data and smart city (ICBDSC)*. IEEE, 2016, pp. 1–4. [Cited on page 2.]

[11] S. Deshmukh and S. Sonavane, "Security protocols for Internet of Things: A survey," in *2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2)*. IEEE, 2017, pp. 71–74. [Cited on page 2.]

[12] J. R. Nurse, S. Creese, and D. De Roure, "Security risk assessment in Internet of Things systems," *IT Professional*, vol. 19, no. 5, pp. 20–26, 2017. [Cited on pages 2, 3, 7, 18, 23, 35, and 80.]

[13] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "IoT security: ongoing challenges and research opportunities," in *2014 IEEE 7th international conference on service-oriented computing and applications*. IEEE, 2014, pp. 230–234. [Cited on page 2.]

[14] J. Hernández-Ramos, J. Martinez, V. Savarino, M. Angelini, V. Napolitano, A. Skarmeta, and G. Baldini, "Security and Privacy in Internet of Things-Enabled Smart Cities: Challenges and Future Directions," *IEEE Security and Privacy Magazine*, vol. PP, 08 2020. [Cited on page 2.]

[15] J. Hearn, "Does the common criteria paradigm have a future?[security and privacy]," *IEEE Security & Privacy*, vol. 2, no. 1, pp. 64–65, 2004. [Cited on page 2.]

[16] C. W. Axelrod, "Enforcing security, safety and privacy for the Internet of Things," in *2015 Long Island Systems, Applications and Technology*. IEEE, 2015, pp. 1–6. [Cited on pages 2 and 17.]

[17] J. Voas and P. A. Laplante, "IoT's certification quagmire," *Computer*, vol. 51, no. 4, pp. 86–89, 2018. [Cited on page 2.]

[18] H. Badran, "IoT Security and Consumer Trust," in *Proceedings of the 20th Annual International Conference on Digital Government Research*, ser. dg.o 2019. New York, NY, USA: ACM, 2019, pp. 133–140. [Online]. Available: http://doi.acm.org/10.1145/3325112.3325234 [Cited on page 2.]

[19] Legroju, "The eu cybersecurity act," Jun 2019. [Online]. Available: https://ec.europa.eu/digital-single-market/en/eu-cybersecurity-act [Cited on page 2.]

[20] E. C. S. Organisation, "European Cyber Security Certification," Tech. Rep., Dec 2017. [Online]. Available: https://www.ecs-org.eu/documents/publications/5a3112ec2c891.pdf [Cited on pages 3, 8, 11, 17, 46, and 60.]

[21] J. Voas and P. A. Laplante, "The IoT Blame Game," *Computer*, vol. 50, no. 6, pp. 69–73, 2017. [Cited on pages 3 and 42.]

[22] R. S. Ross, S. W. Katzke, and L. A. Johnson, "Minimum security requirements for federal information and information systems," Tech. Rep., 2006. [Cited on page 6.]

[23] S. Górniak, R. Atoui, J. Fernandez, J.-P. Quemard, and M. Schaffer, "Standardisation in support of the cybersecurity certification," Dec 2019. [Online]. Available: https://www.enisa.europa.eu/publications/recommendations-for-european-standardisation-in-relation-to-csa-i [Cited on pages 7, 8, 9, 10, 12, and 46.]

[24] R. Atoui. (2019, jun) Iot device certification scheme. [Online]. Available: https://www.eurosmart.com/wp-content/uploads/2019/06/e-IoT-SCS-Eurosmart_IoT_Device_Certification_v1.0_RELEASE.pdf [Cited on pages 7 and 27.]

[25] E. C. S. Organisation, "Overview of existing Cybersecurity standards and certification schemesv2," Tech. Rep., Dec 2017. [Online]. Available: https://www.ecs-org.eu/documents/uploads/updated-sota.pdf [Cited on page 7.]

[26] I. Barreira, H. Dettmer, M. Masi, L. O. Echevarria, and A. Sfakianakis, "Standards supporting certification," Dec 2019. [Online]. Available: https://www.enisa.europa.eu/publications/recommendations-for-european-standardisation-in-relation-to-csa-ii [Cited on pages 8 and 79.]

[27] Council of European Union, "Regulation (eu) 2019/881 of the european parliament and of the council of 17 april 2019," 2019. [Online]. Available: https://bit.ly/3jffXiW [Cited on pages 8 and 9.]

[28] ISO/IEC, "ISO/IEC 15408-1: Information technology — Security techniques — Evaluation criteria for IT security," Tech. Rep., Dec 2009. [Cited on pages 10, 12, and 40.]

[29] P. P. Ray, "A survey on Internet of Things architectures," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 3, pp. 291–319, 2018. [Cited on page 14.]

[30] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, "Technical guide to information security testing and assessment," *NIST Special Publication*, vol. 800, no. 115, pp. 2–25, 2008. [Cited on pages 15 and 16.]

[31] G. Purdy, "Iso 31000: 2009—setting a new standard for risk management," *Risk Analysis: An International Journal*, vol. 30, no. 6, pp. 881–886, 2010. [Cited on page 16.]

[32] Y. Klochkov, S. Odinokov, E. Klochkova, M. Ostapenko, and A. Volgina, "Development of certification model," in *2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. IEEE, 2016, pp. 120–122. [Cited on page 16.]

[33] A. Jacobsson, M. Boldt, and B. Carlsson, "A risk analysis of a smart home automation system," *Future Generation Computer Systems*, vol. 56, pp. 719–733, 2016. [Cited on pages 16 and 17.]

[34] C.-K. Chen, Z.-K. Zhang, S.-H. Lee, and S. Shieh, "Penetration testing in the IoT age," *Computer*, vol. 51, no. 4, pp. 82–85, 2018. [Cited on pages 16 and 35.]

[35] O. Memo, "M-03-22," *OMB Guidance for Implementing the Privacy Provisions of the E-Government Act of 2002*. [Cited on page 17.]

[36] H. Lin and N. Bergmann, "IoT privacy and security challenges for smart home environments," *Information*, vol. 7, no. 3, p. 44, 2016. [Cited on page 18.]

[37] A. Jacobsson and P. Davidsson, "Towards a model of privacy and security for smart homes," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 727–732. [Cited on page 19.]

[38] P. Gope and T. Hwang, "BSN-Care: A secure IoT-based modern healthcare system using body sensor network," *IEEE Sensors Journal*, vol. 16, no. 5, pp. 1368–1376, 2015. [Cited on page 19.]

[39] A. Act, "Health insurance portability and accountability act of 1996," *Public law*, vol. 104, p. 191, 1996. [Cited on page 19.]

[40] S. T. U. Shah, H. Yar, I. Khan, M. Ikram, and H. Khan, "Internet of Things-Based Healthcare: Recent Advances and Challenges," in *Applications of Intelligent Technologies in Healthcare*. Springer, 2019, pp. 153–162. [Cited on page 19.]

[41] A. Bartoli, J. Hernández-Serrano, M. Soriano, M. Dohler, A. Kountouris, and D. Barthel, "Security and privacy in your smart city," in *Proceedings of the Barcelona smart cities congress*, vol. 292, 2011, pp. 1–6. [Cited on page 20.]

[42] A. Q. Rodriguez, B. B. AS, M. Menon, S. Ziegler, A. M. P. H. AS, E. K. DG, and S. Bianchi, "Dynamic Security and Privacy Seal Model Analysis." [Cited on pages 21 and 22.]

[43] S. N. Matheu, J. L. Hernandez-Ramos, and A. F. Skarmeta, "Toward a Cybersecurity Certification Framework for the Internet of Things," *IEEE Security & Privacy*, vol. 17, no. 3, pp. 66–76, 2019. [Cited on pages 21 and 23.]

[44] I. Labs, "Internet of Things (IoT) Security Testing Framework," ICSA Labs website, ICSA Labs, Tech. Rep., Octo 2016. [Online]. Available: https://www.icsalabs.com/sites/default/files/body_images/ICSALABS_IoT_reqts_framework_v2.0_161026.pdf [Cited on pages 21 and 24.]

[45] "News: Announcing ul 2900 outlines," acessed: 2019-11-27. [Online]. Available: https://ulstandards.ul.com/downloads/news-announcing-ul-2900-outlines/ [Cited on pages 21 and 24.]

[46] "Testing and certification for IoT connected devices." [Online]. Available: https://www.bsigroup.com/en-GB/industries-and-sectors/Internet-of-Things/IoT-Assurance-Services/ [Cited on pages 21 and 26.]

[47] "About," 2016. [Online]. Available: https://www.iot.org.au/ [Cited on pages 21 and 26.]

[48] EUSOSMART, "[tr-e-iot-scs-part-1] process & policy v1.2," EUSOSMART, Tech. Rep., oct 2019. [Cited on pages 21 and 27.]

[49] *ETSI EG 203 251, Methods for Testing & Specification Risk-based Security Assessment and Testing Methodologies v1.1.1*, 2016. [Cited on page 23.]

[50] S. N. Matheu-García, J. L. Hernández-Ramos, A. F. Skarmeta, and G. Baldini, "Risk-based automated assessment and testing for the cybersecurity certification and labelling of IoT devices," *Computer Standards & Interfaces*, vol. 62, pp. 64–83, 2019. [Cited on page 24.]

[51] "IoT Security & Privacy," Nov 2015. [Online]. Available: https://www.icsalabs.com/technology-program/iot-testing [Cited on page 24.]

[52] U. Laboratories, "About: Underwriters Laboratories," Accessed: 2020-05-03. [Online]. Available: https://ul.org/about [Cited on page 24.]

[53] "Food and Drug Administration Modernization Act of 1997: Modifications to the List of Recognized Standards, Recognition List Number: 049," Jun 2018. [Online]. Available: https://www.federalregister.gov/documents/2018/06/07/2018-12222/food-and-drug-administration-modernization-act-of-1997-modifications-to-the-list-of-recognized?linkId=52763286 [Cited on page 25.]

[54] J. Heyl, "Overview of ul 2900," Octo 2017. [Online]. Available: https://cybersecuritysummit.org/wp-content/uploads/2017/10/4.00-Justin-Heyl.pdf [Cited on page 25.]

[55] W. . Security and N. R. of the IoTAA, "Internet of Things Security Guideline V1.2," nov 2017. [Cited on page 26.]

[56] EUSOSMART, "[tr-e-iot-scs-part-3] evaluation methodology v1.2," EUSOSMART, Tech. Rep., oct 2019. [Cited on page 27.]

[57] EUSOSMART, "[e-iot-scs-part-2] gpp v1.2," EUSOSMART, Tech. Rep., oct 2019. [Cited on page 27.]

[58] T. Alladi, V. Chamola, B. Sikdar, and K.-K. R. Choo, "Consumer IoT: Security vulnerability case studies and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 2, pp. 17–25, 2020. [Cited on page 35.]

[59] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoT-POT: analysing the rise of IoT compromises," in *9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*, 2015. [Cited on page 36.]

[60] V. Sachidananda, S. Siboni, A. Shabtai, J. Toh, S. Bhairav, and Y. Elovici, "Let the cat out of the bag: A holistic approach towards security analysis of the Internet of Things," in *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*, 2017, pp. 3–10. [Cited on pages 37 and 39.]

[61] O. A. Waraga, M. Bettayeb, Q. Nasir, and M. A. Talib, "Design and implementation of automated IoT security testbed," *Computers & Security*, vol. 88, p. 101648, 2020. [Cited on pages 37 and 39.]

[62] G. Chu and A. Lisitsa, "Penetration testing for Internet of Things and its automation," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2018, pp. 1479–1484. [Cited on pages 37 and 39.]

[63] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: Security evaluation of home-based IoT deployments," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1362–1380. [Cited on pages 37 and 39.]

[64] J. Chen, W. Diao, Q. Zhao, C. Zuo, Z. Lin, X. Wang, W. C. Lau, M. Sun, R. Yang, and K. Zhang, "IoTFuzzer: Discovering Memory Corruptions in IoT Through App-based Fuzzing." in *NDSS*, 2018. [Cited on pages 38 and 39.]

[65] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT sentinel: Automated device-type identification for security enforcement in IoT," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2177–2184. [Cited on pages 38 and 39.]

[66] "SPAN and RSPAN," Accessed: 2020-09-24. [Online]. Available: https://community.cisco.com/t5/networking-documents/understanding-span-rspan-and-erspan/ta-p/3144951 [Cited on page 44.]

[67] N. Viola, S. Corpino, M. Fioriti, and F. Stesina, "Functional analysis in systems engineering: Methodology and applications," in *Systems engineering-practice and theory*. InTechOpen, 2012. [Cited on page 45.]

[68] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014. [Cited on page 48.]

[69] “Open Container Initiative,” Accessed: 2020-06-24. [Online]. Available: https://opencontainers.org/ [Cited on page 48.]

[70] “Firmware Analysis and Comparison Tool,” Accessed: 2020-08-4. [Online]. Available: https://github.com/fkie-cad/FACT_core [Cited on page 48.]

[71] “Mariadb,” Accessed: 2020-08-4. [Online]. Available: https://mariadb.org/ [Cited on page 48.]

[72] “Wi-Fi Alliance,” Accessed: 2020-08-3. [Online]. Available: https://www.wi-fi.org/ [Cited on page 48.]

[73] N. Kushalnagar, G. Montenegro, and C. Schumacher, “IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals,” Internet Requests for Comments, RFC Editor, RFC 4919, August 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4919.txt [Cited on page 48.]

[74] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, and M. A. Spirito, “An IDS framework for internet of things empowered by 6LoWPAN,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 1337–1340. [Cited on page 48.]

[75] Z.-W. Alliance, “The Internet of Things is powered by Z-Wave.” 2020. [Online]. Available: https://z-wavealliance.org/ [Cited on page 48.]

[76] Z. Alliance, “Zigbee alliance,” Jun 2020. [Online]. Available: https://zigbeealliance.org/ [Cited on page 48.]

[77] B. SIG, “Mesh Networking Specifications,” 2020. [Online]. Available: https://www.bluetooth.com/specifications/mesh-specifications/ [Cited on page 49.]

[78] S. Pallavi and V. A. Narayanan, “An Overview of Practical Attacks on BLE Based IOT Devices and Their Security,” in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*. IEEE, 2019, pp. 694–698. [Cited on page 49.]

[79] “Python,” Accessed: 2020-06-24. [Online]. Available: https://www.python.org/ [Cited on page 51.]

[80] "The go programming language," Accessed: 2020-08-4. [Online]. Available: https://golang.org/ [Cited on page 51.]

[81] I. ISO, "IEC 9899: 2011 Information technology—Programming languages—C," *International Organization for Standardization, Geneva, Switzerland*, vol. 27, p. 59, 2011. [Cited on page 51.]

[82] "Scapy," Accessed: 2020-08-4. [Online]. Available: https://scapy.net/ [Cited on page 51.]

[83] "Nfstream," Accessed: 2020-08-4. [Online]. Available: https://pypi.org/project/nfstream/ [Cited on page 51.]

[84] J. M. Perkel, "Programming: pick up Python," *Nature News*, vol. 518, no. 7537, p. 125, 2015. [Cited on page 51.]

[85] "Gopacket," Accessed: 2020-08-2. [Online]. Available: https://github.com/google/gopacket [Cited on page 52.]

[86] "Identification of flows based on network ports," Accessed: 2020-08-2. [Online]. Available: https://github.com/mushorg/go-dpi [Cited on page 52.]

[87] "Tcpdump/libpcap public repository." [Online]. Available: https://www.tcpdump.org/ [Cited on page 52.]

[88] "Issues 329," Accessed: 2020-06-25. [Online]. Available: https://github.com/google/gopacket/issues/329 [Cited on page 52.]

[89] "Pf_ring dev documentation," Accessed: 2020-08-4. [Online]. Available: https://www.ntop.org/guides/pf_ring/get_started/index.html [Cited on page 53.]

[90] "Pf_ring documentation," Accessed: 2020-06-25. [Online]. Available: https://www.ntop.org/guides/pf_ring/index.html [Cited on page 53.]

[91] "Pull 553," Accessed: 2020-06-25. [Online]. Available: https://github.com/google/gopacket/pull/553 [Cited on page 53.]

[92] "Performance difference between af_packet and libpcap," Accessed: 2020-08-2. [Online]. Available: https://discuss.elastic.co/t/performance-difference-between-af-packet-libpcap/69766/2 [Cited on page 53.]

[93] “packet(7) — linux manual page,” Accessed: 2020-06-25. [Online]. Available: https://www.man7.org/linux/man-pages/man7/packet.7.html [Cited on page 53.]

[94] G. Insolvibile *et al.*, “The linux socket filter: Sniffing bytes over the network,” *Linux Journal*, vol. 86, 2001. [Cited on page 53.]

[95] “Identification of flows based on network ports,” Accessed: 2020-08-2. [Online]. Available: https://github.com/mushorg/go-dpi/issues/53 [Cited on page 54.]

[96] “Package plugin,” Accessed: 2020-08-2. [Online]. Available: https://golang.org/pkg/plugin/ [Cited on page 55.]

[97] “grpc,” Accessed: 2020-08-3. [Online]. Available: https://grpc.io/ [Cited on page 55.]

[98] “Protocol buffers,” Accessed: 2020-08-3. [Online]. Available: https://developers.google.com/protocol-buffers/ [Cited on page 55.]

[99] “Go cron,” Accessed: 2020-08-2. [Online]. Available: https://github.com/robfig/cron [Cited on page 56.]

[100] “nmap,” Accessed: 2020-08-3. [Online]. Available: https://nmap.org/ [Cited on page 56.]

[101] “OpenVAS - Open Vulnerability Assessment Scanner,” Accessed: 2020-08-3. [Online]. Available: https://openvas.org/ [Cited on page 56.]

[102] “Ullaakut/nmap,” Accessed: 2020-08-3. [Online]. Available: https://github.com/Ullaakut/nmap [Cited on page 56.]

[103] “Open Scanner Protocol,” Accessed: 2020-08-3. [Online]. Available: https://docs.greenbone.net/API/OSP/osp.html [Cited on page 57.]

[104] B. Moran, M. Meriac, and H. Tschofenig, “Firmware Manifest Format,” Working Draft, IETF Secretariat, Internet-Draft draft-moran-suit-manifest-01, January 2018. [Online]. Available: http://www.ietf.org/internet-drafts/draft-moran-suit-manifest-01.txt [Cited on page 58.]

[105] K. Zandberg, K. Schleiser, F. Acosta, H. Tschofenig, and E. Baccelli, “Secure firmware updates for constrained IoT devices using open standards: A reality check,” *IEEE Access*, vol. 7, pp. 71 907–71 920, 2019. [Cited on page 58.]

[106] B. Moran, M. Meriac, H. Tschofenig, and D. Brown, "A Firmware Update Architecture for Internet of Things Devices," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-suit-architecture-05, April 2019. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-suit-architecture-05.txt [Cited on page 58.]

[107] K. Zandberg, K. Schleiser, F. Acosta, H. Tschofenig, and E. Baccelli, "Secure Firmware Updates for Constrained IoT Devices Using Open Standards: A Reality Check," *IEEE Access*, vol. 7, pp. 71 907–71 920, 2019. [Cited on page 58.]

[108] J. Schaad, "CBOR Object Signing and Encryption (COSE)," Internet Requests for Comments, RFC Editor, RFC 8152, July 2017. [Cited on page 58.]

[109] "Flask," Accessed: 2020-07-23. [Online]. Available: https://github.com/pallets/flask [Cited on page 61.]

[110] "Jinja," Accessed: 2020-07-23. [Online]. Available: https://jinja.palletsprojects.com [Cited on page 61.]

[111] "qemu," Accessed: 2020-09-15. [Online]. Available: https://www.qemu.org/ [Cited on page 66.]

[112] "Openwrt," Accessed: 2020-09-15. [Online]. Available: https://openwrt.org/ [Cited on page 66.]

[113] "Owasp/iotgoat," Accessed: 2020-09-15. [Online]. Available: https://github.com/OWASP/IoTGoat [Cited on page 66.]

[114] C. O'Flynn, "Getting root on philips hue bridge 2.0," *as accessed on*, vol. 2, no. 8, 2019. [Cited on page 66.]

[115] "Telegraf Open Source Server Agent," Accessed: 2020-09-15. [Online]. Available: https://www.influxdata.com/time-series-platform/telegraf/ [Cited on page 66.]

[116] "question Tls example," Accessed: 2020-08-21. [Online]. Available: https://github.com/google/gopacket/issues/687#issuecomment-532591699 [Cited on page 70.]

[117] "Package tls," Accessed: 2020-09-09. [Online]. Available: https://golang.org/pkg/crypto/tls/ [Cited on page 70.]

[118] "Adventures in Go: Accessing Unexported Functions," Accessed: 2020-03-3. [Online]. Available: https://www.alangpierce.com/blog/2016/03/17/adventures-in-go-accessing-unexported-functions/ [Cited on page 70.]

[119] "spance/go-callprivate," Accessed: 2020-03-3. [Online]. Available: https://github.com/spance/go-callprivate [Cited on page 70.]

[120] T. Polk, K. McKay, and S. Chokhani, "Guidelines for the selection, configuration, and use of transport layer security (TLS) implementations," *NIST special publication*, vol. 800, no. 52, p. 32, 2014. [Cited on page 71.]

[121] F. Andreasen, N. Cam-Winget, and E. Wang, "TLS 1.3 Impact on Network-Based Security," 2017. [Cited on page 71.]

[122] D. Brumley, I. Jager, T. Avgerinos, and E. J. Schwartz, "BAP: A binary analysis platform," in *International Conference on Computer Aided Verification*. Springer, 2011, pp. 463–469. [Cited on page 73.]

[123] "Yara," Accessed: 2020-08-24. [Online]. Available: https://virustotal.github.io/yara/ [Cited on page 74.]

[124] "zricethezav/gitleaks," Accessed: 2020-08-24. [Online]. Available: https://github.com/zricethezav/gitleaks [Cited on page 74.]

[125] "odomojuli/regexapi," Accessed: 2020-08-15. [Online]. Available: https://github.com/odomojuli/RegExAPI [Cited on page 75.]

[126] "Mrsuicideparrot/yara-secrets." [Online]. Available: https://github.com/MrSuicideParrot/Yara-Secrets [Cited on page 75.]

[127] "Mrsuicideparrot/fact-search-secrets," Accessed: 2020-08-24. [Online]. Available: https://github.com/MrSuicideParrot/FACT-search-secrets [Cited on page 75.]

[128] A. Takanen, J. D. Demott, C. Miller, and A. Kettunen, *Fuzzing for software security testing and quality assurance*. Artech House, 2018. [Cited on page 75.]

[129] "ansible," Accessed: 2020-09-12. [Online]. Available: https://www.ansible.com/ [Cited on page 76.]

[130] *Cyber Security for Consumer Internet of Things*, Nov 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.00.00_20/en_303645v020000a.pdf [Cited on page 79.]

[131] "Etsi types of standard," Accessed: 2020-09-1. [Online]. Available: https://www.etsi.org/standards/types-of-standards [Cited on page 79.]

[132] "ETSI and Uk government," Accessed: 2020-09-1. [Online]. Available: https://www.gov.uk/government/publications/etsi-industry-standard-based-on-the-code-of-practice [Cited on page 79.]

[133] J. S. Resende, P. R. Sousa, R. Martins, and L. Antunes, "Breaking mpc implementations through compression," *International Journal of Information Security*, vol. 18, no. 4, pp. 505–518, 2019. [Cited on page 83.]