Bachelor Thesis

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Strategies for the Security Assessment of IoT Devices by Certification Authorities

Moritz Anton Finke
Department of Computer Science
Chair of Computer Science II (Software Engineering)
Secure Software Systems Research Group

Prof. Dr.-Ing. Alexandra Dmitrienko
First Reviewer & First Advisor

Submission
26th May 2020

www.uni-wuerzburg.de

Abstract

The Internet of Things (IoT) is a global infrastructure that interconnects physical and virtual things based on evolving information and communication technologies [1]. Along with the rise of the IoT, multiple unprecedented forms of security issues and attacks have emerged. While their limited resources disallow the utilization of computing-intensive security measures, the services of IoT devices represent an attractive target for vulnerability exploitation. Certification Authorities (CAs) have responded to the trend and offer security assessments and certification for IoT devices. To ensure qualitative and fair certification, security testing requires well-defined strategies. This bachelor thesis analyzes the existing IoT security assessment models and proposes a novel, CA oriented Testing Guide Model (TGM) for standardized and reproducible assessments. The TGM describes a complete security assessment approach for application in IoT device security certification procedures of CAs. As part of the bachelor thesis, the TGM is specified, implemented, and its applicability evaluated with the help of existing IoT devices.

CAs account for the security of certified products. The certification of devices that (subsequently) prove to be insecure can lead to a consumer's loss of trust in the CA. Therefore, the decision for issuing certificates must be comprehensible and made carefully. For the risk estimation of security issues and vulnerabilities, it is common to make use of numerical scoring systems. Well-defined scoring systems allow precise and reproducible estimations. In addition to the introduction of the TGM, this bachelor thesis analyzes the scoring systems established in the field of IT security in regard to their applicability for the scoring of IoT related vulnerabilities, complete IoT devices, and for the specific requirements of CAs. This thesis uncovers incompatibilities and insufficiencies in the existing systems that disallow application for the scoring of IoT devices and for application by CAs. To counter these issues, the thesis introduces a novel Security Scoring System (SSS) for rating IoT devices based on the risks of their vulnerabilities, services, and operational contexts. The SSS is integrated into the TGM, implemented in software, and evaluated with a comparison to the existing scoring systems.

To summarize, the contributions of this bachelor thesis are two novel models for the security assessment of devices of the IoT: the TGM and the SSS. The models are intended to find application for CAs and both standardize and improve the security testing and assessment procedures.

Zusammenfassung

Das Internet der Dinge (IoT) stellt eine globale Infrastruktur dar, welche physische und virtuelle Geräte mit stetig weiterentwickelten Informations- und Kommunikationstechnologien verbindet [1]. Mit der Verbreitung des IoT in Industrie, öffentlicher Infrastruktur und Wohnhäusern entstehen bisher unbekannte Schwachstellen und Angriffsmöglichkeiten. Geräte des IoT werden in unterschiedlichen Umgebungen eingesetzt und können aufgrund ihrer individuellen Funktionalitäten neben den Auswirkungen von konventionellen Angriffen auch physische Schäden an den Geräten selbst sowie ihrer Umgebung verursachen. Für IoT Geräte ist die Absicherung vor Angriffen deshalb von hoher Wichtigkeit. Zertifizierungsstellen bieten die Überprüfung der Sicherheit und Entdeckung möglicher Schwachstellen als Dienstleistung mit Vergabe von Zertifikaten oder Prüfsiegeln an. Hierfür bedarf es festgelegter Strategien und Modelle, welche die Form des Verfahrens zur Zertifizierung definieren und eine vollständige und gleichbehandelnde Überprüfung ermöglichen. Da bestehende Modelle für die Überprüfung von IoT-Sicherheit den individuellen Anforderungen von Zertifizierungsstellen nicht gerecht werden, wird mit dieser Bachelor-Thesis erstmalig ein Testing Guide Modell (TGM) für die konkreten Anforderungen von Sicherheitstests für IoT Geräte im Kontext von durch Zertifizierungsstellen durchgeführte Überprüfungen entwickelt. Das TGM wird im Rahmen der Bachelor-Thesis implementiert und anhand von existierenden IoT Geräten hinsichtlich dessen Anwendbarkeit evaluiert.

Mit der Vergabe von Prüfsiegeln bestätigen Zertifizierungsstellen gegenüber den Kunden von Produkten deren geprüfte Sicherheit. Werden unsichere Geräte fälschlicherweise zertifiziert, kann dies zum Verlust des Vertrauens in die Zertifizierungsstelle führen. Die Entscheidung zur Vergabe muss genau abgewogen und nachvollziehbar sein. Zur Einschätzung von Schwachstellen werden im Gebiet der IT-Security numerische Bewertungssysteme genutzt. Mit eindeutig definierten Bewertungssystemen sind Einschätzungen sowohl begreiflich als auch reproduzierbar. Mit dieser Bachelor-Thesis werden zusätzlich zur Entwicklung TGM die etablierten Bewertungssysteme hinsichtlich ihrer Anwendbarkeit für die Bewertung von IoT Geräten sowie für die Anforderungen von Zertifizierungsstellen untersucht. Anschließend wird ein neues Bewertungssystem, das Security Scoring System (SSS), entwickelt, welches die Sicherheit von IoT Geräten anhand dessen Schwachstellen, Funktionalität und operationeller Umgebung bewertet und für den Einsatz durch Zertifizierungsstellen geeignet ist. Das SSS wird im TGM integriert, implementiert und anhand eines Vergleichs mit bestehenden Bewertungssystemen evaluiert.

Mit dieser Bachelor-Thesis werden die zwei neuartigen Modelle TGM und SSS für die Sicherheitsüberprüfung von Geräten des IoT entwickelt. Die Modelle sollen bei Prüfverfahren von Zertifizierungsstellen Anwendung finden und sowohl das Prüfverfahren selbst, als auch die Sicherheitsbewertung von IoT Geräten standardisieren und verbessern.

Contents

Glossary

Black Box "A system [...] whose inputs, outputs, and general function are known but whose contents or implementation are unknown or irrelevant" [2].

Certification Authority (CA) A CA assesses a DUT's compliance to specified certification criteria via a certification process. If all requirements are satisfied, the CA issues a certificate for the Device Under Test (DUT)'s product type.

Certification "The process of confirming that a system or component complies with its specified requirements and is acceptable for operational user" [2].

Certification Criteria "A set of standards, rules or properties to which an asset must conform in order to be certified to a certain level" [2].

Certification Process "The process of assessing whether an asset conforms to predetermined certification criteria appropriate for that class of asset" [2].

Client A company or a similar entity that requests a CA to perform a certification process on its product in order to obtain a product certification.

Device Base Risk Score The Device Base Risk Score (D_{BRS}) is a component of the SSS. It measures an IoT device's risks based on its service capabilities and operational contexts. A high D_{BRS} indicates risky service capabilities and low risk operational contexts, while a low D_{BRS} indicates risky operational contexts or few and/or low risk service capabilities.

Device Risk Score The Device Risk Score is the product of the WRR model and represents an IoT device's risk estimation based on its vulnerabilities, service capabilities and operational contexts.

Device Under Test (DUT) A DUT is a physical or virtual item that is subject of a CA's certification process. It is a sample device of a product type that is to be issued a certificate.

Iterative Testing Procedure (ITP) The ITP is a testing procedure that is intended to be used by CAs for performing security assessments. It utilizes TGM components and covers the complete testing cycle required in a certification process.

Risk "The combination of the probability of an abnormal event or failure and the consequence(s) of that event or failure to a system's components, operators, users, or environment" [2].

Risk Acceptance "Acknowledgement of a risk factor's existence along with a decision to accept the consequences if the corresponding problem occurs" [2].

Risk Analysis "The process of examining identified risk factors for probability of occurence, potential loss, and potential risk-handling strategies" [2].

Risk Category "A class or type of risk" [2].

Risk Identification "An organized, systemic approach to determining the risk factors associated with a planned activity, project, or program" [2].

Risk Mapping Database (RMD) The RMD, defined in [3], is a database that holds information and risk scores for the vulnerabilities, service capabilities, and operational contexts of IoT devices.

Risk Mitigation "A course of action to reduce the probability of and potential loss from a risk factor" [2].

Risk Register "The document containing the results of the qualitative risk analysis, quantitative risk analysis, and risk response planning. The risk register details all identified risks, including description, category, cause, probability of occurring, impact(s) on objectives, proposed responses, owners, and current status" [2].

Security Risk Assessment (SRA) Model An SRA model describes the steps and strategies for assessing a certain asset's security risk.

Security Score The Security Score is a numerical estimation of the security of an IoT device. The Security Score is computed by the SSS.

Security Scoring System (SSS) The SSS assigns numerical scores to the security of IoT devices. The security score is based on the devices' service capabilities, operational contexts, and vulnerabilities.

Test "An activity in which a system is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component" [2].

Test Case "Documentation specifying inputs, predicted results, and a set of execution conditions for a test item" [2].

Test Execution "Act of performing one ore more test cases" [2].

Testing Guide Model (TGM) The TGM is an abstract set of components and relations that is utilized in a certification process. It is specified in Chapter 4.

Test Management System (TMS) The TMS is an abstract set of components and relations that is utilized in a testing procedure.

Weighted Risk Ranking Method (WRRM) The WRRM, defined in [3], assigns weights to risks provided as input.

Weighted Risk Ranking (WRR) Model The WRR model, defined in [3], assigns numerical risk scores to the security of IoT devices. The risk score is based on the devices' service capabilities, operational contexts, and vulnerabilities.

White Box "A system [...] whose internal contents or implementation are known" [2].

Abbreviations

BRS	Base Risk Score
BSI	Bundesamt für Sicherheit in der Informationstechnik
CA	Certification Authority
CAPEC	Common Attack Pattern Enumeration and Classification
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DoS	Denial of Service
DUT	Device Under Test
IoT	Internet of Things
IoTSF	IoT Security Foundation
ITP	Iterative Testing Procedure
NVD	National Vulnerability Database
OSI	Open Systems Interconnection
RMD	Risk Mapping Database
SRA	Security Risk Assessment
SSS	Security Scoring System
TGM	Testing Guide Model
TL	TestLink
TMS	Test Management System
VDE	Verband der Elektrotechnik, Elektronik und Informationstechnik
WRRM	Weighted Risk Ranking Method
WRR	Weighted Risk Ranking
YAML	YAML Ain't Markup Language

1. Introduction

The Internet of Things (IoT) is a constantly evolving field with growing market share [4]. For the year 2020, a number of 5.8 billion IoT endpoints is projected [4]. This huge number of devices and their unique conjunction between software and the real, physical world creates unprecedented security vulnerabilities and allows for novel attacks. In late 2016, the distributed Denial of Service (DoS) attack Mirai harmed multiple Internet connected high-profile targets. The Mirai attack peaked with a number of 600.000 infections and was largely backed by IoT devices – platforms that are characterized by their low level of resources [5]. Yet, the Mirai attack was sufficiently powerful to cause outages in multiple targets, including servers of Deutsche Telekom [5]. Mirai is just an example for a multitude of attacks that either exploit IoT devices as a tool to harm their targets or damage the devices themselves.

With the rise of attacks in the IoT, the interest in IoT security has increased correspondingly [6]. In an effort to motivate manufacturers to develop more secure IoT products and allow consumers to make purchase decisions based on a product's assessed security, the Bundesamt für Sicherheit in der Informationstechnik (BSI) was commissioned to roll out a certification program for IoT products [7]. Similar certification programs are also offered by independent Certification Authorities (CAs) like Verband der Elektrotechnik, Elektronik und Informationstechnik (VDE). At VDE, manufacturers of IoT devices voluntarily request information security assessments for their products. If deemed secure, devices can be marketed with a proprietary VDE seal to signify to consumers that the manufacturer attributes importance to information security and offers reviewed products that satisfy a high security standard [8].

For proprietary IoT security certificates, it is in the interest of CAs to keep their testing procedures secret, as sharing those with other CAs could implicate disadvantages in competition. However, publicly developed testing procedures may be of higher advantage, since those are subject to peer-review and discussed by a greater audience. This bachelor thesis initiates this proposition and introduces a public testing procedure model for IoT device security assessments in the context of CA testing conditions. It represents an open alternative to proprietary approaches and is intended to be utilized and enhanced by CAs and the general public.

Depending on the consulted definition, it can be argued that there is no such thing as absolute security. Regardless, if this is deemed true or false, CAs must decide over the security level, at which they are willing to grant certification and thereby pay for potentially

undetected security issues with their public trust or certification license. This means that CAs do not only have to perform comprehensive security tests, but also be able to interpret the test results correctly. For precise security ratings, "quantitative methods are needed to ensure that the decisions are not based on subjective perceptions only" [9]. Such quantitative methods are typically realized in the form of scoring systems, such as the Common Vulnerability Scoring System (CVSS) [10].

Contributions and Outline The contributions of this bachelor thesis can be summarized as follows:

- Analysis of the existing risk and vulnerability scoring systems CVSS and Weighted Risk Ranking (WRR) [3] in Chapter 2. It is discovered that the components and equations of these systems hold insufficiencies that disallow the precise scoring of IoT devices and their individual vulnerabilities. Furthermore, the systems' components are found to be inapplicable for the use cases of CAs.
- Development of the SSS in Chapter 3. The introduced scoring system SSS is designed to properly express the overall security of IoT devices. The developed methods for score computation consider a device's vulnerabilities and put those in relation to its service capabilities and operational contexts. The SSS is implemented in software in Chapter 5 and compared to the existing systems CVSS and WRR in Chapter 6. It is determined to have beneficial features that improve application by CAs.
- Development of the TGM in Chapter 4. The TGM consists of multiple, interdependent segments that already find application in product certification processes. The TGM represents a proposal for standardizing the security risk assessment methods of CAs and incorporates the introduced SSS. Its segments' components are combined to form a testing procedure that covers a complete certification process. The software elements of the TGM are implemented in Chapter 5 and the TGM's applicability for utilization by CAs is evaluated in Chapter 6.

Overall, the thesis proposes a solution for the security testing and assessment of IoT devices tailored for the needs of CAs and has the potential to improve the security of IoT systems.

2. Background and Related Work

This chapter examines the scoring system solutions Common Vulnerability Scoring System (CVSS) [10] and Weighted Risk Ranking (WRR) [3] that build the foundation for the research of Chapter 3 and investigates the existing work related to those systems. Additionally, the work related to existing Security Risk Assessment (SRA) models and the requirements for Internet of Things (IoT) security testing by Certification Authorities (CAs) is surveyed (see Section 2.4).

The behavior of both scoring systems is briefly explained and their applicability for CAs analyzed (see Sections 2.2 and 2.3). Especially CVSS, a well-established and researched model [9], has experienced many changes in the past that affect the compatibility both for IoT and CAs. Its problems and the efforts made in the history of CVSS towards an IoT compatible system are investigated in greater detail with both existing work and new contributions.

2.1. The Need for Security Scoring

It is essential for security analysts to properly ascertain the severity, exploitability, and accessibility of detected security risks. Risks that are falsely assumed to only cause minor impact and therefore are ignored in later mitigation procedures can have fatal consequences [11]. A common approach for vulnerability rating is to make use of a specially designed scoring system such as the CVSS, that "has now become almost an industrial standard for assessing [...] security vulnerabilities" [9]. With CVSS, vulnerabilities are rated by their severity on a linear scale. This helps security analysts and developers in "communicating the characteristics and severity of software vulnerabilities" [10] and determining which vulnerabilities are most important to mitigate or could be ignored because of lack of impact.

Attempting to quantify the level of security of a Device Under Test (DUT) or to make any other assertion about its security is associated with many challenges. Security assessments are limited in their resources, including available amount of time for testing, the researchers' education, available information, etc. Hence, a scoring system can, at best, only represent the DUT's security level based on all available knowledge gained through prior testing. It must be clear that scoring systems only provide an approximation for a DUT's security level. Still, scoring systems are indispensable for security analysts and CAs, as they are capable of expressing and summarizing complex test results and also

Qualitative Rating	CVSS Score
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Table 2.1.: Qualitative Severity Rating Scale. Table taken from [10].

allow untrained personell to comprehend and interpret such results. Furthermore, scoring systems ensure that security ratings do not primarily rely on the researcher's subjective opinions [9].

When assessing the security of devices of the IoT, determining the severity of single vulnerabilities and risks is not sufficient, since a conclusion about the complete DUT must be drawn. The IoT introduces new aspects that both require new security mechanisms and, at the same time, exceed the scope of classical systems like CVSS. Not only a DUT's software, but also its hardware and physical components, as well as environmental aspects must be considered by a scoring system. While hardware attacks primarily impact the integrity, confidentiality, or availability of devices and networks, misuse of the physical functionality of IoT devices can lead to major disasters such as the injury of individuals [11].

2.2. Common Vulnerability Scoring System

The Common Vulnerability Scoring System (CVSS) is described as "an open framework for communicating the characteristics and severity of software vulnerabilities" [10]. It consists of three metric groups, Base, Temporal, and Environmental. Through a set of equations, the groups' metrics are merged to form a quantitative score on a range from 0.0 to 10.0 that can also be mapped onto a qualitative scale (see Table 2.1). The sections below shortly describe the model's components and investigate the applicability of CVSS for IoT and CAs.

2.2.1. CVSS Model

The CVSS score rates a vulnerability by its severity. The score is composed of the values of CVSS metrics. It is computed through CVSS equations that use the individual metrics' values [10].

2.2.1.1. Metrics

CVSS defines three metrics groups: Base, Temporal, and Environmental. Over the course of CVSS development, the groups' metrics sets have changed drastically. The metrics discussed in this section are part of the latest CVSS specification (version 3.1).

Base Metrics The Base metric group, consisting of Exploitability, Scope, and Impact metrics, holds the fundamental vulnerability properties that are not affected by time or environment [10].

Exploitability Metrics The Exploitability metrics group consists of the metrics Attack Vector, Attack Complexity, Privileges Required, and User Interaction. The Attack Vector metric measures the degree of exploitability with regard to the Open Systems Interconnection (OSI) layer an attack is executed on [10]. Attacks are rated more severe,

the more remote the attacker can be located. The Attack Complexity metric measures the prevalent conditions that are out of the attacker's control and that must be fulfilled before the attack is executed. The Privileges Required metric (originally termed Authentication metric in CVSS v2.0 [12]) is used to measure the system specific privileges required for successful exploitation [10]. The User Interaction metric indicates, if a user must perform a specific action before an attack can be executed successfully [10].

Scope Metric The Scope metric is introduced in CVSS v3.0, after "CVSS v2.0 presented difficulties for vendors when scoring vulnerabilities that would fully compromise their software, but only partially affect the host operating system" [13]. The Scope metric focuses on a component's security scope that includes all objects with the same security authority. The metric indicates, if a vulnerability is capable of impacting resources beyond the security scope of the attacked object, i.e., if it affects objects beyond its security authority [10].

Impact Metrics The Impact metrics group measures the worst case that can result from an attack [10]. The group consists of the metrics Confidentiality, Integrity, and Availability. Each metric measures the loss of its characteristic through a successfully executed attack. The metrics can be combined with the Scope metric to respect their effects on components located beyond the security scope [10].

Temporal Metrics The Temporal metrics group consists of the metrics Exploit Code Maturity, Remediation Level, and Report Confidence. It measures "the current state of exploit techniques or code availability, the existence of any patches or workarounds, or the confidence in the description of a vulnerability" [10]. It is questionable, whether Temporal metrics can be utilized by CAs, as it can be argued that newly discovered vulnerabilities in unpublished products would always be assigned the same metrics values. The effectiveness of Temporal metrics is discussed further in Section 2.2.4.

Environmental Metrics The Environmental metrics group consists of the Security Requirements and Modified Base Metrics. The latter metric replaces the metrics Collateral Damage Potential and Target Distribution of CVSS v2.0 [13]. The Security Requirements metrics enable users to modify CVSS scores based on the user's assigned priorities for confidentiality, availability, and integrity. The Modified Base Metrics subgroup redefines the existing Base metrics for a given environment. This enables users to overwrite the existing Base metrics to respect the user's specific environment in which the vulnerable device operates [10].

2.2.1.2. Scoring

The CVSS Score is composed of the assigned metric values that are combined through equations specified in [10]. The numerical score is defined to lay between 0.0 and 10.0 and mapped to a qualitative range (see Table 2.1).

To describe and transfer the characteristics of vulnerabilities better, it is possible to use a CVSS Vector String [10]. The Vector string represents all rated metrics and their assigned values in a compact form and helps expressing a vulnerability's characteristics beyond its quantitative or qualitative score. The Vector String holds all information required for computing a CVSS Score.

For scoring, it is optional to include the Temporal and Environmental metrics groups. If only Base metrics are used, the score is referred to as Base Score. The Base Score is typically produced by the vulnerable product's maintaining organization. It is recommended to leave the scoring of Temporal and Environmental metrics to CVSS consumers that are able to score these aspects with regard to their specific use cases and environments [10].

Hence, databases, such as the National Vulnerability Database (NVD), generally only store the vulnerabilities' Base Scores [14].

2.2.2. Risk Scoring with CVSS

The CVSS guide states "that CVSS is designed to measure the severity of a vulnerability and should not be used alone to assess risk" [15, p. 4]. To understand why CVSS is unable to provide precise scores for risks, the composition of risks must be investigated. In [2], multiple broadly accepted risk definitions are listed. Definition 2 describes risk as "the combination of the probability of an abnormal event or failure and the consequence(s) of that event or failure to a system's components, operators, users, or environment." Definition 4 depicts risk as "a measure that combines both the likelihood that a system hazard will cause an accident and the severity of that accident." CVSS is designed to produce "scores indicating the severity of a vulnerability" [15, p. 2] but can not be used for rating the probability of such attack. Hence, risks – consisting of both the severity and probability of an attack or vulnerability – can not be rated with CVSS. CVSS does employ exploitability metrics, which could be understood as probability rating mechanisms. However, the CVSS's exploitability metrics group lacks components for comprehensive probability assessment, such as service capabilities metrics and operational contexts metrics. Also, during CVSS score calculation, the exploitability and impact metrics are added, instead of multiplied, which is abnormal to generally accepted risk calculation [9].

2.2.3. Version Changes affecting the Applicability for the Internet of Things

CVSS is continuously developed. The first version of CVSS was published in 2005 and version 2.0 followed in 2007 [12]. Because CVSS v3.0 (2015) and v3.1 (2019) are considerably more recent, CVSS v2.0 scores are still well-established and used by databases such as the NVD [14]. The CVSS has undergone heavy changes over the course of its published versions. Each major version change has added and removed multiple metrics. Those changes also affect the applicability for IoT vulnerabilities. The sections below analyze the differences in the CVSS versions in regard to compatibility with the IoT.

2.2.3.1. CVSS Version 2.0

For assessing an IoT device's physical risks, it is insufficient to only focus on attacks that are made possible through physical security issues: risk assessment must also consider the physical consequences of attacks. Devices of the IoT are typically utilized to automate physical tasks. Depending on the task, misuse can lead to injury of individuals or damage of other nearby physical components. In CVSS v2.0, such harm is considered by the Collateral Damage Potential metric, a member of the Environmental metrics group. The metric is deliberately drafted to measure "the potential for loss of life or physical assets through damage or theft of property or equipment" [12] and offers to choose from six different qualitative and quantitative values. However, as also briefly mentioned in [11], the Collateral Damage Potential metric's impact is much smaller than the impact of other metrics, such as the environmental metric Target Distribution. This metric "measures the proportion of vulnerable systems" [12] in a certain environment. It is argued that the impact of the Collateral Damage Potential metric on the Environmental Score should be increased in respect to the physical consequences it can measure [11]. To investigate this criticism, the equations used for Environmental Score computation must be analyzed.

$$\begin{aligned} \text{EnvironmentalScore} = \text{round\_to\_1\_decimal}((&\text{AdjustedTemporal} \\ &+ (10 - \text{AdjustedTemporal}) \cdot \text{CollateralDamagePotential}) \\ &\cdot \text{TargetDistribution}) \end{aligned} \tag{2.1}$$

Target Distribution	Target Distribution Description	Collateral Damage Potential	Adjusted-Temporal	Environmental Score
None (0)	No target systems exist	High (0.5)	5.0	0.0
Low (0.25)	1% to 25% of the total environment is at risk	High (0.5)	5.0	1.9
Medium (0.75)	26% to 75% of the total environment is at risk	High (0.5)	5.0	5.6
High (1.0)	76% to 100% of the total environment is considered at risk	High (0.5)	5.0	7.5

Table 2.2.: Environmental Scores under CVSS v2.0. Target Distribution Descriptions taken from [12].

The Environmental Score is computed with Equation 2.1 [12]. In this equation, the AdjustedTemporal variable represents the previously computed Base or Temporal Score. The variables CollateralDamagePotential and TargetDistribution hold the corresponding metrics' values.

The issues of the Collateral Damage Potential metric's impact are discussed with the help of an exemplary vulnerability. Let this vulnerability's Base Score be 5 (AdjustedTemporal = 5) and its Collateral Damage Potential set to High (0.5) in order to indicate high potential for physical harm induced through exploitation. Table 2.2 lists the resulting Environmental Scores based on the different values of the Target Distribution metric.

For a High Target Distribution, a High Collateral Damage Potential is capable of increasing the Base Score of 5.0 by 50 percent (7.5). However, for a Low Target Distribution, the same Collateral Damage Potential is unable to increase the Base Score: the Environmental Score of a vulnerability with Base Score 5.0 and Target Distribution Low is 1.9. For a Medium Target Distribution, the Base Score is increased by 12 percent. From this observation, it can be concluded that huge impact through the Collateral Damage Potential is only possible, if the vulnerability's Target Distribution is rated High.

Translating those scores to the high potential of loss of live that can be represented through the chosen value for the Collateral Damage Potential metric, a vulnerability with potentially deadly consequences that exists in "1% – 25% of the total environment" [12] results in the qualitative CVSS score Low (1.9). A potentially deadly vulnerability that exists in "76% – 100% of the total environment" [12] results in the qualitative CVSS score High (7.5), far from the value Critical. These circumstances indicate that the utilization of CVSS v2.0 Environmental Scores for rating the physical consequences of exploited IoT devices is inexpedient.

With CVSS v3.0, both metrics were removed from the Environmental metrics group, as "the Environmental metrics of Target Distribution and Collateral Damage Potential were not found to be useful" [13, p. 8]. While the above analysis affirms this conclusion, the removal of the Collateral Damage Potential also means that physical harm is no longer deliberately considered by the Environmental metric.

2.2.3.2. CVSS Version 3.0

With CVSS v3.0, the Attack Vector value Physical is introduced. An attack with Attack Vector value Physical "requires the attacker to physically touch or manipulate the vulnerable component. Physical interaction may be brief [...] or persistent" [10]. For devices of

the IoT, physical aspects are a crucial extension to CVSS. IoT devices like weather stations, doorbells, cameras, or alarm bells are typically located outdoors. Without sufficient tamper protection, such devices are vulnerable to physical attacks. However, CVSS does not consider the physical security mechanisms that need to be bypassed in order to gain physical access to a device. Physical access to a weather station located in a smart home's front yard is treated the same as access to a server rack locked in a secured room. While the Attack Complexity metric could be utilized to express physical security mechanisms, it is not developed for such use case. On the one hand, Attack Complexity only differentiates between the metric values Low and High, which can be considered as insufficient differentiation. On the other hand, in most cases, even in the case of the weather station, the vulnerability must be rated High, because the exemplary requirement, "the attacker must gather knowledge about the environment in which the vulnerable target/component exists" [10], is satisfied with the explanation that the attacker must gather information about the front yard. The insufficient recognition given to physical security is also represented through the importance given to the value Physical. Out of all qualitative values for the Attack Vector metric, Physical has the lowest quantitative value (0.2). The small value is rooted in the idea that "the metric value [...] will be larger the more remote an attacker can be" [10].

Another point of criticism to the updated Attack Vector metric is the lack of differentiation in the metric's values. This is affirmed with an example for attacks through WiFi connections described in [11]. This example can be generalized for all wireless communication. Since a large part of IoT devices communicates wirelessly, exploitability must be determined correctly. WiFi connections require physical proximity, but, for the Attack Vector value of a WiFi enabled device, "the closest option is 'Adjacent Network', which roughly means 'on the same Layer 2 segment"'[11]. While "Version 3.0 introduces an additional option, Physical, [...] it's intended to indicate the need to actually touch the device, not account for mere proximity" [11]. The Attack Vector values Network, Adjacent, and Local focus on the layers of the OSI model. Those stand in strong contrast to the value Physical that prescribes direct contact. Additional values that respect physical proximity are missing, but would be helpful, as an example of "a radio in a car [that] requires a physical proximity from a few feet (key fobs) to thousands of miles (satellite radio)" [11] affirms.

In addition to the introduction of the value Physical for the Attack Vector metric, CVSS v3.0 also introduces a new Base metric labeled Scope. This metric offers the values Changed and Unchanged and indicates, if "a vulnerability in one software component [is able] to impact resources beyond its means, or privileges" [16]. Since the Collateral Damage Potential metric was removed in CVSS v3.0 [13, p. 8], it is plausible to investigate, if the Scope metric constitutes a suitable alternative for including physical consequences in a CVSS v3.0 score. In [11], the Scope metric is applied to a vulnerable automobile component that possibly affects the security of the car itself. "While the vulnerability is in the [component], the scope of control is the entire car" [11]. Therefore, the Scope is determined to be Changed. However, as it already was the case in CVSS v2.0, the metric does not have huge impact on the score: "changing Scope to Unchanged would only drop the CVSS 3.0 score from 9.6 to 8.3" [11]. Since Scope only distinguishes between Changed and Unchanged, it can only express that humans or other components are affected by a vulnerability and not how, or to which degree they are affected.

As already depicted in Section 2.2.3.1, the environmental metrics Collateral Damage Potential and Target Distribution were removed in CVSS v3.0 and "replaced with Mitigating Factors" [13]. The metrics and values introduced in CVSS v3.0, namely Scope and the Attack Vector value Physical, do not compensate the removal of Collateral Damage Potential. To increase the impact of the physical aspects of the IoT, a raise of the metric's impact would have had to be enforced, instead of the metric's removal.

2.2.3.3. CVSS Version 3.1

CVSS v3.1 does not modify the set of metrics of CVSS v3.0. However, the CVSS guide for version 3.1 advises the extension of CVSS through additional, self-developed metrics "to allow industry sectors such as privacy, safety, automotive, healthcare, etc., to score factors that are outside the core CVSS standard" [15, p. 14]. The CVSS Extensions Framework, introduced in [15], defines rules for implementing such extensions. While it prohibits the modification of existing metrics and metric groups, which would be necessary to implement measures that solve the problems described in Section 2.2.3.2, the framework allows the definition of new metrics. Those could be utilized to consider the various aspects of vulnerabilities specific to the IoT. However, due to the very recent release of CVSS v3.1, CVSS extensions, that solve prior issues of CVSS for the IoT and that are made in accordance to the CVSS Extensions Framework, are not known to this date.

Because this bachelor thesis introduces a scoring system for the security of IoT devices, which can not be achieved through CVSS alone (as concluded in Section 2.2.2), focus is not laid on utilizing the CVSS Extensions Framework to solve the problems of CVSS for the IoT. Instead, it investigates the applicability of a risk rating model that includes the CVSS as one of several components (see Chapter 3). Thereby, future CVSS extensions developed deliberately for IoT support can easily be added to the introduced scoring system.

2.2.4. CVSS for Certification Authorities

CAs test DUTs under different conditions: while some DUTs may be tested over the course of development, the product line of others may already be released on the market, when testing is begun. Further, some vulnerabilities are newly discovered by the CA, while others are already publicly known. These conditions can influence the mitigation complexity and the availability of functional exploits for the products' vulnerabilities and consequently have impact on the DUT's security assessment. Of all of the CVSS's metrics groups, the Temporal metrics group is most suited for these requirements, as it "measure[s] the current state of exploit techniques or code availability, the existence of any patches or workarounds, or the confidence in the description of a vulnerability" [10]. It is to be investigated, if these metrics can be used for the CAs' different test conditions.

For analyzing the applicability of Temporal metrics, a set of CA specific test conditions, that potentially impact the applicability, was identified:

1. A detected vulnerability is publicly known.
2. A vulnerability is newly discovered by the CA.
3. The DUT's product line is available on the market.
4. The DUT's product line is unpublished.
5. A specific product version of the DUT is tested.
6. The DUT's product version is changed over the course of security assessment (i.e. the product is in a phase of development).

It is found that the applicability of Temporal metrics varies, based on the set of conditions that are present in a testing procedure. The applicability of the individual metrics is analyzed in the sections below.

2.2.4.1. Exploit Code Maturity

The Exploit Code Maturity metric "measures the likelihood of the vulnerability being attacked, [...] based on the current state of exploit techniques, exploit code availability, or

active, 'in-the-wild' exploitation" [10]. If a vulnerability, discovered in a DUT, is publicly known, e.g., because it is situated in a DUT's component that is also present in other devices, this metric can help the CA in expressing the likelihood of exploitation to the client. However, if a vulnerability was newly discovered by the CA, the current state of exploit techniques is at the very beginning. If the CA does not develop a Proof-of-Concept exploit, the metric must either be left out, or its value is set to Not Defined [10].

2.2.4.2. Remediation Level

The Remediation Level metric measures the development of vulnerability mitigating measures. "The less official and permanent a fix, the higher the vulnerability score" [10]. The metric holds the values Official Fix, Temporary Fix, Workaround, Unavailable, and Not Defined. CAs typically demand manufacturers to provide comprehensive fixes that can be rated as Official Fix. If the vulnerable product has not yet entered the market and if the manufacturer is able to develop such fix, the vulnerability can be considered no longer existent. The same is true for products already released on the market, if only the most recent product version is assessed and the manufacturer is able to remove the vulnerability in that version. In those cases, the vulnerability entry and with it the CVSS score would no longer be considered for security assessment. Under these conditions, the Remediation Level metric is redundant for the vulnerability score. In contrast to this scenario, the metric can come to use, if the certification process allows risk acceptance: the metric's value Unavailable can be utilized to indicate that the manufacturer decided to accept the vulnerability and provide no security fix. However, the full range of metric values is not used.

If the DUT's product line has already been released to the market and if only a specific product version of the DUT is tested, the metric can be utilized as intended: to increase the chance of product certification, it is in the interest of the manufacturer to provide official fixes that reduce the vulnerability's severity. In this case, the metric can be utilized to measure the manufacturer's development progress of a mitigating fix.

2.2.4.3. Report Confidence

The Report Confidence metric "measures the degree of confidence in the existence of the vulnerability and the credibility of the known details" [10]. It holds the values Not Defined, Confirmed, Reasonable, and Unknown. As such, it is primarily applicable to already known vulnerabilities that are based on foreign research the CA has only limited knowledge about. In that case, the metric can be used as intended: to indicate the CA's confidence in a vulnerability's existence. Additionally, if the vulnerability is not yet rated Confirmed and the CA is able to reproduce it, the metric value can be updated by the CA.

For vulnerabilities that are newly discovered by the CA, the original purpose of this metric is restricted. As the CA must be able to consider itself trustworthy, the Report Confidence is typically always rated Confirmed. In that case, the metric is redundant, as it rates every newly discovered vulnerability the same. If the CA is unable to determine the vulnerability's cause and therefore can not confirm its own findings, it can ask the client to provide the information required for confirmation. Only if the client fails to provide this information, the metric can be utilized to indicate the CA's (lack of) confidence in its own findings.

2.3. Weighted Risk Ranking Model

The Weighted Risk Ranking (WRR) model, proposed in [3], is a scoring scheme that "focuses on quantifying the static and dynamic properties of a device, in order to define a

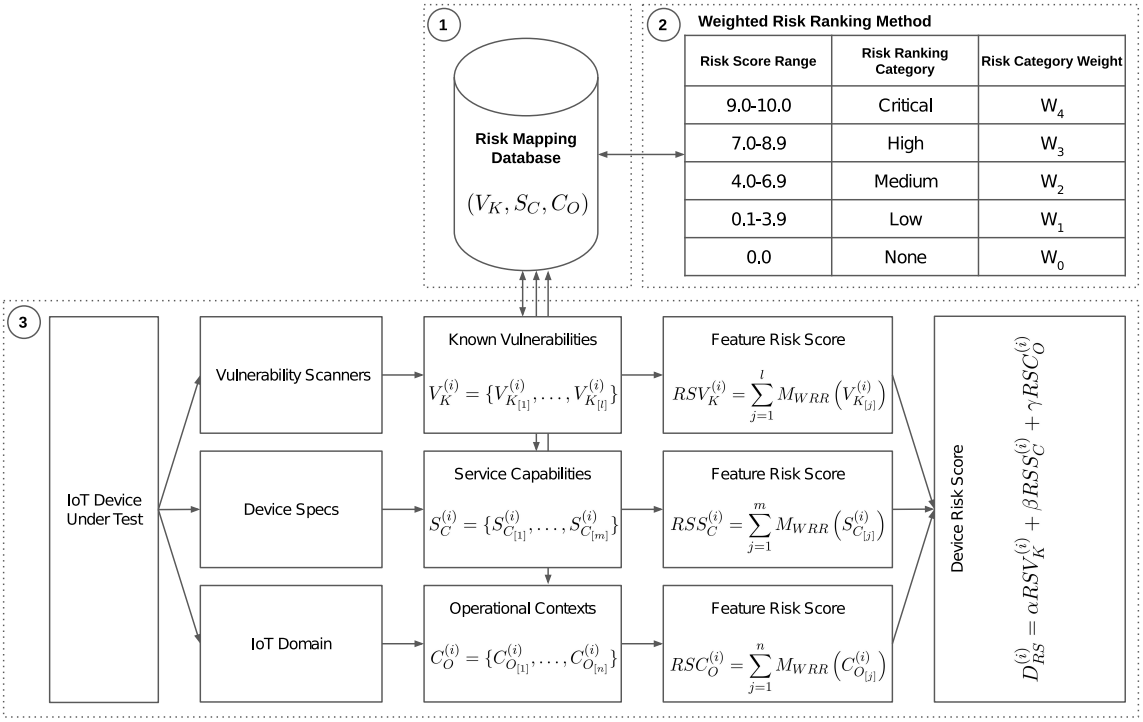

Figure 2.1.: Weighted Risk Ranking Model. Figure taken from [3].

risk score" [3] that determines "the security risks that IoT devices pose to the environment they [are] operated in" [3]. The WRR model represents a scoring system that is deliberately developed for risk scoring complete IoT devices. "The feasibility of the suggested solution as a tool for device risk assessment in modern networks where IoT devices are widely deployed" [3] is demonstrated with a practical Proof-Of-Concept with "several IoT devices in the context of an enterprise network" [3]. The WRR model consists of three components: Risk Mapping Database (RMD), Weighted Risk Ranking Method (WRRM), and Device Risk Score Calculation. These components are described in greater detail in Section 2.3.1.

The main difference of the WRR model to the classic CVSS is that it focuses both on risks – instead of vulnerabilities – and on the security of a complete device – instead of one vulnerability at a time. These characteristics are requirements for utilization in the testing procedure of CAs. Because the WRR model is able to satisfy these requirements, its applicability is worth to be analyzed in greater detail. The analysis performed in Section 2.3.2 draws the conclusion that, while it has certain advantages over CVSS, the WRR model has characteristics that make it unsuited for use by CAs. Chapter 3 attempts to modify the WRR model such that all properties that cause this incompatibility are eliminated.

2.3.1. WRR Model Description

The WRR model (see Figure 2.1) is a device-centric approach that "uses both static and dynamic features of the device [...]. Static features are device specific, and include Known Vulnerabilities and Service Capabilities elements of an IoT device, which do not change over time [...]. Dynamic features are IoT domain-related, and include the contexts in which the IoT device operates" [3]. Along with the WRR model specification, a model implementation that is evaluated with a set of 13 IoT devices is provided [3]. The implementation defines a set of Service Capabilities and Operational Contexts. The set of Service Capabilities includes methods of communication (Wi-Fi, Cellular, Bluetooth, etc.)

Risk Score Range	Risk Ranking Category	Risk Category Weight
9.0 - 10.0	Critical	W_4
7.0 - 8.9	High	W_3
4.0 - 6.9	Medium	W_2
0.1 - 3.9	Low	W_1
0.0	None	W_0

Table 2.3.: Weighted Risk Ranking Method. Table taken from [3].

and sensor types (motion detectors, lighting sensors, microphones, cameras, etc.). The Operational Contexts focus on the DUT's operation time (morning, afternoon, evening, night) and location (server room, meeting room, internal, external, etc.). These elements are assigned scores that represent their level of risk. The third feature (Known Vulnerabilities) is comprised of the set of DUT specific vulnerabilities. Those vulnerabilities are rated by a classic vulnerability scoring system.

Along with the three feature sets, the WRR model defines the model components RMD, WRRM, and Device Risk Score Calculation, that describe the methods for rating and managing the individual features.

2.3.1.1. Risk Mapping Database

The Risk Mapping Database (RMD) is a database that holds entries for the three feature sets Known Vulnerabilities (V_K), Service Capabilities (S_C), and Operational Contexts (C_O) in tables, where each table "contains the possible risk information associated with its elements, along with the appropriate base risk score for each element" [3]. The set of entries associated with an IoT device i are derived from the database, i.e., $V_K^{(i)} \in V_K$, $S_C^{(i)} \in S_C$, and $C_O^{(i)} \in C_O$ [3]. The 3-tuple $(V_K^{(i)}, S_C^{(i)}, C_O^{(i)})$ is utilized by the Device Risk Score Calculation component for computing the risk of device i. RMD is illustrated as component 1 in Figure 2.1.

2.3.1.2. Weighted Risk Ranking Method

The Weighted Risk Ranking Method (WRRM) maps quantitative base risk scores (range: 0.0 to 10.0) to a risk category weight W_i (see Table 2.3). A feature's weighted risk score is computed through function M_{WRR}, which multiplies the feature's base risk score with the weight it is mapped to through WRRM. M_{WRR} is applied to all elements of $(V_K^{(i)}, S_C^{(i)}, C_O^{(i)})$ in the Device Risk Score Calculation for a device i.

The WRRM's weight values are not prescribed by the WRR model specification and can be chosen freely. The WRR model implementation of [3] proposes "a binary-based logarithmic scale (namely, $W_0 = 0$, $W_1 = 2^0$, $W_2 = 2^1$, $W_3 = 2^2$, $W_4 = 2^3$)" that proves to be suitable, if risk rating is designed to prioritize few severe risks over multiple uncritical risks [3]. WRRM is illustrated as component 2 in Figure 2.1.

2.3.1.3. Device Risk Score Calculation

The central element of the WRR model is the Device Risk Score Calculation component. It holds all equations required for computing the device's risk score that is denoted as D_{RS}. For a device i, the risk score is defined as $D_{RS}^{(i)}$. In an initial step, the DUT's 3-tuple $(V_K^{(i)}, S_C^{(i)}, C_O^{(i)})$ is determined with the help of the RMD. Thereafter, the risk score

components $RSV_K^{(i)}$, $RSS_C^{(i)}$, and $RSC_O^{(i)}$ are computed with the help of the WRRM (see Equations 2.2, 2.3, and 2.4).

$$RSV_K^{(i)} = \sum_{j=1}^{|V_K^{(i)}|} M_{WRR}\left(V_{K_{[j]}}^{(i)}\right) \tag{2.2}$$

$$RSS_C^{(i)} = \sum_{j=1}^{|S_C^{(i)}|} M_{WRR}\left(S_{C_{[j]}}^{(i)}\right) \tag{2.3}$$

$$RSC_O^{(i)} = \sum_{j=1}^{|C_O^{(i)}|} M_{WRR}\left(C_{O_{[j]}}^{(i)}\right) \tag{2.4}$$

$D_{RS}^{(i)}$ is composed of the three risk score components (see Equation 2.5). The risk score components themselves are weighted as well: the three constants α, β, and γ assign the individual components a certain importance and thereby influence the resulting device risk score.

$$D_{RS}^{(i)} = \alpha RSV_K^{(i)} + \beta RSS_C^{(i)} + \gamma RSC_O^{(i)} \tag{2.5}$$

In the case of the WRR model implementation in [3], the following values are used: $\alpha = 0.384, \beta = 0.341, \gamma = 0.275$. Those values were retrieved through a questionnaire answered by security researchers: "in total, 23 experts [of varying expertise] answered the questionnaire" [3]. The questionnaire "asked [...] to rank the model's features" and "the service capabilities and Operational Contexts elements of the model" to determine their importance for the risk score.

2.3.2. Applicability of the WRR Model for the IoT and CAs

The WRR model, being deliberately developed for rating the security risks of IoT devices, is a seemingly good candidate for utilization by CAs. Before utilization, the validity of this assertion must be assessed. The sections below investigate the potential advantages and problems of the WRR model.

2.3.2.1. Representation of IoT Specific Risk Categories

The WRR model considers environmental and physical aspects of a DUT. The Service Capabilities feature set is extensible to consider a DUT's physical consequences emerging through a successful attack. In regard to CVSS v2.0, this is a clear advantage: in CVSS v2.0 only little significance is tributed to physical consequences, such as harm of individuals, as determined in Section 2.2.3.1. As for CVSS v3.0, it is possible to indicate that physical consequences exist, but their severity does not have sufficient influence (see Section 2.2.3.2). With the WRR model, both the existence of physical consequences and their severity can be acknowledged through a DUT's Service Capabilities.

A similar benefit exists in the WRR model's Operational Context feature set. While CVSS v3.0 only allows limited rating of environmental security measures, as described in Section 2.2.3.2, such aspects are explicitly considered in WRR scores through the DUT's Operational Contexts. The WRR model implementation in [3] proposes two C_O categories: time of operation and operation location. It assigns all possible combinations of time and location a base risk value in the form of a two-dimensional matrix. That matrix, or the C_O set itself, is easily extended with additional dimensions and features.

Server Room	8.1	7.2	8.0	8.2
Meeting Room	6.4	5.5	6.4	0.0
CxO Office	7.0	6.1	6.9	7.1
IT Department	7.4	6.5	7.3	7.5
Internal	4.8	3.9	4.7	4.9
External	3.5	2.6	3.4	3.6
Contexts	Morning	Afternoon	Evening	Night

Table 2.4.: Operational Context Risk Matrix. Table taken from [3].

Feature Set Extension The researchers of [3] announce a feature set extension in order to be able to consider more risk categories and thereby increase scoring precision. Because this announced extension is not yet published, this section investigates the feature sets' possible rooms for improvement.

The Service Capabilities feature set, proposed in [3], holds 15 entries (including sensors, communication protocols, etc.) and misses ordinary IoT services like Zigbee or Z-Wave. Hence, it can be assumed that this feature set does not sufficiently cover all IoT services. The risk scores of this feature set were computed with the help of expert consultation. As also stated in [3], it is inefficient to reconsult experts each time a new service is to be added. However, the method used for risk calculation requires expert reconsultation and complete recalculation of all risk scores, if just one new feature is added [3]. Before used in practice, an improved risk weighting mechanism should be developed. It is planned to either use Machine Learning and/or Big Data for this problem [3].

The extension of the Operational Contexts feature set is more complex. The currently present Operational Contexts features location and time of operation are represented through a two-dimensional matrix (see Table 2.4). For a single location, the risk value varies based on the time of operation. Extending such matrix with additional features depends on the relations between the individual features. If interdependent, as it is the case for time and location, the introduction of additional dimensions is reasonable. However, for Operational Context features that do not have any relations to each other, the definition of additional dimensions results in a redundant and complex set of risk scores. Such independent Operational Context should not be added to a matrix and instead, just as in the S_C set, introduced as independent feature. The methods of feature extension are not discussed in [3]. Because the existing features time and location are regarded insufficient for correctly representing a device's operational contexts [3], the development of those methods is desirable.

2.3.2.2. Security Scoring Limitations of WRR

Missing Upper Boundary The WRR model aims to rank multiple DUTs by their risks. However, the ranking process is not of high priority to CAs. While it is desirable to be able to compare previously assessed DUTs, such comparison can not be utilized for the decision-making process for certification, as that process requires a solitary risk score that is capable of expressing the security qualitatively. A DUT's Risk Score $D_{RS}^{(i)}$ has no upper boundary and lays in the interval $[0, \infty[$. In contrast, quantitative vulnerability scores of the CVSS lay in the interval $[0, 10]$. The latter allows easier mapping to qualitative values, since qualitative values only need to cover a range limited in its size. This is not the case for WRR scores: devices can have highly varying Device Risk Scores. The example for a WRR score assignment provided in [3] lists a Smart Fridge (Samsung RS757LhQESR/ML) with $D_{RS} = 5.6$ and a Smartphone (LG G4) with $D_{RS} = 55.8$. An IP-Camera, analyzed

in the context of this bachelor thesis's IoT device security assessment, has a Device Risk Score of 195.1 (see "TP-Link TL-SC 3130 IP Camera" in Appendix C.2).

While the complete Device Risk Score does not have an upper boundary, it is possible to define such boundary for one of the feature sets: C_O. The main difference between C_O and the sets V_K and S_C is that every single one of the C_O's features applies to a device: every device has a specific time and location of operation. By following certain design rules, this property can also be achieved for future feature extensions. While $RSV_K^{(i)}$ and $RSS_C^{(i)}$ are influenced both by the score of the individual features and the number of features present in $V_K^{(i)}$ and $S_C^{(i)}$, it holds for every device i: $|C_O^{(i)}| = |C_O|$. Let $C_{O_{[j]}}^{(\max)}$ be the maximal base risk score of the Operational Contexts feature with index j. If the approach of [3] is retained and for every $j \in \{1, \ldots, |C_O|\}$, a maximal $C_{O_{[j]}}^{(\max)}$ (in [3], the maximal score of each feature is 10.0) is defined, a maximal Operational Contexts Risk Score can be determined (see Equation 2.6). By putting $RSC_O^{(i)}$ of device i in relation to $RSC_O^{(\max)}$, the device's Operational Contexts Risk can be interpreted without being dependent on comparisons to the Operational Contexts Risks of other devices.

$$RSC_O^{(\max)} = \sum_{j=1}^{|C_O|} M_{WRR}\left(C_{O_{[j]}}^{(\max)}\right) \tag{2.6}$$

As already determined, this approach can not be applied to V_K and S_C. Though this approach adds information, the WRR has missed the chance of utilizing $RSC_O^{(\max)}$.

Certification Criteria For CAs, the presence of high-risk Service Capabilities or Operational Contexts should not be decisive for the certification process. Devices designed to operate in risky environments must still be certifiable, if they do not have too many or too severe vulnerabilities. Equally, with the number of Service Capabilities, the number of Known Vulnerabilities typically increases, since each service potentially adds vulnerabilities. CAs must take the complexity of a DUT in regard. Under the assumption that highly complex devices have a larger set of vulnerabilities that is still more tolerable than a smaller set of vulnerabilities in a simple device, the WRR model is counter-productive. A risk score is not suited for a certification process that focuses on vulnerabilities, if it increases monotonically with the number of Known Vulnerabilities and Service Capabilities and the riskiness of Operational Contexts. For certification, the relations between RSV_K, RSS_C, and RSC_O must be investigated. A simple summation of these values does not meet these requirements.

2.4. Security Risk Assessment for Certification Authorities

CAs embody unique conditions for IoT security testing. As a third party with active communication with the manufacturer (referred to as client), CAs have elevated access options for vulnerability related proprietary information, that is not available to independent security researchers or adversaries. Simultaneously, CAs do not have access to the full set of information the client has. Depending on the certification requirements and the client's transparency, the degree of insight varies from case to case, resulting in a mix between black box and white box testing. This circumstance is amplified by the fact that, in the case of IoT devices, the option to physically access the DUT impacts the black box assumption, as hardware analysis can provide information about the DUT's contents and implementation [2]. The option to obtain additional information through clients and methods like hardware analysis and thereby shift the testing conditions from black box to white box is barely represented in existing SRA models.

Research in the information security sector has lead to the development of various models and strategies for IoT security testing. The following paragraphs describe the related work in SRA development.

There exists a multitude of SRA models and tools, including CORAS [17], OCTAVE, EBIOS, MEHARI, and CRAMM [18]. Most of them, especially CORAS and OCTAVE, have in common that they are developed to assess the security of an organization's assets and not the security of a single product. While some SRA models are developed for highly specialized environments, such as health care systems [19], a model that suffices the unique requirements for IoT device testing in the context of CAs is missing.

There do exist IoT related SRA models, such as ELK [18], that qualify for assessing the security of IoT infrastructure and single devices. ELK is a practical approach designed to be used continuously to find security risks in real time during production. As this is not the goal of security assessments performed by CAs, who test IoT devices nonrecurringly and outside the production scope, ELK is not a fitting SRA model for CAs.

With [20], a more generic and abstract model is introduced. The model represents an epistemological set of design principles for assessing IoT cyber risks for national high-tech strategies with a focus on recovery planning [20]. While, just as ELK, it is primarily developed for organization level risk assessment and not for security testing conducted by CAs, some of its design principles, e.g. the five primary segments of the Epistemological Framework, are found to be suitable for utilization, if sufficiently adapted.

Models like CRAMM, MEHARI, or EBIOS [18] describe even more generic SRA procedures that can be used as foundation for CAs security testing. However, those models themselves are not sufficient to be used as standalone SRA procedures, as they lack crucial steps and strategies, such as client communication.

Besides SRA models, security research also focuses on the nature of IoT specific security risks. In [21], an exemplary approach for determining security risks in home automation is provided with the help of a practical example that involves a smart lighting system. Through the practical example, different categories of information and protection requirements are identified. By combining these categories, the priority for security requirements of different assets is defined. The approach of [21] serves as an example for risk identification and assessment in the IoT.

3. Security Scoring System

This chapter introduces a new scoring system, the Security Scoring System (SSS). In Section 2.2, the conclusion is drawn that both the Common Vulnerability Scoring System (CVSS) [10] and the Weighted Risk Ranking (WRR) [3] model are insufficient for rating the security of Internet of Things (IoT) devices. This makes the search for a more suited system necessary. SSS aims to rate the overall security of Devices Under Test (DUTs) based on their vulnerabilities and environmental factors with a deliberate focus on devices of the IoT and the applicability through Certification Authorities (CAs). SSS is largely based on the work of [3]. SSS is an adaption to the WRR model (see Section 2.3) that is required to make the WRR model applicable for security scoring.

3.1. SSS Requirements

This section analyzes the requirements and defines the properties, and desired behavior the SSS shall satisfy or possess.

3.1.1. Feature Set Relations

The SSS shall satisfy a set of rules regarding the relations between the individual feature sets. The rules listed below are identified to solve the problems of the WRR model discussed in Section 2.3.2.2:

1. A high RSC_O value indicates an insecure environment that requires increased security precautions. The allowed RSV_K shall decrease, the higher RSC_O is. I.e., the less secure the operational environment, the less severe vulnerabilities are allowed.
2. A high RSS_C value indicates that the device offers multiple and/or dangerous services. A device with a larger service set may hold an increased set of vulnerabilities due to its complexity. By assuming that the V_K set is spread over the single services, the allowed RSV_K value shall increase with the RSS_C value.
3. High RSC_O and RSS_C values indicate additional risk through a large and/or dangerous set of services that operate in an insecure environment. The allowed RSV_K value shall be lower.
4. A high RSS_C and low RSC_O value indicates that the device offers multiple and/or complex services that operate in a secure environment. The RSV_K value is allowed to be higher.

3.1.2. Impact of Risk Weighting on SSS

While it is possible to use the WRR model without any weights (achieved with $W_i = 1$ for every i, denoted as Base Risk Score (BRS) [3]), there are many arguments in favor of risk weighting. The WRR model's risk score components RSV_K, RSS_C, and RSC_O represent the sum of risks (each multiplied with the corresponding weight) for each feature set. If weighting were not applied, features of high risk would not stand out. A feature f with risk score 9 would equal the importance of three features that each have a risk score of 3. However, if using a logarithmic scale like the WRR model implementation of [3], it holds that $M_{WRR}(f) = 72$, which equals the sum of weighted risk scores of 24 features, each having a risk score of 3. Logarithmic weighting allows few severe risks to have higher influence on a DUT's WRR score than multiple modest risks. If CAs focus on addressing the DUT's highest risks, risk weighting is crucial.

The described behavior of weighted risks shall also apply to the SSS. By using a logarithmic Weighted Risk Ranking Method (WRRM), the SSS shall be able to consider the importance of severe risks. If highly severe risks are prevalent in its features and a logarithmic WRRM is used, a DUT shall be rated secure far less likely. By deciding over the WRRM (linear or logarithmic), the CA shall be able to express that its focus either lays on multiple small vulnerabilities, or on few severe vulnerabilities.

3.1.3. Applicability of Simple Fractions

Approach This section discusses the question, if using a simple fraction for scoring a DUT's security based on its risk score components is sufficient for security scoring. Putting the DUT's vulnerabilities in relation to its Service Capabilities and Operational Contexts could be represented by this fraction: $\frac{RSV_K}{RSS_C+RSC_O}$. If this fraction were used for scoring, a smaller value would indicate higher security. However, if security were rated only according to this fraction, an important security factor would be missed: while the fraction recognizes the ratio between RSV_K and $RSS_C + RSC_O$, it is unable to distinguish between devices with identical ratios and different absolute values, as emphasized by the example below.

Example Let i and j be devices. It shall hold that $RSV_K^{(i)} = 10 \cdot RSV_K^{(j)}$ and $RSS_C^{(i)} + RSC_O^{(i)} = 10 \cdot (RSS_C^{(j)} + RSC_O^{(j)})$. Under these conditions, Equation 3.1 holds true. The security scores of i and j are equal, even though j has a vulnerability risk score ten times higher than the one of i. This example shows that neither an increased risk, induced through severe vulnerabilities, nor an increased service capability can be expressed by such fraction. This violates the rules for security scoring defined in Section 3.1.1.

$$\frac{RSV_K^{(i)}}{RSS_C^{(i)} + RSC_O^{(i)}} = \frac{RSV_K^{(j)}}{RSS_C^{(j)} + RSC_O^{(j)}} \tag{3.1}$$

Conclusion A security score function shall not only consider the ratio between RSV_K and RSS_C, and RSC_O. What must be considered, as well, is the sizes of RSS_C and RSC_O: a device with far reaching capabilities or an unprotected physical environment should have a smaller RSV_K to $RSS_C + RSC_O$ ratio than a device with close to zero service capabilities that is located in a secure environment. Therefore, the value of RSV_K, that is required for getting accepted for certification, must also be controlled by the sizes of RSS_C and RSC_O.

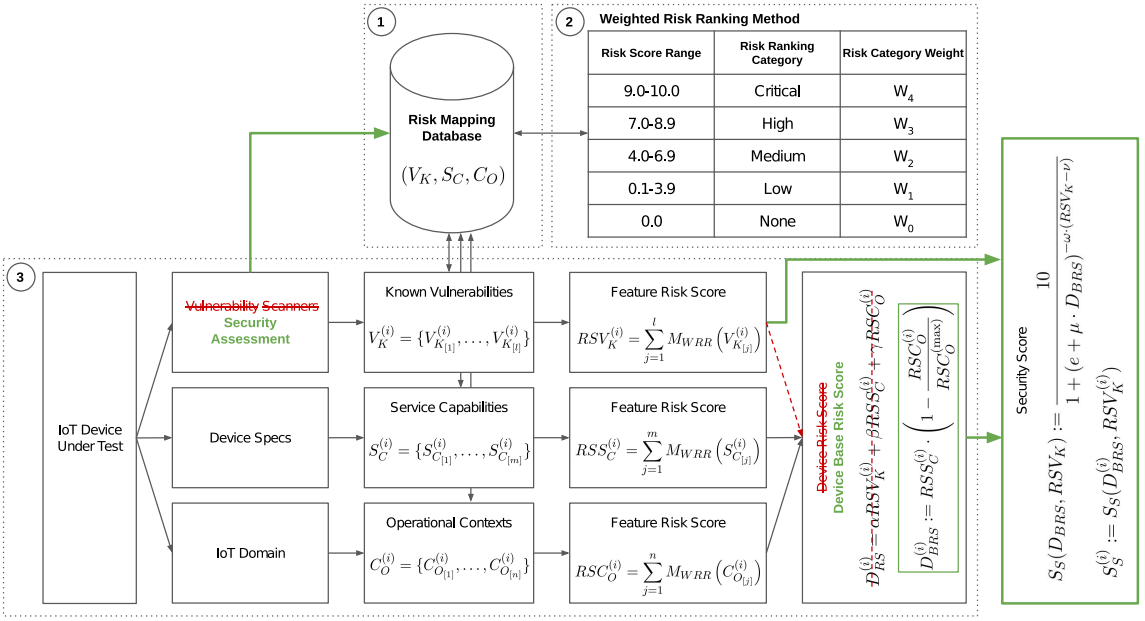

Figure 3.1.: WRR Model (Figure taken from [3]) with Modifications of the SSS Model (red: removal & green: addition).

3.1.4. Finite Score Boundaries

WRR scores lay within $[0, \infty[$, which makes setting a threshold value for certification difficult. CAs shall be able to set a certain score as threshold for deciding over certification: devices with scores atop of that threshold shall be denied certification, devices below the threshold shall be accepted. A finite interval allows CAs to define the threshold as percentage or translate scores to qualitative values and shall be supported by the SSS.

3.2. SSS Specification

The SSS is an adaptation of the WRR model, which replaces components of the Device Risk Score Calculation segment (see Figure 3.1). The component Vulnerability Scanners is replaced by the component Security Assessment, specified in Section 3.2.1. The component Device Risk Score is replaced by the components Device Base Risk Score, and Security Score, specified in Section 3.2.2.

3.2.1. Security Assessment Component

The WRR model assumes that, during risk rating, all of the DUT's vulnerabilities are already known. However, when utilized by a CA, this is not true for all vulnerabilities, as it is the CA's task to uncover new vulnerabilities during the testing procedure. Especially, if the DUT's product line has not yet entered the market, many vulnerabilities can not be known beforehand. To accommodate these circumstances, the WRR model's Vulnerability Scanners component needs to be extended. In SSS, this component is exchanged with a Security Assessment component. This component shall include the Vulnerability Scanner component's features and additionally make use of testing procedures to detect previously unknown vulnerabilities. Newly discovered vulnerabilities must be added to the Risk Mapping Database (RMD), as illustrated in Figure 3.1. Only after the Security Assessment component's tasks are finished, security rating can be conducted.

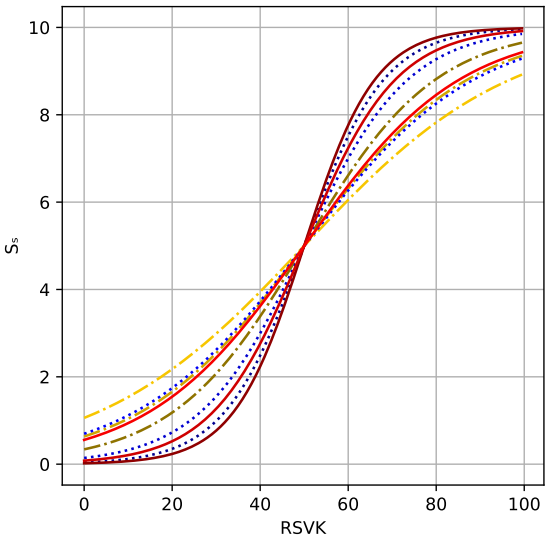

Figure 3.2.: Security Score Limit S_S by RSV_K and D_{BRS}.

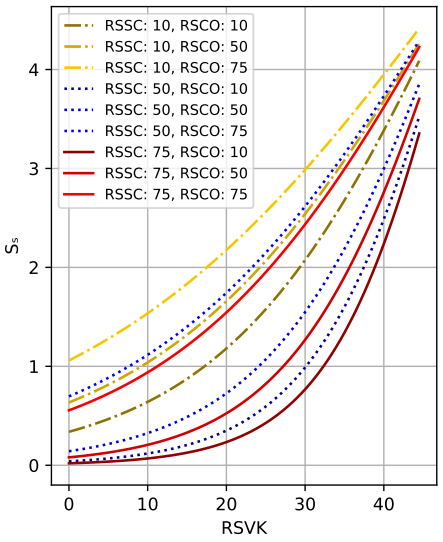

Figure 3.3.: Enlarged X-Axis of Figure 3.2.

3.2.2. Security Score

This section specifies the primary component of the SSS, the security score function. The function is defined in Section 3.2.2.2 and incorporates the Device Base Risk Score introduced in Section 3.2.2.1. In Section 3.2.2.3, an approach for qualitative scoring with the SSS is described.

3.2.2.1. Device Base Risk Score

The SSS introduces the Device Base Risk Score $D_{BRS}^{(i)}$ that is to be computed at the beginning of a testing procedure. The Device Base Risk Score is specified in Equation 3.2. It is composed of the DUT's Service Capabilities and Operational Context features and utilizes $RSC_O^{(\max)}$ (defined in Section 2.3.2.2). The Device Base Risk Score measures the risks of a device's Service Capabilities based on its Operational Contexts. The image of D_{BRS} is $0 \leq D_{BRS}^{(i)} \leq RSS_C^{(i)}$. It holds that $D_{BRS}^{(i)} = RSS_C^{(i)}$, if $RSC_O^{(i)} = 0$, and $D_{BRS}^{(i)} = 0$, if $RSC_O^{(i)} = RSC_O^{(\max)}$. The Device Base Risk Score is not influenced by the DUT's vulnerabilities and is capable of representing the potential dangers caused by the DUT as well as the environmental risks and security precautions.

$$D_{BRS}^{(i)} := RSS_C^{(i)} \cdot \left(1 - \frac{RSC_O^{(i)}}{RSC_O^{(\max)}}\right) \tag{3.2}$$

Note that the Device Base Risk Score is not related to the Element Base Risk Score defined in [3].

3.2.2.2. Security Score Function

The Security Score function $S_S : \mathbb{R}^+ \times \mathbb{R}^+ \to \{x | x \in \mathbb{R} \wedge 0 < x < 10\}; (D_{BRS}, RSV_K) \mapsto S_S(D_{BRS}, RSV_K)$ is defined in Equation 3.3. The Security Score of a device i is labeled as $S_S^{(i)}$ and defined in Equation 3.4. S_S is based on the logistic curve to accommodate the requirement of fixed boundaries (defined as requirement in Section 3.1.4). Setting the numerator of the logistic function to 10 guarantees that the image of S_S is $0 < S_S < 10$. The logistic function is modified such that S_S not only respects the ratio between RSV_K

and D_{BRS}, but also the size of D_{BRS} (defined as requirement in Section 3.1.3). This is achieved by extending the natural logarithm base e with D_{BRS}. The exponent is set to RSV_K.

To allow control over the curve's offset and steepness, S_S makes use of additional constants ω, μ, and ν (see Equation 3.3). Both ω and μ control the curve's steepness. ω influences the impact of D_{BRS} on the steepness, while μ does the same for RSV_K. ν is defined as the turning point of the function and is capable of moving the complete function on the RSV_K-axis (see Figure 3.2). Determining the optimal values for μ, ν, and ω is crucial for obtaining correct security scores. Suitable values are discussed in Section 3.3.

$$S_S(D_{BRS}, RSV_K) := \frac{10}{1 + (e + \mu \cdot D_{BRS})^{-\omega \cdot (RSV_K - \nu)}} \tag{3.3}$$

$$S_S^{(i)} := S_S(D_{BRS}^{(i)}, RSV_K^{(i)}) \tag{3.4}$$

The relations between S_S, D_{BRS}, and RSV_K, are illustrated in Figure 3.2. The graphs of this figure are computed with the constant values defined in Section 3.3. Each curve represents a different value for D_{BRS}. It is observable that the initial slope (from 0 to ν) is lower, the higher the Device Base Risk Score is.

With $RSV_K > \nu$, the slope of the curve drops, as S_S approximates 10. This turning point ν lays in the nature of the sigmoid function. In regard to the S_S, it also serves a purpose, which is illustrated in the following example: it holds for device i that, in the range $\nu < RSV_K^{(i)} < \infty$, $S_S^{(i)}$ is higher, the higher the value of $D_{BRS}^{(i)}$ is. In the range $0 < RSV_K^{(i)} < \nu$, $S_S^{(i)}$ is rated lower, the lower the value of $D_{BRS}^{(i)}$ is. Hence, ν marks a turning point after which a set of highly severe vulnerabilities is less tolerated, the larger the overall risk, induced through high risk Service Capabilities and/or low risk Operational Contexts, is. This behavior serves the following purpose: devices with a high D_{PRS} value are considerably more complex, as they typically have a larger set of service capabilities. Those devices are initially allowed a larger set of vulnerabilities. This tolerated set of vulnerabilities is explained through the wide range of interfaces and functionalities the device offers and that have the potential of each holding a small set of unrelated, moderate vulnerabilities. To account for these vulnerabilities, such devices are rated more secure than devices with the same RSV_K value and a smaller D_{BRS} value. This is true, until the tipping point ν is reached. If $RSV_K > \nu$, a huge RSV_K value is considered worse for a device with large D_{BRS}. The initial tolerance is no longer present and devices with many service capabilities and severe vulnerabilities are considered more insecure. In contrast, devices with a limited set of Service Capabilities and low-risk Operational Contexts approach $S_S = 10$ slower for $RSV_K > \nu$. Those devices have a limited set of Service Capabilities, which means that there are less severe consequences that could be caused by an attack and/or they are only operating in a secure environment, whereby attacks are limited in their probability. This behavior accommodates the requirement that both the ratio between RSV_K and D_{BRS} and the total value of D_{BRS} is considered for security rating (see Section 3.1.3), as the size of D_{BRS} has direct consequences on the tolerance for vulnerabilities, both for $RSV_K < \nu$ and for $RSV_K > \nu$.

3.2.2.3. Security Score Limit

It is intended that CAs can utilize a Security Score Limit S_{SL} with $0 \leq S_{SL} \leq 10$ that serves as threshold for deciding over certification. With the S_{SL}, the two qualitative values Secure and Insecure are introduced. If it holds for device i that $S_S^{(i)} \leq S_{SL}$, then i is considered Secure and certification is granted. Otherwise, i is considered Insecure.

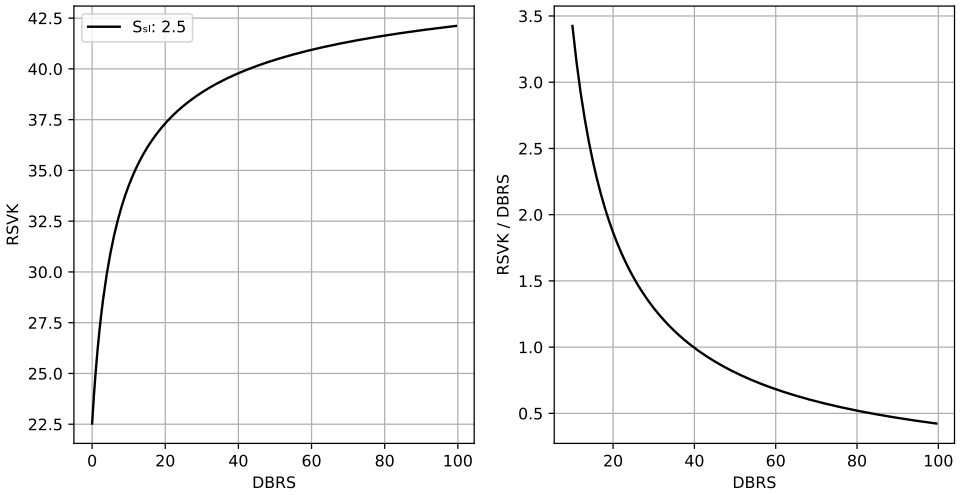

Figure 3.4.: Accepted RSV_K by D_{BRS} for $S_{SL} = 2.5$.

As for certification processes, no further differentiation is required. Therefore, the SSS refrains from defining qualitative scores with more graduation like CVSS or WRR.

Figure 3.4 illustrates the maximal allowed RSV_K values, based on D_{BRS} and a S_{SL} of 2. These maximal values are retrieved through Equation 3.5, which is obtained by solving Equation 3.3 for RSV_K. The left plot of Figure 3.4 allows the observation that, with an increasing D_{BRS}, the tolerance for a large RSV_K increases. However, as already depicted, RSV_K must always be regarded in relation to the size of D_{BRS}. The right plot of Figure 3.4 demonstrates the ratio between the maximum RSV_K and D_{BRS} for $S_{SL} = 2.5$. Though the maximal allowed value of RSV_K increases with D_{BRS}, the allowed ratio between RSV_K and D_{BRS} decreases.

$$RSV_K = \nu - \frac{1}{\omega} \cdot \log_{e+\mu \cdot D_{BRS}} \frac{10 - S_S}{S_S} \tag{3.5}$$

Equations 3.3 and 3.5 are implemented in software (see Appendix 8.2).

3.2.3. SSS Application in Testing Procedures

This section describes, how a testing procedure may utilize the SSS. Before a testing procedure can be started, the RMD and WRRM must be ready for operation. Furthermore, the SSS specific values S_{SL}, ω, μ, and ν must be defined. As part of the testing procedure, the $D_{BRS}^{(i)}$ of DUT i shall be determined in an initial step. With Equation 3.5, the maximum allowed $RSV_K^{(i)}$ value is computed. During testing, the RMD is extended with vulnerabilities of i. After testing is finished, the set $V_K^{(i)}$ is expected to be complete and $RSV_K^{(i)}$ and $S_S^{(i)}$ can be determined. If $S_S^{(i)} > S_{SL}$, the DUT is considered Insecure. Depending on the testing procedure, the client can then be asked to lower the value of $RSV_K^{(i)}$ by mitigating detected risks.

3.3. SSS Constants

The constants μ, ν, and ω are capable of influencing S_S and its behaviour for certain RSV_K and D_{BRS} values. The SSS specification does not define values for those constants,

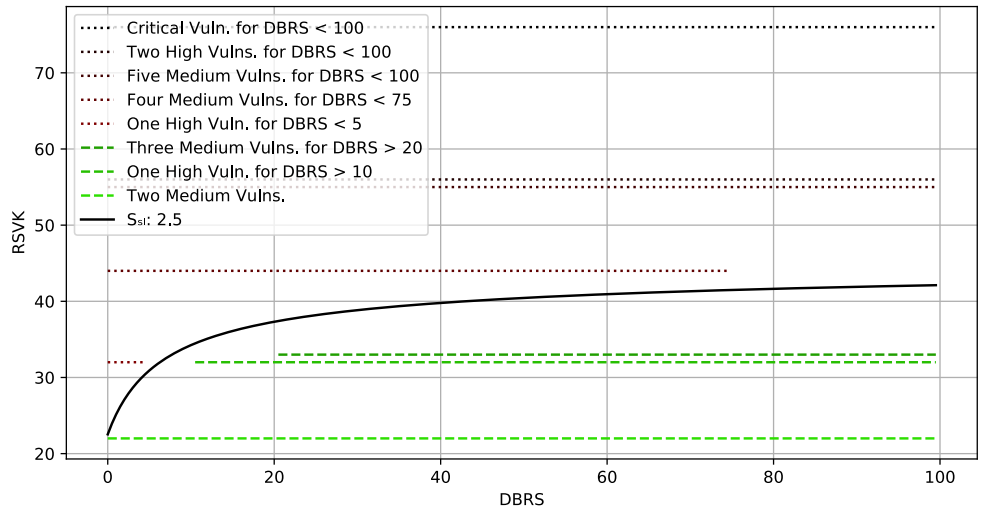

Figure 3.5.: Security Score Limit $S_{SL} = 2.5$ by RSV_K and D_{BRS} with ω, μ, ν defining Rules.

as it is up to the CA to find a configuration that is suited for its testing procedures. Before the SSS is being applied in testing procedures, CAs must define the values for μ, ν, ω, and S_{SL}. Those values may remain unchanged over time in order to guarantee that all DUTs are certified with the same security requirements, i.e., the certificate's quality remains consistent. Finding methods for determining the optimal values for these constants may be subject to future work (see Section 7.4). However, for being able to evaluate the SSS's general applicability, a value set must be defined. This section attempts to find those values by defining rules the S_S shall fulfill. The following rules are determined:

1. A device with a single critical vulnerability is considered Insecure for $D_{BRS} < 100$.
2. A device with two high vulnerabilities is considered Insecure for $D_{BRS} < 100$.
3. A device with five medium vulnerabilities is considered Insecure for $D_{BRS} < 100$.
4. A device with four medium vulnerabilities is considered Insecure for $D_{BRS} < 75$.
5. A device with one high vulnerability is considered Insecure for $D_{BRS} < 5$.
6. A device with three medium vulnerabilities is considered Secure for $D_{BRS} > 20$.
7. A device with one high vulnerability is considered Secure for $D_{BRS} > 10$.
8. A device with two medium vulnerabilities is considered Secure for $D_{BRS} > 0$.

These rules do not only depend on the values of μ, ν, and ω, but also on the value of S_{SL}. The attempt of finding the values of μ, ν, and ω that satisfy the above rules is made according to a security score limit of $S_{SL} = 2.5$. This limit is chosen, because a security score allows high variety between RSV_K and D_{BRS} values, as observable in Figure 3.3. Figure 3.5 illustrates the eight defined rules. Red lines depict the rules 1 to 5 that must be undercut (Insecure) and green lines represent the rules 6 to 8 that are to be exceeded (Secure). Finding μ, ν, and ω was achieved via trial and error and resulted in these values:

$$\mu = 0.3$$
$$\nu = 50.0$$
$$\omega = 0.04$$

These values are applied to $S_{SL} = 2.5$ through Equation 3.5 and illustrated in Figure 3.5 as a graph that is situated between the eight rules.

4. Testing Guide

This chapter introduces the Testing Guide Model (TGM), a Security Risk Assessment (SRA) model for utilization in the Internet of Things (IoT) device certification procedures of Certification Authorities (CAs). The TGM consists of multiple components discussed in Section 4.1 and is extended with a testing procedure defined in Section 4.3. The TGM adds a layer of abstraction to security testing, which allows the components of the lower layer to be replaceable. This enables the TGM to be independent from changes, e.g. caused by latest research in those areas. The layer of abstraction is described in more detail in Section 4.2.

4.1. TGM Segments

The model is split into four major segments: Testing (T), Test Management (TM), Risk Assessment (RA), and Client Communication (CC). These individual segments are identified to be the four major aspects of CAs' testing procedures and to separately lay in the focus of information security research. Each segment consists of a collection of components. The TGM does not define the single components' inner workings, as there already exist multiple solutions developed through prior research for most of them (e.g. Risk Register [22] or Test Management System [23] [3] implementations). This is the case for the components of the segments Testing, Test Management, and Risk Assessment. As stated in Section 2.4, one of the unique CA related tasks is client communication, which has not been considered by past research. Hence, the Client Communication segment and all of its components represent an unprecedented contribution. The same exception applies to the Risk Assessment component Scoring System: because of missing existing solutions, a novel security scoring system for IoT devices is developed in Chapter 3.

This section describes the segments' key aspects and relations and defines the individual components' tasks. Relations are identified with Rx, where x is a number between 1 and 12 (see Figure 4.1).

4.1.1. Testing

The Testing segment holds all mechanisms that are required for running qualitative security tests. It covers all test types, such as penetration testing, documentation review, and static program analysis, and describes the general approach for security testing. That approach includes test preparation, documentation of the test execution and post-execution efforts.

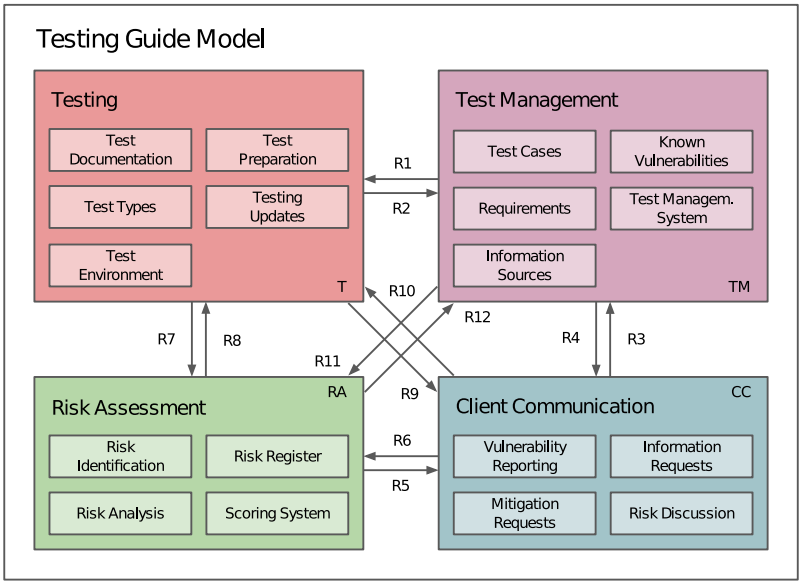

Figure 4.1.: Testing Guide Model

It is mandatory that test documentation is easily comprehensible and repeatable. Each test that leads to the discovery of vulnerabilities must fulfill these requirements. This requirement is rooted in the relations between the different TGM segments: in order to convince clients of the existence of a detected vulnerability (Vulnerability Reporting), CAs must be able to repeatedly demonstrate their findings. Further, qualitative test documentation is required by segment RA to craft precise risk estimations and correct threat models (R1). This also helps in the risk discussion process with clients (R9). Finally, the findings of previously executed tests should be added to the TM segment's systems (R2). While Testing utilizes the knowledge present in those systems (R1), future tests of similar Devices Under Test (DUTs) can be sped up if knowledge, gained through previous tests, is present.

Test Preparation Before a test case is executed, preparatory steps must be conducted. This includes the acquirement of testing tools, preparing the Test Environment, and setting up the DUT's hardware and software components.

Test Environment The Test Environment describes all items required for executing a Test Case. E.g., if a DUT's hardware interface is to be tested, the utilized testing device connected to the interface is part of the Test Environment. To increase efficiency, the similarity of Test Environments may be utilized to determine the order of Test Case execution.

Test Documentation During and after testing, observations made in regard to the DUT and its behavior during testing is to be documented. The documentation helps for vulnerability reporting and reproduction of test results.

Test Types Test Cases can be categorized. Exemplary Test Types are documentation review, static program analysis, and software, hardware, physical, penetration testing, and configuration management testing. Each Type has different characteristics and may be performed by different, specially trained testers.

Testing Updates This component is utilized, if a DUT was tested before and the manufacturer provided an updated product version that has the potential of affecting the DUT's security positively or negatively. If that is the case, the DUT must be

retested. It is inefficient to utilize the complete set of test cases for retesting. Instead, the DUT's components affected by the product update should be identified. Thereby, test cases that do not handle those affected components, can be left out of retesting. This enables a leaner testing procedure.

4.1.2. Test Management

In order to be able to assess a DUT's security, a comprehensive collection of test cases for known vulnerabilities, weaknesses and security requirements must be available. Such collection should be filterable by device type, hardware or software components, and security level. These services are typically provided by a software based Test Management System (TMS). Publicly available and contributable databases, such as Common Attack Pattern Enumeration and Classification (CAPEC) [24], Common Weakness Enumeration (CWE) [25], and Common Vulnerabilities and Exposures (CVE) [26], should be used as information source for a TMS. If a CA promises to keep detected vulnerabilities in clients' products secret and only share information about those with the clients to help absolve their mitigation process, the CA is disallowed to contribute information about those vulnerabilities to publicly managed databases. Instead, the CA must operate and maintain their own, private TMS and contribute new information solely to that system. TMSs are not only utilized during test preparation (R1), their collection of known vulnerabilities can be extended to hold the corresponding security scores. Thereby, they can also be used by segment RA (R11), where the severity of known vulnerabilities needs to be gathered. TMSs are typically capable of generating test reports for completed test executions. Those reports help during communication with clients by providing summarized and/or detailed information about test results (R4).

Test Management System The main component of the test management segment is the Test Management System (TMS). This component can be implemented as a software based service. Its task is to store and provide test cases for security requirements and known vulnerabilities. It further must be able to provide test plans for certification procedures.

Test Cases "Test Cases are a set of test inputs, execution conditions, and expected results developed for a particular objective, such as [...] to verify compliance with a specific requirement" [2].

Known Vulnerabilities All vulnerabilities that are either publicly known or discovered by the CA shall be registered by the TMS. For each vulnerability, one or more test cases are to be created.

Security Requirements If they are not proprietary, certificates are typically issued for a product's compliance to a specific standard, such as Common Criteria [27] or IEC 62443-4-2 [28]. To be capable of testing DUTs for a given standard, the security requirements defined by that standard must be translated to test cases and added to the TMS.

Information Sources Information Sources hold information about vulnerabilities or device characteristics that are required for security testing. Different types of sources exist, e.g. databases on the Internet, or documents provided by the client.

4.1.3. Risk Assessment

Risk Assessment (RA) is an essential part of security testing. It is responsible for correctly determining a DUT's security level and typically utilizes a risk register and a scoring system. These components are populated with information gained through risk analysis and

testing. To achieve precise results, such analysis considers various aspects of a DUT. With regard to the IoT domain, these include device specific vulnerabilities and functionality, network capabilities, operational contexts, etc. [3]. Risk Assessment is exerted before and after test execution, but differs in both use cases. Risk registers created before test execution help in detecting potentially vulnerable aspects that should be tested in greater detail (R8). They further allow the discussion of potential risks with the client before having started expensive testing (R5). After testing is completed, risk estimations are adapted to consider the newly detected vulnerabilities (R7). Thereafter, as part of the decision making process for the DUT's certification, a scoring system is utilized to determine the DUT's security. To reduce the risk analysis task, risk ratings of known vulnerabilities should be fetched through a TMS (R11). Likewise, estimations for newly discovered vulnerabilities that are analysed during risk analysis should be added to the utilized TMS (R12).

Risk Identification Risk Identification is the process of determining a DUT's risks [2]. This can take place in various forms, such as vulnerability discovery during test execution, service analysis, or the assessment of a DUT's target environment [3].

Risk Analysis Risk Analysis typically follows Risk Identification [2] and represents the process of determining an identified risk's probability and severity [2].

Risk Register "The risk register details all identified risks, including description, category, cause, probability of occurring, impact(s) on objectives, proposed responses, owners, and current status" [2].

Scoring System A scoring system assigns numerical values to a device's set of risks or vulnerabilities. Depending on the Scoring System, the computed score expresses the severity of different aspects, such as exploitability, destructive consequences, or loss of confidentiality, integrity, and availability.

4.1.4. Client Communication

It is in the interest of CAs to create long term relationships with clients. Hence, to ensure future collaboration, any irritations or displeasing circumstances must be circumvented in communication with clients. This can prove to be difficult during the testing procedure, where vulnerability reporting and requesting security fixes both criticize the client's product development and create client-side costs in the form of the development of mitigating measures (R9, R10). Also, CAs may see themselves forced to request the client to provide product related, proprietary information (R3, R4), in order to resume testing. This requires a certain level of trust for the CA by the client. To prevent negative customer experience, a comprehensible set of arguments that support the CA's ambitions should always be prepared before any communication with clients.

Vulnerability Reporting If a CA reports a detected vulnerability back to the client, proof for the existence of the reported vulnerability is crucial. To support the proof, CAs shall be able to provide well-documented test execution results as well as a Proof-Of-Concept that allows reproduction or exploitation of the vulnerability and affirms the severity of a hypothetical attack. Vulnerability Reporting is required to ensure that the client understands the nature of the vulnerability and the necessity to remove it.

Risk Discussion CAs must be able to explain hypothetical and discovered risks, such that the client agrees on the CA's risk severity ratings (R5). If the client is unable to comprehend the CA's motives behind the assigned severity for certain risks, the client might become disappointed and desist from future collaboration. Especially in the case of voluntary certificates, such experience must be avoided. It is in the business interest of CAs to report well-founded risks and be able to constructively recommend risk mitigation solutions.

Mitigation Requests CAs typically ask the client to mitigate all discovered vulnerabilities. However, it is to the client to decide, if a mitigation is developed, or if the risk is accepted. In the latter case, the chance for certification is decreased. But, if mitigation efforts are too costly and the vulnerability itself is not too severe, the client may prefer risk acceptance (R6).

Information Requests As determined in Section 2.4, CAs have the unique opportunity to shift from strict black box testing conditions to a white box setting. Their active communication with the manufacturer (client) and the client's dependency on certification can be exploited to request and obtain additional, proprietary information of the DUT. This information can then help in the risk and vulnerability discovery processes of the TGM and confirm or disprove suspected vulnerabilities.

4.2. Layer of Abstraction

The TGM segments, as defined in Section 4.1, each contain a set of components. E.g., the risk register component is part of the Risk Assessment segment. Each component has an individual task and both requires some type of input and provides some form of output. For instance, the task of the risk register is to manage the DUT's risks and their severity [22]. The risk analysis component supplies the risk register with risk descriptions and severity estimations as a form of input. The scoring system utilizes the risk register's output to receive an overview of the DUT's risks in order to determine the DUT's security score. However, the exact way, by which a component's task is performed, is not prescribed by the TGM. As long as the same requirements for the input interface and the same results provided via the output interface are given, the component itself can be interchanged freely. Therefore, the TGM does not define the specific components that are to be used. In the case of the risk register component, this means that the exact register type is not prescribed. As long as the performed task and the input and output type meet the TGM's requirements, any risk register can be used. This freedom of choice represents a layer of abstraction between the TGM and the components that are utilized in practice. When the TGM is implemented for use in practice (see Chapter 5), the benefits and drawbacks of certain components must be discussed to find the ideal combination of components.

4.3. Iterative Testing Procedure

As described in Section 4.1, TGM segments have relations to one another. This section introduces a testing procedure that attempts to combine the segments' relations to facilitate a strategic certification procedure. The procedure, termed Iterative Testing Procedure (ITP), is designed to both reduce redundant tasks and to make the decision making process for certification more comprehensible and consistent. The ITP is built around the Security Scoring System (SSS) (introduced in Chapter 3) that is used for the certification decision making process. The ITP is split into two phases, the setup phase (4.3.1) and the iterative phase (4.3.2). It is illustrated in Figure 4.2. The sections below refer to the flowchart of Figure 4.2 for illustrative purposes.

4.3.1. Initial Phase

The initial phase is started with the acceptance of a client's request for product certification (Init-1). This initial preparation is conducted without running any tests and contains TGM components of the client communication and risk assessment segments. It focuses on the DUT's known characteristics without regard to its vulnerabilities. The phase's goal is to set up an initial risk register (Init-3) and discuss the DUT specific security requirements with the client (Init-4). As illustrated in the flowchart of Figure 4.2, the initial phase does

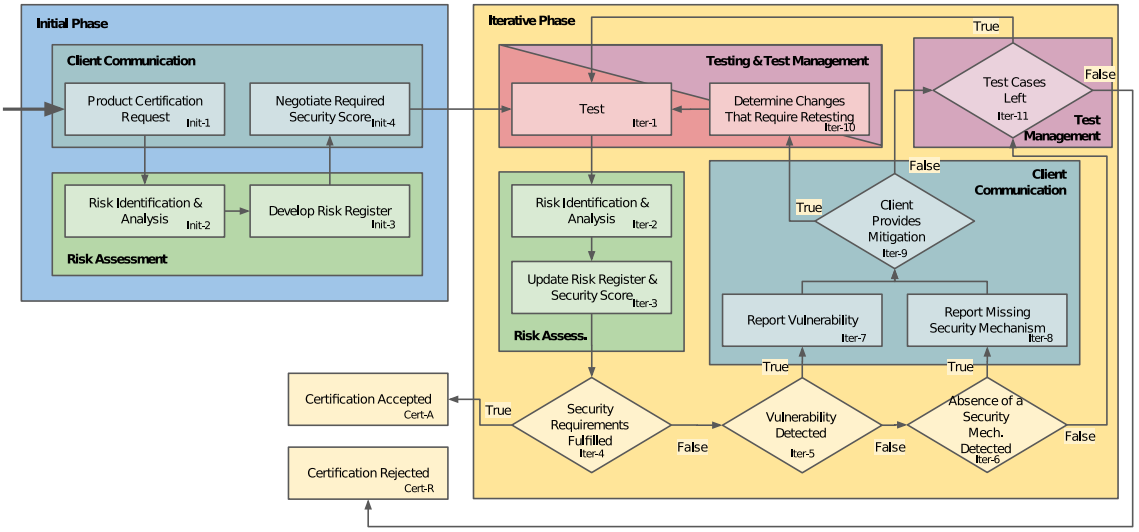

Figure 4.2.: Iterative Testing Procedure

not contain any decision blocks; its four processes are plainly run in the order Init-1 → Init-2 → Init-3 → Init-4.

The succeeding phase is entered, after a required security score is negotiated (Init-4). In the case of SSS, this negotiated score corresponds to the security score limit S_{SL} (defined in Section 3.2.2.3). To keep the security requirements consistent for different DUTs, S_{SL} shall remain unchanged over the course of the CA's individual testing procedures.

Besides S_{SL}, an artificial security score $S_S^{(i)}$ is to be computed for DUT i. This score shall represent the security score of device i for that all security tests have failed and, hence, all vulnerabilities are deemed to be present in i. Under these circumstances, it shall hold that $S_S^{(i)} > S_{SL}$. After $S_S^{(i)}$ and S_{SL} are defined, the initial phase is finished.

4.3.2. Iterative Phase

The iterative phase is started after the initial values $S_S^{(i)}$ and S_{SL} are determined and an initial risk register is developed. This phase requires an operational TMS that is capable of providing all test cases that are required to assess the device's security.

The iterative phase's name originates from the loop that all of its components are part of and can be summarized as a testing procedure that is repeated as long as there are both test cases left to be processed and the DUT's measured security score does not satisfy the required security score.

In every iteration, one test case is executed (Iter-1). If a vulnerability or the absence of a security mechanism is detected (Iter-2), the risk register is updated (Iter-3) to include those findings. Thereafter, the security score $S_S^{(i)}$ is adjusted to consider the new findings and the client is informed about the discoveries (Iter-7, Iter-8). In addition to the report, the client is requested to develop measures that mitigate or remove the security issue (Iter-9). If the client is able to provide an updated version of the DUT that holds the new security measures, a new iteration is started with the updated DUT. Prior to starting a new iteration, the DUT's changes to the previous version are analysed (Iter-10). Thereby, the new iteration can focus on those changes and does not need to retest the complete DUT. But, if the client is unable or unwilling to develop security improving measures, the testing procedure is not finished. It is up to the client to accept risks or provide mitigation.

Path Id.	Path Nodes	Description
Path-1	Iter-1 → Iter-2 → Iter-3 → Iter-4 (T) → Cert-A.	The DUT is deemed secure, the DUT's product line is certified.
Path-2	Iter-1 → Iter-2 → Iter-3 → Iter-4 (F) → Iter-5 (T) → Iter-7 → Iter-9 (T) → Iter-10 → Iter-1	A vulnerability is detected and the client provides mitigation measures that require retesting.
Path-3	Iter-1 → Iter-2 → Iter-3 → Iter-4 (F) → Iter-5 (T) → Iter-7 → Iter-9 (F) → Iter-11 (T) → Iter-1	A vulnerability is detected but the client does not mitigate it. A new iteration for the next test case is started.
Path-4	Iter-1 → Iter-2 → Iter-3 → Iter-4 (F) → Iter-5 (T) → Iter-7 → Iter-9 (F) → Iter-11 (F) → Cert-R	A vulnerability is detected but the client does not mitigate it. The DUT is not ready for certification and there are no test cases left. Certification is rejected.
Path-5	Iter-1 → Iter-2 → Iter-3 → Iter-4 (F) → Iter-5 (F) → Iter-6 (T) → Iter-8 → Iter-9 (T) → Iter-10 → Iter-1	The absence of a security requirement is detected and the client provides mitigation measures that require retesting.
Path-6	Iter-1 → Iter-2 → Iter-3 → Iter-4 (F) → Iter-5 (F) → Iter-6 (T) → Iter-8 → Iter-9 (F) → Iter-11 (T) → Iter-1	The absence of a security requirement is detected but the client does not mitigate it. A new iteration for the next test case is started.
Path-7	Iter-1 → Iter-2 → Iter-3 → Iter-4 (F) → Iter-5 (F) → Iter-6 (T) → Iter-8 → Iter-9 (F) → Iter-11 (F) → Cert-R	The absence of a security requirement is detected but the client does not mitigate it. The DUT is not ready for certification and there are no test cases left. Certification is rejected.
Path-8	Iter-1 → Iter-2 → Iter-3 → Iter-4 (F) → Iter-5 (F) → Iter-6 (F) → Iter-11 (T) → Iter-1	The test case did not lead to the discovery of a vulnerability or unsatisfied security requirement. A new iteration for the next test case is started.
Path-9	Iter-1 → Iter-2 → Iter-3 → Iter-4 (F) → Iter-5 (F) → Iter-6 (F) → Iter-11 (F) → Cert-R	The test case did not find a vulnerability or unsatisfied security requirement. There are no test cases left. Certification is rejected.

Table 4.1.: Possible Paths of a Single Iteration in the Iterative Phase of the Iterative Testing Procedure

If the client chooses risk acceptance, the security risk entry remains in the risk register and the negative impact on the security score's value keeps unchanged. In that case, decision block Iter-11 is entered, leading to certification rejection, if no unprocessed test cases are left. Otherwise, a new iteration is started. Iter-11 is also entered, if test case execution did not lead to the discovery of a vulnerability and exited with certification rejection, if no test cases are left. This condition is based on the circumstance that, in order to be able to enter Iter-11, Iter-4 must have returned False after the last test case is finished, i.e. the security requirements are not met and no test cases are left.

The goal of each iteration is to increase the measured security score's preciseness. With Iter-9, the client is able positively impact the security score by mitigating vulnerabilities or implementing additional security measures. Hence, the measured security score does not change monotonically: with each detected security issue, the score is incremented and with every satisfied security requirement or developed mitigation measure, the score is decremented.

The iterative phase holds decision blocks that, based on the findings of Iter-1, either result in the boolean values true or false. Based on their outcome, each iteration takes one of the nine possible paths. All possible decision block allocations are listed in Table 4.2, where "T" means "True", "F" means "False", and "–" represents an undefined value. A description of each path is presented in Table 4.1. Four paths cause the loop to be exited: Path-1 leads

Path Id.	Iter-4	Iter-5	Iter-6	Iter-9	Iter-11	Result
Path-1	T	–	–	–	–	Certification Accepted
Path-2	F	T	–	T	–	New Iteration
Path-3	F	T	–	F	T	New Iteration
Path-4	F	T	–	F	F	Certification Rejected
Path-5	F	F	T	T	–	New Iteration
Path-6	F	F	T	F	T	New Iteration
Path-7	F	F	T	F	F	Certification Rejected
Path-8	F	F	F	–	T	New Iteration
Path-9	F	F	F	–	F	Certification Rejected

Table 4.2.: Logic Table for ITP Decision Blocks

to the acceptance of the product certification request, while Path-4, Path-7, and Path-9 imply that the product is not getting certified.

4.3.3. Deciding Over Certification

The iterative phase is exited after the question, if the DUT's product line is accepted for certification, is answered positively through Iter-4 or negatively through Iter-11. The main decision process is located in Iter-4: as soon as all test cases are finished and the DUT proves to satisfy the security requirements, Iter-4 returns true and leads to the certification acceptance (Path-1). If the DUT does not satisfy the security requirements, Iter-4 returns false (Path-2 to Path-9). If no mitigation is performed by the client, Iter-11 is entered (Path-3, Path-4, Path-6, Path-7, Path-8, and Path-9). Entering Iter-11 requires Iter-4 to have returned false and the client to have declined the request for mitigation measures. If no test cases are left, the DUT is not qualified for certification and Iter-11 returns false, resulting in the DUT failing the ITP (Path-4, Path-7, and Path-9).

The main decision process takes place in Iter-4. If TMS is used for testing a DUT for the compliance to a specific standard, the decision of Iter-4 is dependent on the standard's prescriptions. If a scoring system, such as SSS is utilized, rating security is performed according to that system.

4.3.4. Parallelism

The ITP model prescribes that process Iter-1 is left after a test case is completed. If a vulnerability or an unfulfilled security requirement is detected, the client is asked for mitigation measures. It is inefficient to pause testing and wait for the client to develop and provide such measures before continuing. Hence, this section extends the ITP model to support parallelism: instead of running a single iteration at a time, multiple iterations for different test cases are run in parallel. For all test cases that do not depend on each other, an individual loop should be run. E.g., with the transition from Init-4 to Iter-1, the set of test cases that are not interdependent should be identified and for each test case, a separate loop is started. With the transition from Iter-10 to Iter-1, the set of independent test cases, that is (because of the changes made by the client) required to be retested, should be identified and for each test case, a separate loop is started. Through those parallel test processes, the iterative phase is split into multiple entities. Because the mitigation development no longer blocks the entire testing procedure, the set of test cases is worked off in a shorter timespan.

Executing test cases in parallel is restricted to test cases that do not have reciprocal relations. If a test case A has relations to, or is dependent on another test case B and

B is part of an uncompleted iteration, A should be delayed, until the iteration for B is finished. After B is completed, it shall be checked if A must still be run. This rule is defined with the aspect of reduction of redundant labor in mind, as the following example demonstrates: Let A and B be test cases. A is designed to check the security aspects of a specific hardware interface and B tests, whether there are redundant debugging hardware interfaces that are not deactivated. B draws the conclusion that the interface that would be tested by A should be deactivated. Hence, if the client provides an update that disables the interface, A does not need to be executed anymore. However, if the client refuses to deactivate the hardware interface, A must be executed. Because the outcome of Iter-9 for test case B is not known in advance, delaying the (potential) execution of A to the point, where B is completed, reduces redundant labor.

If parallelism is utilized, the decision block Iter-11 must be adapted to respect the possibility that some tests are unfinished. With parallelism, it is possible that Iter-11 returns false, even though there are unfinished iterations. Iter-11 must be adjusted to serve as a sink for iterations, for which it is true that no test cases are left, but unfinished iterations still exist.

5. Implementation

This chapter describes the implementation of a set of Testing Guide Model (TGM) components in software. This includes the components Risk Register, Scoring System, Test Management System (TMS), Test Cases, Requirements, Information Sources, Vulnerability Reporting, Test Documentation, Test Preparation, Test Environment, and Update Testing.

The TGM's software implementation is publicly available at `https://gitlab2.informatik.uni-wuerzburg.de/s354363/bachelor-thesis-code` [29].

5.1. Test Management System

As specified in Section 4.1.2, the TMS holds information about Test Cases, Known Vulnerabilities, Risk Scores, Security Requirements, and Test Preparation. The test management software TestLink (TL) is found to be suited for these tasks and chosen for implementation.

TestLink is an open source test management and execution tool [23]. It holds test cases that are executed according to a test plan and produces test reports of finished test plans. Not all TestLink components have directly equivalent TGM components. E.g., TestLink holds a Test Plan component that could be utilized to represent the TGM Testing Procedure. To fully comply with TGM, these TestLink components must be translated to TGM components, as described below.

Test Plan : Testing Procedure The TL component Test Plan is is a collection of test cases that are to be executed in the context of a testing procedure.
Relation to TGM: All Test Cases that apply to a Device Under Test (DUT) are collected in a Test Plan. During the testing procedure, the testing is executed according to the Test Plan. Each Test Plan is linked to a DUT.

Custom Fields : Security Score & Test Type The TL component Custom Fields holds properties for test cases and requirements. Test cases and requirements can be filtered by these properties.
Relation to TGM: Custom Fields are utilized to hold Common Vulnerability Scoring System (CVSS) scores and vector strings for test cases that cover known vulnerabilities. Another custom field is utilized to describe a test case's Test Type.

Keywords : Test Environment & Test Preparation The TL component Keywords is assigned to Test Cases. Just as Custom Fields, Test Cases can be filtered by Keywords.

Relation to TGM: The Keywords component is utilized to represent the Test Environment and Preparation steps that are required by test cases. This allows Test Cases with the same Test Environment to be executed consecutively.

Test Report : Vulnerability Reporting The TL component Test Report summarizes passed and failed test cases.
Relation to TGM: The Test Report component is utilized to summarize detected vulnerabilities for Vulnerability Reporting and collecting the present vulnerabilities for Security Scoring.

Build : DUT The TL component Build is related to a Test Plan and identifies entities (and their statuses and versions) that are to be tested according to the Test Plan.
Relation to TGM: The Build component is utilized to identify the DUTs' versions. For each new product version provided by the client, a new Build is created.

Test Execution : Test Documentation On Test Execution, the assigned tester performs the test steps as defined by the current Test Case. For each step, the tester is allowed to store test notes.
Relation to TGM: The Test Execution component is utilized both for performing the test procedure and to do Test Documentation. For documentation, the tester is shall store all information in TestLink's test notes feature.

5.1.1. Test Cases

In order to be able to precisely evaluate the applicability of the TGM, the implemented TGM is specialized to consider a certain type of Internet of Things (IoT) devices. Namely, it is decided to specialize on IoT devices that support the wireless communication standard Zigbee. This allows the TGM evaluation to be performed with real devices, scenarios and vulnerabilities. The specialization primarily influences the set of Test Cases and shall not prevent future TGM extension for different device types. In addition to Zigbee specific Test Cases, the Requirements developed by the IoT Security Foundation (IoTSF), specified in [30], are implemented. These focus on IoT security in a more generic approach and are suited for every IoT device type. For each IoTSF requirement, a separate Test Case is created. For each vulnerability, a CVSS v3.1 score is calculated and assigned to the Test Cases via Custom Fields. The implemented TGM holds 12 Zigbee related and 199 IoTSF related Test Cases [29].

5.1.2. Creating Test Plans

For each new DUT, a separate Test Plan shall be created. TestLink allows new Test Plans to inherit from existing ones. To simplify Test Plan creation, Template Test Plans are developed that hold all Zigbee and/or IoTSF Test Cases. New Test Plans shall inherit from these templates accordingly.

5.1.3. Updating Test Cases

During security assessment, previously unknown vulnerabilities may be discovered. In this case, a new test case shall be added to TestLink and all affected (template) Test Plans. The Custom Fields that are used for risk analysis (CVSS score, vector string) shall be filled to allow later security scoring.

5.1.4. Testing Updates

If the client provides an updated version of the product that is to be certified, or if the client requests certification for a certified product's successor, device components, that are not

affected by the version change, shall not be retested. As described above, a separate Build is created for each product version. On Test Plan execution, TestLink offers the option to filter by passed or failed Test Cases of previous Builds. With this filter, previously failed Test Cases (i.e. detected vulnerabilities) can be determined. If the new product version declares to have fixed vulnerable components, the corresponding filter should be used. On the other hand, if the new product version contains additional functionality, previously passed relevant Test Cases (i.e. determined to not be vulnerable) shall be filtered and retested.

5.1.5. Information Sources

TestLink is filled with Test Cases for multiple vulnerabilities. Test Cases for generic IoT security issues are created based on the requirement specifications of [30]. For device specific vulnerabilities, the Common Vulnerabilities and Exposures (CVE) database National Vulnerability Database (NVD) [14] is used.

The Zigbee related vulnerabilities were identified in the context of extended research on Zigbee security performed by this thesis's author. This research is based on the existing work on Zigbee security conducted in in [31], [32], [33], [34], [35], and [36]. The vulnerabilities are embedded into Test Cases and added to the TestLink system.

5.2. Risk Register

A simple chart file is used for the Risk Register. This allows easy exchange and collaboration with clients. For maximum compatibility, the Open Document Sheet format is utilized. The risk register holds all S_C, C_O, and V_K entries and follows the approach of [22]. A risk register template is available at [29].

5.3. Scoring System

The Security Scoring System (SSS), introduced in Chapter 3, is implemented in software, using the Python programming language (see Appendix 8.1 & 8.2). It is available for installation at [29] and can be used either as Python module or for scoring a device's Service Capabilities, Operational Contexts, and Known Vulnerabilities that are stored in a specially formatted file. For this purpose, the file format YAML Ain't Markup Language (YAML) is chosen, because it is human readable [37] and allows quick editing after each risk register update.

To consider potential Operational Contexts extensions, as discussed in Section 2.3.2.1, the program allows the definition of additional C_O risk features.

See Listing 5.1 for an exemplary device risks file (for surveillance camera Bosch NBN-498 Dinion 2X). The computed results for this file are illustrated in Listing 5.2.

Listing 5.1: YAML Input File.

```yaml
# Bosch NBN-498 Dinion2X
service-capabilities:
  - camera
  - mic
  - ethernet
  - motion_detector
operational-contexts:
  time:
    - morning
    - afternoon
    - evening
    - night
  location:
    - server-room
    - it-department
    - internal
known-vulnerabilities:
  - id: cve-2015-6970
    score: 9.8
```

Listing 5.2: Security Score Program Output

```
RSSC: 34.5
RSCO: 32.8
RSVK: 78.4
DRS: 50.9
Security Score: 9.22
```

The developed SSS program requires the separate specification of risk values for the feature sets Service Capabilities and Operational Contexts. If executed on the command line, these values are provided through an additional YAML file [29].

6. Evaluation

In this chapter, the Testing Guide Model (TGM) and the Security Scoring System (SSS) (introduced in Chapter 3 and 4, implemented in Chapter 5), are evaluated. In Section 6.1, the TGM is utilized to assess the security of two Internet of Things (IoT) devices in order to determine the model's operability for Certification Authorities (CAs). In Section 6.2, the SSS is assessed through a score ranking comparison with the existing models Common Vulnerability Scoring System (CVSS) and Weighted Risk Ranking (WRR) discussed in Chapter 2.

6.1. TGM Evaluation

The TGM, implemented in Chapter 5, is evaluated by testing the security of two IoT devices according to the Iterative Testing Procedure (ITP). It is assessed, whether previously determined vulnerabilities are detected and correctly processed by the TGM. The two devices are very similar in their Service Capabilities, Operational Contexts, and Known Vulnerability ratings. This evaluation assesses, if the TGM successfully produces similar results for both devices.

6.1.1. Testbed

The two Zigbee enabled gateways Philips Hue Bridge v2 (European article number: 046-677458478), denoted as device i, and Osram Lightify Home (European article number: 4052899926172), denoted as device j, are chosen for evaluation. Both gateways serve similar tasks, as they are intended to serve as interface between Zigbee connected light bulbs and light switches. They are connected to the Internet, where they communicate with backends to allow users to control light bulbs through mobile apps.

Both devices, i and j, have a set of known vulnerabilities. This set is based on Common Vulnerabilities and Exposures (CVE) entries of the National Vulnerability Database (NVD) [14] and on Zigbee security research conducted for i and j [35]. All vulnerabilities that are to be detected are listed in Table 6.1. The table displays, which vulnerabilities apply to which device and indicates, which vulnerabilities are fixed or unfixed and therefore should or should not be considered in the testing procedures' final results. Vulnerabilities that have already been fixed by the manufacturer are initially treated as unfixed, simulating the client side mitigation process of Iter-9 (see Figure 4.2). This allows the utilization and assessment of Path-2 and Path-5 (see Table 4.1) of the ITP.

Device	Summary / Identification	Description	Status
i	Touchlink Bug	A bug enables Touchlink Commissioning beyond the allowed physical ranges [35].	Fixed
i, j	Factory Reset	Network configuration can be reset with inter-PAN command frames [35].	Unfixed
i, j	Default Global Link Keys	The default values for the Zigbee Global Link Key are used.	Mitigated for i, unfixed for j
i	Root Access through Hardware Attack	By setting a hardware pin to ground, a root shell can be accessed through a hardware interface [38].	Unfixed
i	CVE-2020-6007	Heap buffer overflow allows remote code execution [39].	Fixed
j	CVE-2016-5054	Because the Zigbee Frame Counter is not checked, replayed packets are falsely accepted [40].	Fixed
j	CVE-2016-5053	Unauthenticated command execution in local networks [40].	Fixed

Table 6.1.: Vulnerabilities of Philips Hue Bridge v2 (i) and Osram Lightify Home (j).

The TGM, implemented in Section 5 and used for this evaluation, holds 217 test cases. The majority of those does not apply to the present vulnerabilities (see Table 6.1). A full security test that includes all test cases is deemed too costly and also not required for this evaluation. Instead, only the test cases that are known to be related to the present vulnerabilities are executed and discussed in the sections below. The Test Case descriptions are part of the Test Management System (TMS) that is available at [29] and also listed in Table B.1.

6.1.2. Initial Phase

During Initial Phase, risk analysis is conducted. For the sake of compatibility with the SSS model, risks are distinguished between service capability (S_C), operational context (C_O), and vulnerability (V_K). For risk identification, all available device related documents, such as product manuals [41] [42], are analyzed and the devices themselves physically observed.

Operational Contexts Both gateways are available on the consumer market and do not exhibit features that allow integration into existing solutions. Hence, utilization in environments of high risk, such as Server Rooms, is determined unlikely. As neither one is supposed to be operated outdoors, the operational context External is ruled out, as well. The remaining options (Meeting Room, CxO Office, IT Department, and Internal) are deemed probable.

Because controlling light bulbs is a task that is not bound to the time of day, the time of operation is not specific. IT Department and Night is identified as the combination of time and location that is linked to the highest risk value, 7.5, and therefore chosen as the Operational Contexts risk (see Table 2.4). Hence, for both devices, it holds that $RSC_O = M_{WRR}(7.5) \cdot 7.5 = 30.0$ (see Equation 2.4).

Service Capabilities Equal to the operational contexts, the service capabilities set is adopted from [3]. Of the service capabilities listed in [3], the Osram Lightify Home

gateway is identified to hold WiFi capabilities and the Philips Hue Bridge v2 has both WiFi and Ethernet capabilities. Additionally, it is discovered that both devices support Zigbee communication and can be controlled through mobile apps. However, the service capabilities Zigbee and App are not listed in [3] and therefore require additional steps to identify and analyze associated risks. Unlike the risk scoring conducted in [3], the risk scoring for Zigbee and App is not performed through a questionnaire filled out by IoT security experts. Instead, the existing service capability risks are used as a benchmark to rate the services Zigbee and App. The following additional service capabilities are created:

Zigbee The IoT device supports the wireless communication standard Zigbee. Risk score: 8.5

App The IoT device interacts with mobile Apps. Risk score: 7.0

Through the conducted analysis, the following S_C sets are determined:

- Philips Hue Bridge v2: WiFi (9.5), Ethernet (4.3), App (7.0), Zigbee (8.5). $RSS_C^{(i)} = 146.6$
- Osram Lightify Home: WiFi (9.5), App (7.0), Zigbee (8.5). $RSS_C^{(j)} = 138.0$

Known Vulnerabilities Initially, neither of the devices has known vulnerabilities. However, to follow the prescription of the ITP, the opposite is assumed: all vulnerabilities are present until proven otherwise (see Section 4.3.1). The TMS is utilized to obtain all vulnerabilities that are to be considered.

Security Score Negotiation After the initial risk analysis is finished, the accepted security score S_{SL} (see Section 3.2.2.3) is negotiated with the hypothetical clients. To guarantee fair treatment, the same security score limit is used for both gateways: $S_{SL} = 2.5$. In particular, in order to be scored Secure qualitatively (see Section 3.2.2.3), the security score of each gateway shall be equal to, or below 2.5. This security score limit is chosen to comply with the rules defined in Section 3.3.

Device Base Risk Score With the Service Capabilities and Operational Contexts feature sets prepared, the gateways' Base Risk Scores (see Section 3.2.2.1) can be computed (see Equations 6.1, 6.2).

$$D_{BRS}^{(i)} = RSS_C^{(i)} \cdot \left(1 - \frac{RSC_O^{(i)}}{RSC_O^{(\max)}}\right) = 146.6 \cdot \left(1 - \frac{30}{80}\right) = 91.625 \tag{6.1}$$

$$D_{BRS}^{(j)} = RSS_C^{(j)} \cdot \left(1 - \frac{RSC_O^{(j)}}{RSC_O^{(\max)}}\right) = 138 \cdot \left(1 - \frac{30}{80}\right) = 86.25 \tag{6.2}$$

6.1.3. Iterative Phase

As prescribed in Section 4.3.2, test iterations are executed consecutively until there are either no test cases left or the device is deemed secure. For the sake of simplicity, but without the loss of generality, this evaluation does not utilize the parallelism feature defined in Section 4.3.4.

The conducted iterations are summarized in Table 6.2 and Table 6.3. The included Path identifiers are taken from Table 4.1 and refer to the blocks illustrated in Figure 4.2. The individual paths taken in each iteration rely on the corresponding Test Case's outcome and the subsequent clients' actions. The Test Case identifiers are taken from the implemented TMS of [29]. See Appendix B.1 for descriptions of all utilized Test Cases.

# Iteration	Path Id.	Test Case Id.	Test Result
1	Path-8	IoT-6: ZB Default Global Link Key	Preconditions not met
2	Path-8	IoT-7: ZB Fallback Global Link Key	Test Case passed
3	Path-8	IoT-8: ZB Frame Counter Unchecked	Test Case passed
4	Path-8	IoT-9: ZB OTA Link Key Transmission	Test Case passed
5	Path-8	IoT-10: ZB DoS Association Flooding	Test Case passed
6	Path-8	IoT-11: ZB Security Level	Test Case passed
7	Path-2	IoT-12: ZB Touchlink range limited	Test Case failed
8	Path-8	IoT-12: ZB Touchlink range limited	Test Case passed (after mitigation)
9	Path-8	IoT-13: ZB Firmware Update	Test Case passed
10	Path-8	IoT-14: ZB Identify Action	Test Case passed
11	Path-3	IoT-15: ZB Factory Reset	Test Case failed
12	Path-8	IoT-16: ZB Hijacking	Test Case passed
13	Path-8	IoT-17: ZB Firmware Update Impossible	Test Case passed
14	Path-2	IoT-18: CVE-2020-6007	Test Case failed
15	Path-8	IoT-18: CVE-2020-6007	Test Case passed (after mitigation)
16	Path-4	IoT-23: Root Access	Test Case failed

Table 6.2.: ITP Iterations for Philips Hue Bridge v2.

Philips Hue Bridge v2 This device has had two Zigbee related vulnerabilities in the past (assessed with test cases IoT-12 and IoT-18) that were fixed through software updates. For evaluation, these software fixes are treated as client-side mitigation measures that are followed by another iteration that retests the fixed device (e.g., see iterations 7 and 8 in Table 6.2). For each fix, a new Build is created in the TestLink software. The produced Test Reports of each Build are available at [29].

The only two known vulnerabilities that are, to this day, unfixed, are related to a Denial of Service (DoS) attack that can be executed during a 30 second time period after a certain button is pressed on the gateway [35] (assessed with test case IoT-15) and to a hardware vulnerability during the boot process, which allows root access (assessed test case IoT-23) [38] (see iterations 11 and 16 in Table 6.2). The absence of security fixes for those vulnerabilities is interpreted as risk acceptance (represented by paths Path-3 and Path-4) and negatively impacts the security score. Other vulnerabilities, e.g., the vulnerabilities assessed with test cases IoT-14 and IoT-16, are present in Philips Hue light bulbs that can be connected to this gateway, but not in the gateway itself [35]. If the Philips Hue system as a whole were tested, these vulnerabilities should be considered. Because the assessment of the complete system is not in the scope of this testing procedure, these vulnerabilities are ignored.

With IoT-15 (associated CVSS score: 7.4) and IoT-23 (associated CVSS score: 7.6) being the only failed test cases, it holds that $RSV_K^{(i)} = M_{WRR}(7.4) \cdot 7.4 + M_{WRR}(7.6) \cdot 7.6 = 60.0$ (see Equation 2.2). The rounded security score $S_S^{(i)}$ of Philips Hue Bridge v2 is 8.0 (see Equations 6.1, 6.3). It holds that $S_S^{(i)} > S_{SL}$ and hence, the iterative phase is correctly exited with Path-4, resulting in certification rejection.

$$S_S^{(i)} = \frac{10}{1 + (e + \mu \cdot 91.625)^{-\omega \cdot (60.0 - \nu)}} \approx 8.0 \tag{6.3}$$

# Iteration	Path Id.	Test Case Id.	Test Result
1	Path-3	IoT-6: ZB Default Global Link Key	Test Case failed
2	Path-8	IoT-7: ZB Fallback Global Link Key	Test Case passed
3	Path-8	IoT-8: ZB Frame Counter Unchecked	Test Case passed
4	Path-8	IoT-9: ZB OTA Link Key Transmission	Test Case passed
5	Path-8	IoT-10: ZB DoS Association Flooding	Test Case passed
6	Path-8	IoT-11: ZB Security Level	Test Case passed
7	Path-6	IoT-12: ZB Touchlink range limited	Preconditions not met
8	Path-8	IoT-13: ZB Firmware Update	Test Case passed
9	Path-8	IoT-14: ZB Identify Action	Test Case passed
10	Path-3	IoT-15: ZB Factory Reset	Test Case failed
11	Path-8	IoT-16: ZB Hijacking	Test Case passed
12	Path-8	IoT-17: ZB Firmware Update Impossible	Test Case passed
13	Path-2	IoT-137: Authent. for Web Interfaces & IoT-20: CVE-2016-5053	Test Case failed
14	Path-9	IoT-137: Authent. for Web Interfaces & IoT-20: CVE-2016-5053	Test Case passed (after mitigation)

Table 6.3.: ITP Iterations for Osram Lightify Home

Osram Lightify Home Unlike the Philips Hue Bridge v2, the Osram Lightify Home Gateway can not initiate Touchlink commissioning [35], a Zigbee security mechanism that reduces transmission power during key exchange. This means that the Default Global Link Key and the Zigbee Network Key, a network wide encryption key that must remain secret at all times [43], are transmitted within a large radius, allowing adversaries to eavesdrop [44]. This is a major security issue causing test case IoT-6 to fail. Iteration 7 simulates the client request to implement this missing security mechanism. Because it was not implemented, Path-6 (client side risk acceptance for an absent security mechanism) is taken.

In addition to the missing security mechanism, the Osram Lightify Home gateway holds the same vulnerability as Philips Hue Bridge v2: both gateways accept the inter-PAN command that performs a factory reset (IoT-15). While the Philips Hue Bridge v2 gateway is only prone to such attack during a specific time span, the Osram Lightify Home gateway can be reset at any time [35].

There are multiple Zigbee related vulnerabilities that were present in Osram Lightify light bulbs (IoT-8, IoT-14, IoT-15, IoT-16) but not in the gateway itself [35]. It could be argued that these influence the availability and integrity of the gateway's services, which should be considered, if the Osram Lightify system as a whole was tested. Because only the security of the gateway itself is assessed, these security issues are left out.

Beyond Zigbee security, the Osram Lightify Home gateway has had a web service related security issue (CVE-2015-5053) that was fixed through software updates [40].

With iteration 14, all test cases are finished and the security score is computed. With IoT-6 (associated CVSS score: 7.4) and IoT-15 (associated CVSS score: 7.4) being the only failed test cases, it holds that $RSV_K^{(j)} = M_{WRR}(7.4) \cdot 7.4 \cdot 2 = 59.2$ (see Equation 2.2). Hence, the rounded security score $S_S^{(j)}$ of Osram Lightify Home is 7.7 (see Equations 6.2, 6.4). It holds that $S_S^{(j)} > S_{SL}$ and the iterative phase is correctly exited with Path-9,

resulting in certification rejection.

$$S_S^{(j)} = \frac{10}{1 + (e + \mu \cdot 86.25)^{-\omega \cdot (59.2 - \nu)}} \approx 7.7 \tag{6.4}$$

6.1.4. Evaluation Results

The evaluating testing procedure follows the prescriptions of the TGM and was executed successfully. The TGM implementation (TMS, Risk Register, and Scoring System) [29] is able to accomplish the required tasks and contributed to the successful and correct security assessment of the two Zigbee gateways. Both devices have similar Service Capabilities, Operational Contexts, and vulnerability ratings. These similarities were correctly translated to security scores of almost even values ($S_S^{(i)} = 8.0$, $S_S^{(j)} = 7.7$).

Minor issues in the introduced TGM were discovered during execution that were not considered during the development of the TGM (see Chapter 4). These issues could be resolved by subsequent adjustments, as described in the sections below.

Similar Test Cases If multiple test cases are interdependent, e.g., because they cover the same vulnerability, the CVSS score of a vulnerability may be added multiple times. If the vulnerability were added to V_K more than once, the computed security score would be distorted and its integrity corrupted. E.g., the vulnerability tested by test case IoT-16 (Zigbee device hijacking) can only be exploited, if the vulnerability that is tested by IoT-15 (Zigbee network configuration reset) is present (see Table B.1) [35]. If a Zigbee device can be hijacked, it is implied that its Zigbee network configuration can be reset, as well. Both test cases are interdependent and the same vulnerability is rated twice.

This problem was not considered during TMS implementation. However, a solution was found subsequently: the utilized TMS tool TestLink allows the definition of relations between test cases. Test cases can be defined to block each other: if one test case has failed (i.e. the vulnerability is present), the other one is not to be run [23]. This functionality can be applied to test cases that assess the same vulnerability.

Device Specific Vulnerability Scoring The TMS, as it is implemented in Section 5.1, does not allow an device specific change of V_K related risk scores. The implementation assigns a static score for each test case. Therefore, device specific differences can not be considered automatically by the utilized CVSS scores. This problem did not cause a major issue for the two devices used for this evaluation, but could require a subsequent solution for security assessments of a larger scale. Such solution could be realized by defining TestLink Custom Fields that are to be filled out during test execution, allowing the insertion of custom CVSS scores in the Test Report. While this solution could resolve the discovered problem, it also requires additional work, since each CVSS score must be recomputed in each round of test execution.

6.2. SSS Evaluation

With this section, the SSS, introduced in Chapter 3, is evaluated. After a testbed of 11 IoT devices is described in Section 6.2.1, the SSS's behavior is analyzed with the help of risk scores of exemplary IoT devices in Section 6.2.2. Thereafter, the SSS is compared to the previously analyzed scoring systems CVSS and WRR in Section 6.2.3.

6.2.1. Testbed

The authors of [3] used a set of 13 IoT devices to evaluate the WRR model implementation. Because the single feature set risk values (RSS_C, RSC_O, and RSV_K) of these 13 devices, which are required for security score computation, were omitted in [3], a simple extension for the SSS is not possible. This circumstance makes the development of a separate list of IoT devices, for which the individual feature sets are known, necessary. Therefore, a new list of 11 IoT devices is prepared for SSS evaluation. The devices and their risk scores are depicted in Appendix C.2. The risk scoring process is performed similar to the process in [3]: for each device's V_K set, the NVD is queried for vulnerabilities. For S_C, "the specification of the device (based on the device type and model)" [3] is used, and for C_O, "the maximum context (the worst-case scenario) among all possible contexts for the IoT device under test" [3] is referred. For both feature sets, the risk scores that were previously obtained through the expert questionnaire [3, Table 1, Figure 2], are utilized.

Note that all vulnerabilities present in the NVD were utilized, regardless if there already exist security fixes for the latest product versions. The two Zigbee gateways Philips Hue Bridge v2 and Osram Lightify Home are an exception to this strategy, because their V_K sets are obtained from the security assessment of Section 6.1. Also, note the special case of Miele XGW 3000, an IoT gateway that is listed twice in Appendix C.2. The difference between the two entries lays in their V_K sets. Both sets hold the same vulnerabilities, however, the entry marked with *MITRE* is assigned CVSS scores by the corporation MITRE, while the other entry uses the default CVSS scores of the NVD. The decision to include both versions is made to highlight the consequences that are caused by different vulnerability ratings.

6.2.2. Security Score Analysis for Existing IoT Devices

After the feature sets for each device are identified (see Section 6.2.1), the devices' RSV_K, D_{BRS}, and S_S values are computed. Figure 6.2 depicts these values, where the X-axis represents D_{BRS} and the Y-axis represents RSV_K. Figure 6.1 illustrates the devices' RSV_K and S_S values.

Through Figure 6.1, it is observed that S_S values are located near the numerical boundaries of SSS (0 and 10). There are numerous reasons that could cause this. For one, such behavior can be explained with the non-linear nature of S_S. Further, the devices' RSV_K values are not uniformly distributed and located on the spectrum's edges, which also has considerable impact on S_S. Lastly, a non-optimal calibration of μ, ν, and ω can contribute to such supersaturation, as well. Because these constants were chosen without extended research and rather based on a small set of rules defined in Section 3.3, the calibration certainly has influence on the suboptimal S_S values' distribution.

As illustrated in Figure 6.2, only one device, namely Siemens Gigaset se361, has a RSV_K value that is within the range, where the value of D_{BRS} can influence the qualitative rating (Secure, Insecure) for $S_{SL} = 2.5$. While the gateway is deemed Secure, a sufficiently smaller D_{BRS} value will result in the rating Insecure. For every other device, the RSV_K value is outside the illustrated boundaries for $D_{BRS} < 140$. This means that for those devices, RSV_K must change first, before D_{BRS} has influence on the qualitative rating. This observation allows the conclusion that, for most of the analyzed devices and the utilized μ, ν, and ω values (see Section 3.3), the size of RSV_K, i.e. the devices' vulnerabilities, have higher impact on the qualitative security rating than their D_{BRS} values. By adjusting μ, ν, and ω, the S_{SL} plot illustrated in Figure 6.2 can be stretched on the Y-axis, attributing more influence to the D_{BRS} value. This observation demonstrates the SSS's options for calibration.

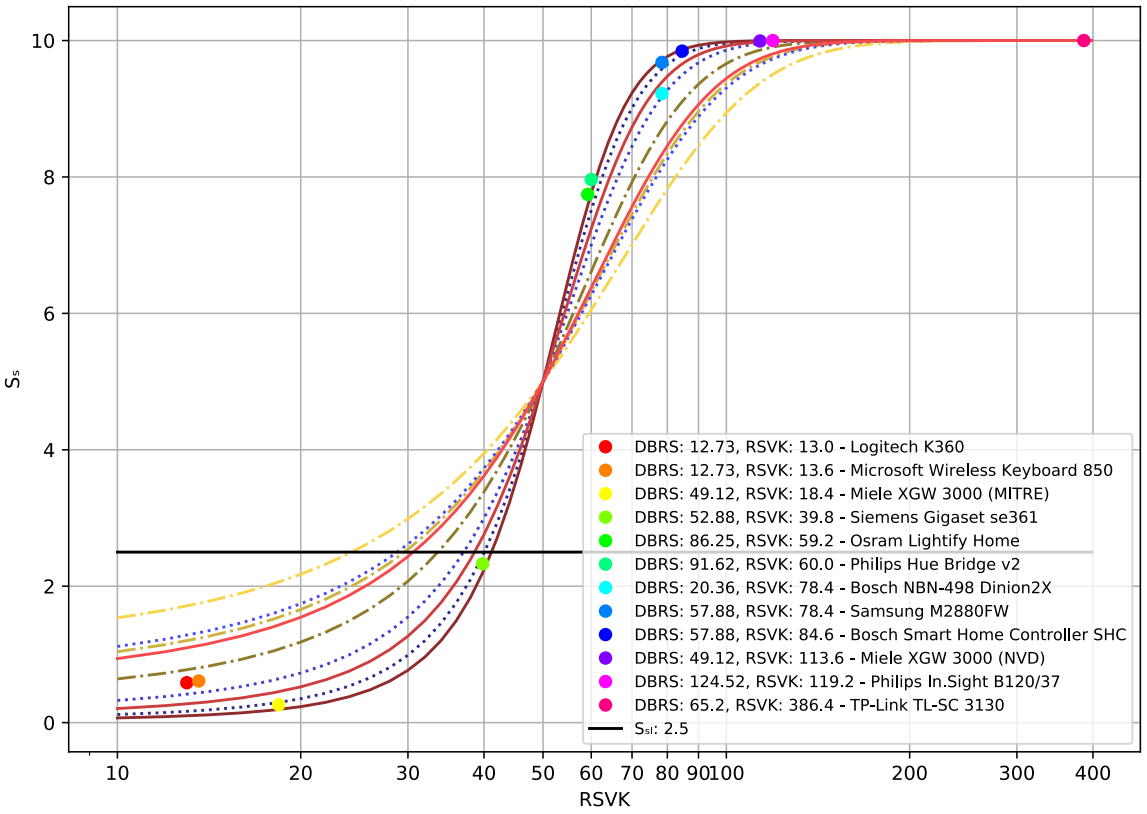

Figure 6.1.: IoT Devices mapped by their Vulnerability Risk Scores (RSV_K) and Security Scores (S_S).

6.2.3. Comparison to WRR and CVSS

The devices used for evaluation in [3] are scored and ranked with the Device Risk Score Calculation method D_{RS} (see Equation 2.5). The ranking is then compared to a ranking that is based on the sums of the vulnerabilities' CVSS scores. This SSS evaluation uses the same approach and adds a ranking that is based on the devices' SSS scores (see Table 6.4).

The evaluation of WRR observes that "the NVD metric presents much different results" [3] than the WRR's metric. This observation can not be made for the devices of the SSS evaluation. The authors of [3] explain the observation with "six of the IoT devices under test [that] have a risk score of 0.0". This is different to the devices used in this evaluation: the sum of CVSS scores ranges from 6.5 to 66.8. Hence, the RSV_K values have a bigger impact on the D_{RS} scores, resulting in higher correlation. However, there are larger differences between the S_S and D_{RS} rankings. These are analyzed in the sections below.

Miele XGW 3000 (MITRE) The Zigbee gateway Miele XGW 3000 (MITRE) is rated most secure by SSS ($S_S = 0.3$), closely followed by two wireless keyboards with $S_S = 0.6$. In the case of rating with both the WRR model, the gateway is rated more risky than both keyboards. The RSC_O values of all three devices are almost equal and the RSV_K values show only small differences. However, the gateway's RSS_C value highly exceeds the keyboards' RSS_C values. This explains both the reason, why the gateway is rated more risky by the WRR model, and, why the gateway is rated more secure by SSS: the WRR model computes the sum of all feature sets. Because the gateway has more service capabilities, it is rated more risky. On the other hand, the SSS allows the presence of more vulnerabilities for a constant RSC_O, if a device has more service capabilities (see

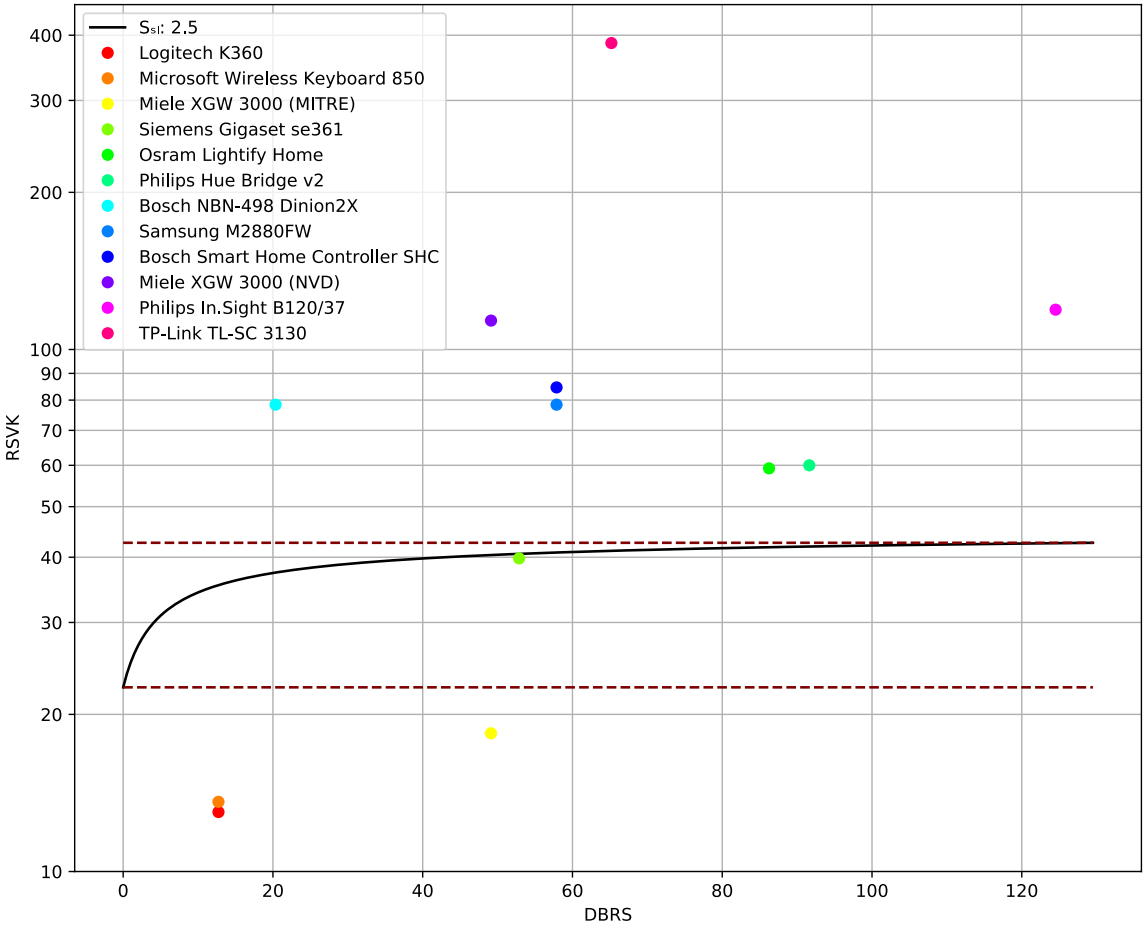

Figure 6.2.: IoT Devices mapped by their Vulnerability Risk Scores (RSV_K) and Device Base Risk Scores (D_{BRS}).

rules in Section 3.1.1). Hence, this difference in ranking is explained by the nature of SSS and can be argued as advantage to the WRR model.

Philips Hue Bridge v2 Similar to Miele XGW 3000 (MITRE), the Zigbee gateway Philips Hue Bridge v2 is ranked very differently because of its RSS_C value. With the highest RSS_C value of all devices and an indifferent RSV_K value (see Table C.2), the gateway is located in the first half of the SSS ranking. However, its high RSS_C value causes the D_{RS} value to be considerably higher, resulting in a higher WRR model ranking.

In a less extreme manner, this observation can also be applied to the device Osram Lightify Home that has slightly lower RSS_C and RSV_K values, but a similar proportion between the two values.

Bosch NBN-498 Dinion 2X Another discrepancy in rating is represented by the surveillance camera Bosch NBN-498 Dinion 2X. Its user manual lists potential use cases that include prison surveillance [45]. Because prisons represent a high risk operational context, but such context is not listed in the time-location matrix in [3], the combination with the highest possible risk value, which is Server Room and Night, is used for scoring. The WRR model does not assign a high impact to the C_O feature set and this surveillance camera does not have a huge set of service capabilities, compared to other cameras (see Table C.2). Therefore, this surveillance camera is ranked far less risky by the WRR model than the surveillance cameras Philips In.Sight B120/37 and TP-Link TL-SC 3130

Device	S_S	S_S Rank	D_{RS}	D_{RS} Rank	$\sum$ CVSS Scores	CVSS Rank
Miele XGW 3000 (MITRE)	0.3	1	42.1	3	9.2	3
Logitech K360	0.6	2	20.0	1	6.5	1
Microsoft Wireless Keyboard 850	0.6	3	20.3	2	6.8	2
Siemens Gigaset se361	2.3	4	52.4	5	12.1	6
Osram Lightify Home	7.7	5	78.0	8	14.8	7
Philips Hue Bridge v2	8.0	6	81.3	10	15.0	8
Bosch NBN-498 Dinion2X	9.2	7	50.9	4	9.8	4,5
Samsung XPress M2880FW	9.7	8	69.9	6	9.8	4,5
Bosch SHC	9.8	9	72.3	7	35.2	11
Miele XGW 3000	10	10	78.7	9	18.6	9
Philips In.Sight B120/37	10	11	96.9	11	22.7	10
TP-Link TL-SC 3130 IP Camera	10	12	195.1	12	66.8	12

Table 6.4.: Scoring System Comparison of SSS, WRR, and CVSS.

IP Camera (see Table 6.4), which both have less risky operational contexts and higher RSV_K and RSS_C values.

The SSS utilizes the C_O set to measure the importance of having as few vulnerabilities as possible. A high RSC_O value influences the S_S more than a high RSS_C value. In the case of this camera, the presence of one severe vulnerability in combination with an operational context that is connected to high risks, leads to a risk higher ranking by the SSS than by the WRR model (see Table 6.4), explaining the observed discrepancy.

Comparison Summary Overall, the rankings of the SSS and the WRR model are mostly congruent. The divergences can be explained with the individual models' behaviors and confirm their applicability for the destined tasks.

7. Future Work

There is large room for improvement and further analysis in the Internet of Things (IoT) security assessment field. For all three discussed scoring systems, Common Vulnerability Scoring System (CVSS) [10], Weighted Risk Ranking (WRR) [3], and Security Scoring System (SSS) (see Chapter 3), there are opportunities to improve both their accuracy and applicability for IoT devices. Furthermore, there are open questions regarding the evaluation of the Testing Guide Model (TGM) (see Chapter 4) that only Certification Authorities (CAs) will be able to answer. The open questions and possible improvements are discussed in greater detail in the sections below.

7.1. TGM Evaluation

The TGM evaluation conducted in Section 6.1 is highly limited since it was performed under hypothetical conditions without real-world application. Practical tests were not conducted, because those would require the discussion of undisclosed vulnerabilities. Likewise, the client communication aspect of TGM could not be evaluated sufficiently. Studies, carried out by CAs in collaboration with clients that are willing to participate, could improve the TGM evaluation: if these studies are published with adequate delay, clients are able to act on discovered vulnerabilities and sufficiently mitigate those before publication. With a participating client, the client communication aspects can be assessed simultaneously.

7.2. CVSS Extensions Framework

The CVSS Extensions Framework introduced with version 3.1 [15] and discussed in Section 2.2.3.3 is found to hold the opportunity of integrating IoT aspects in the CVSS metrics. Because of its recency, there are no known efforts for such implementation. Future research may utilize the Extension Framework to counter the problems of CVSS utilization for IoT vulnerabilities that are discussed in Sections 2.2.3.1 and 2.2.3.2. It is to be investigated, if future CVSS Extensions Framework implementations are also able to improve the accuracy of the systems WRR and SSS.

7.3. WRR Accuracy

The authors of [3] suggest that "in future research, additional device specific and domain related features and elements may be used to enhance the model's capabilities" to allow

more precise risk scoring. They also suggest such additional component: "an important factor that should be considered in the model is whether the device is secured by design, including whether it uses TLS/SSL for data in transit or uses an encryption mechanism for data at rest, and whether the device is physically secured" [3]. To improve accuracy, it is also suggested to add calibration mechanisms for risk weights and extend the expert questionnaire with additional questions [3]. Because those improvements will also affect the accuracy of the SSS, it is desirable that these suggested steps will be developed in the future.

7.4. SSS Constants

The constants μ, ν, and ω used in Equation 3.3 have major influence on the SSS security scores. The correct calibration of these values has high priority, since they should be kept unchanged during application. The values used in this bachelor thesis are determined through a small set of rules (see Section 3.3) and are not meant for practical application. Hence, mechanisms for calibrating these constants must be developed.

There are multiple potential approaches for calibrating the constants' values. One approach, that has already been used in [3], is to use an expert questionnaire and determine the values through practical device examples rated by experts. However, "one of the main disadvantages of using a domain expert questionnaire is that the model is static" [3]. For each future model change, experts need to be consulted, again. In [3], the utilization of machine learning and big data as alternative to the expert questionnaire is considered. Indeed, these approaches could also help in the case of the SSS: with enough input (IoT devices with known S_C, C_O, and V_K sets), the ideal values of μ, ν, and ω could be determined through basic mathematical functions, machine learning, or through optimization problem algorithms. This approach could, unlike the expert questionnaire approach, improve the SSS continuously, if regularly provided with new input.

As for the SSS model itself, only the values of μ, ν, and ω must be determined correctly. Before SSS can be used by CAs, however, the maximal allowed security score S_{SL} must be determined, as well. Just like μ, ν, and ω also S_{SL} should keep unchanged over time. Determining S_{SL} could also be determined through machine learning, big data, and/or expert knowledge: with enough quantitatively rated devices and an expert's indication about their qualitative rating (Secure, Insecure), the S_{SL} can be set to the optimal value.

8. Conclusion

The goals of this bachelor thesis are to standardize Internet of Things (IoT) security assessments of Certification Authorities (CAs) and develop an approach that allows the fair and equal testing of IoT devices by ensuring continuously equal testing conditions and rating mechanisms. This bachelor thesis has investigated the existing methods of IoT security assessments through the view of CAs and proposed the two novel solutions Testing Guide Model (TGM) and Security Scoring System (SSS) to solve the issues prevailing in the existing systems.

The TestLink based TGM implementation has demonstrated the applicability of the introduced TGM and Iterative Testing Procedure (ITP) models. During evaluation with two IoT devices, no substantial flaws were discovered. However, as depicted in Section 7.1, it is concluded that additional evaluation performed by CAs is desirable.

In order to improve CAs' proprietary decision making processes for product certification, the quantitative and qualitative scoring system SSS was developed. The SSS rates the security of IoT devices based on their vulnerabilities in relation to their service capabilities and operational contexts. This system is an extension to an existing risk ranking solution, the Weighted Risk Ranking (WRR) model. The need for the new scoring system SSS originates in the deficiencies of existing scoring systems like Common Vulnerability Scoring System (CVSS), as extensively discussed in Section 2.2. Both the WRR model and the SSS incorporate the CVSS as a subcomponent. Therefore, both models may profit from future improvements for the CVSS, as determined in Section 7.2.

With the help of a risk ranking comparison of multiple IoT devices, it was demonstrated that the SSS both fulfills the previously defined rules of behavior and is able to rank the devices' security more precisely than both WRR and CVSS. However, the main task of the SSS lays in precise rating, not ranking. The calibration of SSS variables that is needed for achieving more precise ratings was only initially performed with this bachelor thesis and should, as suggested in Section 7.4, be improved by future research.

This bachelor thesis has successfully demonstrated new approaches that have both the potential of improving the security assessment strategies of CAs and also shift the development of such methods from the proprietary domain to the public. It is desirable that CAs assess the findings of this bachelor thesis, enhance the developed models and incorporate them in their security assessments.

List of Figures

List of Tables

Listings

Bibliography

[1] "Overview of the Internet of things," tech. rep., International Telecommunication Union, June 2012.

[2] "Systems and software engineering – Vocabulary," ISO 24765, International Organization for Standardization, Dec. 2012.

[3] S. Siboni, C. Glezer, A. Shabtai, and Y. Elovici, "A Weighted Risk Score Model for IoT Devices," *Security, Privacy, and Anonymity in Computation, Communication, and Storage 2019 International Workshops*, pp. 20 – 34, July 2019.

[4] A. Kaushik, P. Havart-Simkin, K. Sharpington, M. Arnott, E. Goodness, A. Velosa, and P. Middleton, "Scenarios for the IoT Marketplace, 2019," *Gartner, Inc.*, July 2019.

[5] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai Botnet," in *26th USENIX Security Symposium (USENIX Security 17)*, (Vancouver, BC), pp. 1093–1110, USENIX Association, Aug. 2017.

[6] P. Middleton and R. Contu, "Forecast: Enterprise and Automotive IoT Edge Device Security, Worldwide, 2018-2024," *Gartner, Inc.*, Mar. 2020.

[7] Bundesministerium der Justiz und für Verbraucherschutz, "Gesetz über das Bundesamt für Sicherheit in der Informationstechnik," 2009.

[8] VDE Prüf- und Zertifizierungsinstitut GmbH, "Informationssicherheit / Cyber Security – Prüfung und Zertifizierung im VDE-Institut." `https://www.vde.com/tic-de/dienstleistungen/informationssicherheit`. Accessed: 2020-05-14.

[9] H. Joh and Y. K. Malaiya, "Defining and Assessing Quantitative Security Risk Measures Using Vulnerability Lifecycle and CVSS Metrics," *Colorado State University*, 2011.

[10] "Common Vulnerability Scoring System version 3.1 - Specification Document," *FIRST*, June 2019.

[11] D. Klinedinst, "CVSS and the Internet of Things," *Software Engineering Institute, Carnegie Mellon University*, Sept. 2015.

[12] "Common Vulnerability Scoring System version 2.0 - Specification Document," *FIRST*, June 2007.

[13] "Common Vulnerability Scoring System version 3.0 - User Guide," *FIRST*, June 2015.

[14] National Institute of Standards and Technology, "National vulnerability database." `https://nvd.nist.gov/`. Accessed: 2020-03-01.

[15] "Common Vulnerability Scoring System version 3.1 - User Guide," *FIRST*, June 2019.

[16] "Common Vulnerability Scoring System version 3.0 - Specification Document," *FIRST*, June 2015.

[17] M. S. Lund, B. Solhaug, and K. Stølen, *Model-Driven Risk Analysis*. Berlin Heidelberg: Springer Verlag, 2011.

[18] W. Abbass, Z. Bakraouy, A. Baina, and M. Bellafkih, "Assessing the Internet of Things Security Risks," *Journal of Communications*, Oct. 2019.

[19] "Security Risk Assessment Tool." `https://www.healthit.gov/topic/privacy-security-and-hipaa/security-risk-assessment-tool`. Accessed: 2020-05-12.

[20] P. Radanliev, D. C. De Roure, J. R. C. Nurse, R. M. Montalvo, S. Cannady, O. Santos, L. Maddox, P. Burnap, and C. Maple, "Future developments in standardisation of cyber risk in the Internet of Things (IoT)," *SN Applied Sciences*, 2019.

[21] S. Gerdes and O. Bergmann, "Security Requirements for Managing Smart Object in Home Automation," *Mobile Networks and Management, 4th International Conference*, pp. 231 – 243, Sept. 2013.

[22] F. D. Patterson and K. Neailey, "A Risk Register Database System to aid the management of project risk," *Elsevier Science Ltd*, 2002.

[23] Teamtest, "TestLink." `https://github.com/TestLinkOpenSourceTRMS/testlink-code/`. Accessed: 2020-05-12.

[24] Mitre, "Common attack and pattern enumeration and classification." `https://capec.mitre.org/`. Accessed: 2020-02-21.

[25] Mitre, "Common weakness enumeration." `https://cwe.mitre.org/`. Accessed: 2020-02-21.

[26] Mitre, "Common vulnerabilities and exposures." `https://cve.mitre.org/`. Accessed: 2020-02-21.

[27] "ISO 15408: Common Criteria for Information Technology Security Evaluation," Standard, International Organization for Standardization, Apr. 2017.

[28] "IEC 62443-4-2," Standard, International Electrotechnical Commission, 2015.

[29] "Testing Guide Model Implementation." `https://gitlab2.informatik.uni-wuerzburg.de/s354363/bachelor-thesis-code`. Accessed: 2020-05-12.

[30] "IoT Security Compliance Framework," Standard, IoT Security Foundation, Dec. 2018.

[31] R. A. Melgares, "802.15.4/ZigBee Analysis and Security: Tools for practical exploration of the attack surface," *Dartmouth College*, May 2011.

[32] X. Fan, F. Susan, W. Long, and L. Shangyan, "Security Analysis of Zigbee," *Massachusetts Institute of Technology*, May 2017.

[33] R. Meyer, "Security Issues and Vulnerability Assessment of ZigBee enabled Home Area Network Implementations," Master's thesis, California State University, Sacramento, 2012.

[34] T. Zillner, "ZigBee Exploited - The Good, The Bad and The Ugly," *Magdeburger Journal zur Sicherheitsforschung*, no. 12, pp. 699 – 704, 2016.

[35] P. Morgner, S. Mattejat, Z. Benenson, C. Müller, and F. Armknecht, "Insecure to the Touch: Attacking ZigBee 3.0 via Touchlink Commissioning," *10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 230 – 240, July 2017.

[36] B. Stelte and D. Rodosek, "Thwarting Attacks on ZigBee - Removal of the KillerBee Stinger," *International Conference on Network and Service Management*, 2013.

[37] O. Ben-Kiki, C. Evans, and I. d. Net, "YAML Specification." `https://yaml.org/spec/`. Accessed: 2020-05-12.

[38] C. O'Flynn, "Getting Root on Philips Hue Bridge 2.0." `https://colinoflynn.com/2016/07/getting-root-on-philips-hue-bridge-2-0/`. Accessed: 2020-04-26.

[39] "CVE-2020-6007." `https://cve.circl.lu/cve/CVE-2020-6007`, Dec. 2020.

[40] "R7-2016-10: Multiple OSRAM SYLVANIA Osram Lightify Vulnerabilities (CVE-2016-5051 through 5059)." `https://blog.rapid7.com/2016/07/26/r7-2016-10-multiple-osram-sylvania-osram-lightify-vulnerabilities-cve-2016-5051-through-5059/`, June 2016.

[41] Philips, "Philips Hue instore app - Installation manual." `https://images.philips.com/is/content/PhilipsConsumer/PDFDownloads/United%20States/MeetHue/ODLI20170630_001-UPD-en_US-HueInstore-App_InstallationManual_v1-0-0.pdf`. Accessed: 2020-05-08.

[42] Osram, "LIGHTIFY App - User-Guide." `https://lightfiy-customer-service.s3.eu-central-1.amazonaws.com/Manuals/LIGHTIFY_User_Guide_EN_Version2.0.pdf`. Accessed: 2020-05-08.

[43] "Base Device Behavior Specification," *ZigBee Alliance*, Feb. 2016.

[44] J. Wright, "KillerBee: Practical ZigBee Exploitation Framework," *InGuardians*, Oct. 2009.

[45] "NBN-498 Dinion2X Day/Night IP Cameras," Manual, Bosch Security Systems, Nov. 2015.

Appendix

A. SSS Implementation

Listing 8.1: Python Implementation of the WRR Model.

```
class WRR():
    alpha = 0.384
    beta  = 0.341
    gamma = 0.275

    weights = [0, 2**0, 2**1, 2**2, 2**3]
    score_to_weights = [0.0, 3.9, 6.9, 8.9, 10.0]
    vk:list = None
    sc:list = None
    co:list = None

    def __init__(self, vk:list, sc:list, co:list):
        self.vk = vk
        self.sc = sc
        self.co = co

    def wrrm(self, score):
        if score < 0.0 or score > 10.0:
            return -1
        for i, s in enumerate(self.score_to_weights):
            if score <= s:
                return self.weights[i] * score

    def rsvk(self):
        return sum([self.wrrm(vk_) for vk_ in self.vk])

    def rssc(self):
        return sum([self.wrrm(sc_) for sc_ in self.sc])

    def rsco(self):
        return sum([self.wrrm(co_) for co_ in self.co])
```

```
32
33      def rsco_max(self):
34          return sum([self.wrrm(10.0) for co_ in self.co])
35
36      def risk_score(self):
37          return self.alpha * self.rsvk() \
38              + self.beta * self.rssc() \
39              + self.gamma * self.rsco()
```

Listing 8.2: Python Implementation of the SSS Model.

```
1  import math
2
3  class SSS():
4      mu  = 0.3
5      nu = 50.0
6      omega = 0.04
7      limit = 10
8
9      def base_risk_raw(self, rssc, rsco, rsco_max):
10         if rsco_max <= 0:
11             rsco_max = 1
12         return rssc * ( 1 - (rsco / rsco_max) )
13
14     def base_risk(self, wrr:WRR):
15         return self.base_risk_raw(wrr.rssc(), wrr.rsco(),
16                                   wrr.rsco_max())
17
18     def score_raw(self, rsvk, base_risk_):
19         result = self.limit / ( \
20                 (math.e + self.mu * base_risk_) ** \
21                 (-self.omega*(rsvk - self.nu)) + 1 )
22         return result
23
24     def score(self, wrr:WRR):
25         return self.score_raw(wrr.rsvk(), self.base_risk(wrr))
26
27     def allowed_rsvk(self, limit_, base_risk_):
28         if limit_ > self.limit or base_risk_ < 0:
29             return - 1
30         result = self.nu - (1 / self.omega) * \
31                 math.log((self.limit - limit_) / limit_,
32                     math.e + self.mu * base_risk_)
33         return max(result, -1)
```

B. TGM Test Cases

Test Case Id.	Name	Description	CVSS
IoT-6	ZB Default Global Link Key	Tests, if the Default Global Link Key is used and if such utilization is not advised.	7.4
IoT-7	ZB Fallback Global Link Key	Tests, if the Default Global Link Key is allowed as fallback.	7.4
IoT-8	ZB Frame Counter Unchecked	Tests, if the Frame Counter mechanism is implemented correctly.	9.0
IoT-9	ZB OTA Link Key Transmission	Tests, if Link Keys are transmitted unencrypted.	8.3
IoT-10	ZB DoS Association Flooding	Tests, if the DUT is prone to simple DoS attacks.	7.2
IoT-11	ZB Security Level	Verifies, if the required Zigbee Security Level is used.	8.8
IoT-12	ZB Touchlink	Tests, if the allowed proximity for Touchlink Commissioning is satisfied.	8.5
IoT-13	ZB Firmware Update	Tests, if firmware updates are properly encrypted.	7.3
IoT-14	ZB Identify Action	Tests, if the DUT is prone to a DoS attack situated in the Zigbee Identify Action.	6.5
IoT-15	ZB Factory Reset	Tests, if the DUT's Zigbee network configuration can be reset with inter-PAN command frames, removing it from the existing network.	7.4
IoT-16	ZB Hijacking	Tests, if the DUT can be removed from the existing network (see IoT-15) and automatically rejoined to an arbitrary one.	9.4
IoT-17	ZB Firmware Update Impossible	Assesses, under which circumstances firmware updates over Zigbee networks become impossible.	8.3
IoT-18	CVE-2020-6007	Tests, if a buffer overflow bug is present in the Zigbee command processing mechanism of the Philips Hue Bridge v2.	4.3
IoT-20	CVE-2016-5053	Tests, if forged commands are accepted by the Osram Lightify Home gateway without requiring authentication.	5.0
IoT-23	Root Access	Tests, if root privileges for the Philips Hue Bridge v2 can be gained through a hardware attack.	7.6
IoT-137	IoTSF-2.4.10.1	Verifies that, where the product or service provides a web based user interface, strong authentication is used.	5.0

Table B.1.: Test Cases used for Evaluation.

C. Risks of IoT Devices

Device Type	Device	Feature Set	Risk Description	Risk Score	D_{BRS}	S_S
Zigbee Gateway	Philips Hue Bridge v2	S_C	WiFi	9.5	91.6	8.0
			Zigbee	8.5		
			Ethernet	4.3		
			App	7.0		
		C_O	IT Department, Night	7.5		
		V_K	IoT-15: Zigbee Factory Reset	7.4		
			IoT-23: Root Access	7.6		
Zigbee Gateway	Osram Lightify Home	S_C	WiFi	9.5	86.3	7.7
			Zigbee	8.5		
			App	7.0		
		C_O	IT-Department, Night	7.5		
		V_K	IoT-15: Zigbee Factory Reset	7.4		
			IoT-6: Zigbee Default Global Link Key	7.4		
Zigbee Gateway	Miele XGW 3000	S_C	USB	4.0	49.1	10
			Zigbee	8.5		
			Ethernet	4.3		
			App	7.0		
		C_O	IT Department, Night	7.5		
		V_K	CVE-2019-20481	9.8		
			CVE-2019-20480	8.8		
Zigbee Gateway	Miele XGW 3000 (MITRE)	S_C	USB	4.0	49.1	0.3
			Zigbee	8.5		
			Ethernet	4.3		
			App	7.0		
		C_O	IT Department, Night	7.5		
		V_K	CVE-2019-20481 (MITRE rating)	4.6		
			CVE-2019-20480 (MITRE rating)	4.6		
Gateway	Siemens Gigaset se361	S_C	WiFi	9.5	52.9	2.3
			Ethernet	4.3		
		C_O	IT Department, Night	7.5		
		V_K	CVE-2009-3322	7.8		
			CVE-2007-4488	4.3		
Gateway	Bosch Smart Home Controller (SHC)	S_C	WiFi	9.5	57.875	9.8
			Ethernet	4.3		
			USB	4.0		
		C_O	IT-Department, Night	7.5		
		V_K	CVE-2019-11891	5.4		
			CVE-2019-11892	6.8		
			CVE-2019-11893	4.9		
			CVE-2019-11894	5.7		
			CVE-2019-11895	5.3		
			CVE-2019-11896	7.1		
Surveillance Camera	Bosch NBN-498 Dinion2X	S_C	Camera	5.8	20.4	9.2
			Mic	5.6		
			Motion Detector	3.1		
			Ethernet	4.3		
		C_O	Server Room, Night	8.2		
		V_K	CVE-2015-6970	9.8		

Surveillance Camera	Philips In.Sight B120/37	S_C	WiFi Camera Mic Motion Detector Thermometer Loudspeaker USB App	9.5 5.8 5.6 3.1 3.0 1.0 4.0 7.0	124.5	10
		C_O	Internal, Night	4.9		
		V_K	CVE-2015-2884 CVE-2015-2883 CVE-2015-2882	7.5 5.4 8.9		
Surveillance Camera	TP-Link TL-SC 3130 IP Camera	S_C	WiFi Camera Mic Motion Detector Ethernet	9.5 5.8 5.6 3.1 4.3	65.2	10
		C_O	Server Room, Night	8.2		
		V_K	CVE-2013-2573 CVE-2013-2572 CVE-2018-18428 CVE-2013-2581 CVE-2013-2580 CVE-2013-2579 CVE-2013-2578 CVE-2013-3688	9.8 7.5 7.5 7.8 7.1 10.0 10.0 7.1		
Printer	Samsung XPress M2880FW	S_C	WiFi USB Ethernet	9.5 4.0 4.3	57.9	9.7
		C_O	IT-Department, Night	7.5		
		V_K	CVE-2015-5729	9.8		
Keyboard	Microsoft Wireless Keyboard 850	S_C	USB Proprietary Wireless Communication	4.0 6.1	12.7	0.6
		C_O	IT-Department, Morning	7.4		
		V_K	CVE-2018-8117	6.8		
Keyboard	Logitech K360	S_C	USB Proprietary Wireless Communication	4.0 6.1	12.7	0.6
		C_O	IT-Department, Morning	7.4		
		V_K	CVE-2019-13055	6.5		

Table C.2.: IoT Devices with Associated Risk Ratings.

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Würzburg, 26th May 2020

.....

(Moritz Anton Finke)