University of Passau
Faculty of Computer Science and Mathematics

Chair of Computer Engineering
Prof. Dr. Stefan Katzenbeisser

Master's Thesis
in
Computer Science

The IoT Security Pattern Landscape: Collection and Analysis of state-of-the-art Security Patterns for IoT Development

Eva Gründinger
85500

Date: 2023-03-22
Supervisors: Prof. Dr. Stefan Katzenbeisser
Prof. Dr. Joachim Posegga
Advisor: Felix Klement

Gründinger, Eva
Bärnbach 6
94116 Hutthurm

ERKLÄRUNG

Ich erkläre, dass ich die vorliegende Arbeit mit dem Titel „The IoT Security Pattern Landscape: Collection and Analysis of state-of-the-art Security Patterns for IoT Development“ selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind.

Mit der aktuell geltenden Fassung der Satzung der Universität Passau zur Sicherung guter wissenschaftlicher Praxis und für den Umgang mit wissenschaftlichem Fehlverhalten vom 31. Juli 2008 (vABlUP Seite 283) bin ich vertraut.

Ich erkläre mich einverstanden mit einer Überprüfung der Arbeit unter Zuhilfenahme von Dienstleistungen Dritter (z.B. Anti-Plagiatssoftware) zur Gewährleistung der einwandfreien Kennzeichnung übernommener Ausführungen ohne Verletzung geistigen Eigentums an einem von anderen geschaffenen urheberrechtlich geschützten Werk oder von anderen stammenden wesentlichen wissenschaftlichen Erkenntnissen, Hypothesen, Lehren oder Forschungsansätzen.

....
(Name, Vorname)

Translation of German text (notice: Only the German text is legally binding)

I hereby confirm that I have composed the present scientific work entitled “The IoT Security Pattern Landscape: Collection and Analysis of state-of-the-art Security Patterns for IoT Development” independently without anybody else's assistance and utilising no sources or resources other than those specified. I certify that any content adopted literally or in substance has been properly identified and attributed.

I have familiarised myself with the University of Passau's most recent Guidelines for Good Scientific Practice and Scientific Misconduct Ramifications of 31 July 2008 (vABlUP Page 283).

I declare my consent to the use of third-party services (e.g. anti-plagiarism software) for the examination of my work to verify the absence of impermissible representation of adopted content without adequate attribution, which would violate the intellectual property rights of others by claiming ownership of somebody else's work, scientific findings, hypotheses, teachings or research approaches.

Supervisor contacts:

Prof. Dr. Stefan Katzenbeisser
Chair of Computer Engineering
University of Passau
Email: stefan.katzenbeisser@uni-passau.de
Web: https://www.fim.uni-passau.de/en/computer-engineering

Prof. Dr. Joachim Posegga
Chair of IT Security
University of Passau
Email: Joachim.Posegga@uni-passau.de
Web: https://www.sec.uni-passau.de/en

Abstract

The IoT Security Pattern Landscape: Collection and Analysis of state-of-the-art Security Patterns for IoT Development

Designing a computer system is a complex project that gets easier by utilizing the right tools. Specific patterns and architectures allow a developer to simplify problems and analyse the structure of a system. By not only applying different patterns for design but also security, the developed system can ensure a security-aware operation. But the development and usage of said security patterns is hindered by the needed expert knowledge in these areas. For this reason extensive documentation and testing in practice is needed to ensure usable patterns that can be easily applied. While design patterns for development are common for most computer systems these days, domain-specific patterns like for Internet of Things (IoT) are rare to find. Because of the different requirements that an IoT system has, existing design patterns are often not suitable for this specific use case. When IoT-specific patterns are hard to find, it is only reasonable to assume that IoT patterns that focus on security aspects are even rarer. Thus, a structured and organized catalogue of IoT security patterns is basically non-existent at this point in time. Therefore, this master's thesis proposes a systematic collection and categorization of IoT security patterns and analyses the gaps of the recent research work regarding IoT security. As a catalogue that combines 61 IoT security patterns in one place and is organized in a top to bottom approach that follows the IoT World Forum Reference Model of the IoT architecture, this collection will play an important part in future development of secure IoT solutions. After gathering the IoT security publications and analysing the created pattern catalogue, our research questions (RQ) lead to the following conclusions. RQ1 indicates that the Data Accumulation layer shows a lack of developed pattern solutions in the IoT security field. When analysing the coverage of security goals in RQ2, the availability objective falls behind the others and is only addressed by around 30% of the total pattern collection. In RQ3 we asked ourselves which common vulnerabilities could be (partially) solved by our patterns and found that each point of the OWASP top ten is at least addressed by one security pattern. Number four among the issues on the list - the lack of ability to securely update the IoT device - was the hardest to find a solution for and is only mentioned once. To sum up, there is still a lot of room for further research in the development of IoT security patterns and expansion of the existing state-of-the-art. Therefore, we suggest a close cooperation between industry and academia to create usable patterns that help to secure our existing IoT infrastructures in the future.

Contents

1 Introduction **1**
1.1 Definition of Internet of Things . . . 1
1.2 Application Areas . . . 2

2 Background **3**
2.1 IoT Security Risks . . . 3
2.2 Top Ten Common IoT Vulnerabilities . . . 4
2.3 Secure IoT Design . . . 5
2.3.1 Design Patterns . . . 5
2.3.2 Security Patterns . . . 5
2.3.3 Security Architectures . . . 6

3 Related Work **7**
3.1 General Patterns for IoT Design . . . 7
3.2 Security-specific IoT Patterns . . . 8

4 Methodology **9**
4.1 Systematic Literature Review . . . 9
4.1.1 Research Questions . . . 9
4.1.2 Selection Criteria . . . 9
4.1.3 Search and Selection Strategy . . . 10
4.2 Taxonomy of the Research Area . . . 11
4.2.1 Pattern Description with POSA . . . 11
4.2.2 IoT World Forum Reference Model . . . 12
4.2.3 Security Concerns . . . 13

5 IoT Security Pattern Catalogue **15**
5.1 Physical Devices & Controllers . . . 19
5.1.1 Hardware IoT Security [SID+18] . . . 19
5.1.2 Secure IoT Thing [BFAO21] . . . 19
5.1.3 Secure Sensor Node [OFA20] . . . 20
5.1.4 Security Segmentation [FFYW20] . . . 20
5.1.5 Trusted Platform Module [MnF20] . . . 21
5.2 Connectivity . . . 21
5.2.1 Authenticated Channel [PFS+21] . . . 21
5.2.2 Encrypted Channel [PFS+21] . . . 22

5.2.3 Middleware Message Broker [TWG+18] . . . 22
5.2.4 Middleware Self-contained Message [TWG+18] . . . 22
5.2.5 Orchestration of SDN Network Elements [TWG+18] . . . 23
5.2.6 Outbound-Only Connection [RBF+17] . . . 23
5.2.7 Password Based Key Exchange [SOP+16] . . . 23
5.2.8 Safe Channel [PFS+21] . . . 24
5.2.9 Secure Remote Readout [URZ15] . . . 24
5.2.10 Signed Message [PFS+21] . . . 24
5.2.11 Symmetric Key Cryptography [SOP+16] . . . 24
5.2.12 Third Party Based Authentication [SOP+16] . . . 25
5.2.13 Trusted Communication Partner [RBF+17] . . . 25
5.2.14 Web Of Trust [SOP+16] . . . 26
5.3 Edge Computing . . . 26
5.3.1 Fog Computing [SFI16] . . . 26
5.3.2 Fogxy [STB18] . . . 27
5.3.3 Secure Cloud-based IoT Architecture [Fer20] . . . 28
5.4 Data Accumulation . . . 28
5.4.1 Encrypted Storage [KED+06] . . . 28
5.4.2 Redundant Storage [PFS+21] . . . 29
5.4.3 Safe Storage [PFS+21] . . . 29
5.5 Data Abstraction . . . 29
5.5.1 Alignment-based Translation [TWG+18] . . . 29
5.5.2 BlockBD [MFFMS19] . . . 29
5.5.3 Discovery of IoT Services [TWG+18] . . . 30
5.5.4 Flow-based Service Composition [TWG+18] . . . 30
5.5.5 IoT Gateway Event Subscription [TWG+18] . . . 30
5.5.6 IoT SSL CROSS-Layer Secure Access [TWG+18] . . . 31
5.5.7 Middleware Message Translator [TWG+18] . . . 31
5.5.8 Middleware Simple Component [TWG+18] . . . 31
5.5.9 REST Request/Response [TWG+18] . . . 31
5.5.10 Server Sandbox [KED+06] . . . 32
5.5.11 Service Orchestration [TWG+18] . . . 32
5.5.12 Translation with Central Ontology [TWG+18] . . . 32
5.6 Application . . . 33
5.6.1 Access Control to Physical Structures [FBDDLP07] . . . 33
5.6.2 Alarm Monitoring [FBDDLP07] . . . 33
5.6.3 Audit Log [LL17, PFS+21] . . . 34
5.6.4 Authenticated Session [PFS+21] . . . 34
5.6.5 Authorization Enforcer [PFS+21] . . . 34
5.6.6 Encrypted Processing [PFS+21] . . . 34
5.6.7 Fault Management [PFS+21] . . . 35
5.6.8 File Authentication [AA18] . . . 35
5.6.9 Matrix Authentication [AA18] . . . 35
5.6.10 Minefield [KED+06] . . . 35
5.6.11 Remote Authenticator/Authorizer [AA18] . . . 36
5.6.12 Role Based Access Control [AA18] . . . 36
5.6.13 Safe Processing [PFS+21] . . . 36
5.6.14 Secure Distributed Publish/Subscribe [FYW20] . . . 36
5.6.15 Uptime [PFS+21] . . . 37
5.7 Collaboration & Processes . . . 37
5.7.1 Account Lockout [PFS+21] . . . 37
5.7.2 Authentication Enforcer [PFS+21] . . . 38
5.7.3 Blacklist [RBF+17, PFS+21] . . . 38

5.7.4 History-Based Authentication [CDE+14] 38
5.7.5 Permission Control [RBF+17] 39
5.7.6 Personal Zone Hub [RBF+17] 39
5.7.7 Relays [FBDDLP07] 40
5.7.8 Single Access Point [PFS+21] 40
5.7.9 Whitelist [RBF+17] 40

6 Evaluation **41**
6.1 Data Set of IoT Security Primary Studies 41
6.2 RQ1: IoT World Forum Reference Model Categorization 42
6.3 RQ2: Protected Security Goals 43
6.4 RQ3: Solutions for Common Vulnerabilities 44
6.5 Use Case Application 46

7 Discussion **51**
7.1 Threats to Validity 51
7.2 Future Work 52
7.3 Conclusion 52

List of Figures **54**

List of Tables **55**

Bibliography **56**

1 Introduction

Today most smart devices are highly connected in a network that allows them to interact with each other. But this level of connectivity makes it difficult even for experts to keep the overview. These intelligent networks can be quite confusing when one does not have knowledge of the right tools/frameworks or when the needed technology does not even exist, yet. Many users also don't realize how dependent one can be on these devices and what security risks a connected world can bring. The first Section 1.1 introduces the term Internet of Things (IoT) and follows up in Section 1.2 with different areas of life that these technologies are likely to be found in to get familiar with IoT for the following analysis.

1.1 Definition of Internet of Things

The term IoT itself is common knowledge and these *"things"* are well-established in our society but only a few people know what exactly the *"Internet of Things"* is. In this section we define which devices belong to IoT and in which areas of life IoT enhances the daily routines of people.

> *"The Internet of Things (IoT) describes the network of physical objects — things — that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet.*[1]*"*

Although this quote from the oracle website gives a good general definition of what IoT encompasses, it is easier to understand the idea of IoT with an example: Picture a smart home in which every gadget and appliance inside the house is connected to the internet and communicates with each other. Using your smartphone one can lock the door and get notifications if an unauthorized person enters the backyard when he is detected by the installed motion sensors. The room temperature is regulated automatically with temperature sensors that are located outside and lights are controlled according to the time of day to save energy. More common examples and different types of IoT devices are:[2]

- Home security: Motion detection, automatic recording, remote arm or disarm
- Industrial IoT: Security or safety systems, machine sensors, project management
- Healthcare and fitness: Wearable healthcare devices, activity trackers

[1] https://www.oracle.com/in/internet-of-things/what-is-iot, last accessed: 12.10.2022.
[2] https://learn.g2.com/iot-devices, last accessed: 21.10.2022.

- Banking and financial services: ATMs, interactive payment cards
- Virtual/Augmented reality: Mobile or head-mounted AR, standalone VR

But the high connectivity and continuous data exchange between devices and applications has both benefits as well as disadvantages.

+ Time savings. By automating activities, it saves the user a lot of time.
+ Enhanced data collection. Information is easily accessible and frequently updated in real time.
+ Safety concerns. Devices sense potential danger and warn users in time.

– Security issues. Hackers may gain access to the system and steal personal information.
– High internet dependence. Devices are unable to function effectively without the internet.
– Complexity. There are many risks of failure in IoT devices.

1.2 Application Areas

After we discussed what IoT is in general and which devices belong to the IoT spectrum, we look into the application areas of this technology. Therefore we divide the field of IoT into five groups[3] and explore the different ways IoT can be utilized.

Consumer Internet of Things (CIoT). All kinds of appliances, gadgets and other connected devices that are meant for user consumption are a part of CIoT. Typical products include smartphones, wearables, home appliances, etc. These are in most cases connected via Wi-Fi or Bluetooth for short-range communication inside the home.

Commercial Internet of Things. Most Commercial IoT focuses on large venues in order to improve customer experience and business conditions. As an example these IoT devices can monitor environmental conditions, manage access control systems or economize utilities in office buildings, stores, healthcare institutions, entertainment venues or hotels.

Industrial Internet of Things (IIoT). In order to improve productivity and efficiency, most large-scale factories or manufacturing plants utilize IoT systems. IIoT can also be often found in the fields of healthcare, automotive, logistics, and agriculture.

Infrastructure Internet of Things. For the development of smart infrastructures this type of IoT tries to design IoT systems that are able to increase efficiency while decreasing its costs. These IoT infrastructures are then used in monitoring and controlling operations that deal with public infrastructures like bridges, railway tracks or even windfarms.

Internet of Military Things (IoMT). Like its name suggests, IoMT or also called Internet of Battlefield Things (IoBT) is an important part of the modern military. By interconnecting ships, planes, tanks, soldiers and even drones on the battlefield, not only the situational awareness during war, but also risk assessments and response times can be greatly improved. The data that is collected during monitoring can then be utilized to enhance the IoT systems, equipment, military practices and strategies.

[3] https://syntegra.net/internet-of-things-the-five-types-of-iot, last accessed: 12.10.2022.

2 Background

In order to simplify the following argumentation in this master's thesis and amplify the importance of secure IoT design, Section 2.1 describes common security risks that users of IoT devices should be aware of. Then we introduce the top ten IoT vulnerabilities according to OWASP (see Section 2.2) and lastly conclude this chapter with different design methods for secure IoT development in Section 2.3.

2.1 IoT Security Risks

When looking back at the smart home example of Section 1.1, we realize how many connections between different IoT devices and appliances are needed as basis for a modern smart system. But each of these communication paths can be seen as a potential risk for a cyber attack and these interfaces are not the only vulnerabilities of an IoT device. Hackers constantly develop new techniques to gain access to and control IoT systems with millions of attacks occurring every day. To give a few examples of what typical cyber attacks on IoT devices are, here are ten types that attackers commonly use:[1]

- Physical Attacks. After gaining access a USB drive is used to spread malicious code.
- Encryption Attacks. Capture of unencrypted data and control of system after decrypting the encryption keys.
- Denial of Service. One target is attacked by a larger number of systems which leads to services being unavailable.
- Firmware Hijacking. When firmware is not updated regularly, attackers can hijack it and download malicious software.
- Botnets. Usage of smart devices to transfer private corporate data or disable devices.
- Man-in-the-Middle. Intercepting the communication between two parties.
- Ransomware. By locking down files, this malware is used to blackmail corporations to pay money in order to get access to their data.
- Eavesdropping. Network traffic on a weak connection can be intercepted by an attack in order to gain sensitive data.

[1] https://micro.ai/blog/10-types-of-cyber-security-attacks-in-the-iot, last accessed: 06.12.2022.

- Privilege Escalation. Hackers use user profiles with higher privileges to deploy malware and steal confidential data that otherwise would be impossible.
- Brute Force Password Attack. Also known as dictionary attack, the attacker tries a huge set of random phrases to access a password-protected service.

2.2 Top Ten Common IoT Vulnerabilities

What are the biggest problems when building and managing an IoT system? This question is analysed and periodically updated by the Open Web Application Security Project or in short: OWASP[2]. The goal of this particular project is to help developers as well as users of IoT to better understand its security risks and give suggestions on how to make better decisions when working with this technology. Table 2.1 shows the top ten common IoT vulnerabilities from the year 2018 when OWASP last updated their results.[3]

Table 2.1: Internet of Things Top Ten (2018) taken from OWASP Wiki[3]

No.	Security Issue	Description
T1	Weak, Guessable, or Hard-coded Password	Use of easily bruteforced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorized access to deployed systems.
T2	Insecure Network Services	Unneeded or insecure network services running on the device itself, especially those exposed to the internet, that compromise the confidentiality, integrity/authenticity, or availability of information or allow unauthorized remote control.
T3	Insecure Ecosystem Interfaces	Insecure web, backend API, cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. Common issues include a lack of authentication/authorization, lacking or weak encryption, and a lack of input and output filtering.
T4	Lack of Secure Update Mechanism	Lack of ability to securely update the device. This includes lack of firmware validation on device, lack of secure delivery (un-encrypted in transit), lack of anti-rollback mechanisms, and lack of notifications of security changes due to updates.
T5	Use of Insecure or Outdated Components	Use of deprecated or insecure software components/libraries that could allow the device to be compromised. This includes insecure customization of operating system platforms, and the use of third-party software or hardware components from a compromised supply chain.
T6	Insufficient Privacy Protection	User's personal information stored on the device or in the ecosystem that is used insecurely, improperly, or without permission.
T7	Insecure Data Transfer and Storage	Lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.

[2]https://owasp.org, last accessed: 13.10.2022.
[3]https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project, last accessed: 09.09.2022.

Table 2.1 continued from previous page

T8	Lack of Device Management	Lack of security support on devices deployed in production, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities.
T9	Insecure Default Settings	Devices or systems shipped with insecure default settings or lack the ability to make the system more secure by restricting operators from modifying configurations.
T10	Lack of Physical Hardening	Lack of physical hardening measures, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device.

2.3 Secure IoT Design

This section focuses on different methods software developers use to simplify programming practices for object-oriented systems. We will discuss relevant patterns and architectures that support the design of secure IoT systems, e.g. design patterns, security-specific patterns, and architectures.

2.3.1 Design Patterns

The concept of design patterns in software development can be traced back to 1994 and the "Design Patterns - Elements of Reusable Object-Oriented Software" book [GHJV94]. The authors stated that a design pattern should be based on the following two principles:

- Program to an interface not an implementation.
- Favour object composition over inheritance.

In general, design patterns are specific to a scenario and provide a standard terminology that help professionals in creating best solutions for certain common problems in their field. According to the referenced book, design patterns can be grouped into four different categories:

- Creational Patterns. Provide a way to create objects while hiding the creation logic.
- Structural Patterns. Inheritance is used to create interfaces and define ways to compose objects to obtain new functionalities.
- Behavioural Patterns. Manage communication between objects.
- J2EE Patterns. Concerned with the presentation tier (identified by Sun Java Center).

Related categories that should be mentioned in this context would be anti-patterns[4] and misuse patterns [FYW09]. These patterns focus on common development mistakes and attacks respectively but because of their specific usage they are not further relevant for the topic of this master's thesis.

2.3.2 Security Patterns

If design patterns are re-usable solutions for common design problems, similar to that security patterns are intended to achieve specific security goals in a system. The security-by-

[4] https://www.bmc.com/blogs/anti-patterns-vs-patterns, last accessed: 17.10.2022.

design principle makes sure that the security aspect is in the forefront of the design process, integrated into the system and not added as an after-thought. The most important security goals are: *Confidentiality, Integrity, Availability, Authentication, and Authorization.* A subcategory of security patterns can be found in the so-called privacy patterns[5] that are specialized to guide developers in finding the best solution for common privacy problems in software development. They support developers in modelling privacy-by-design and creating practical software solutions in a standardized way.

2.3.3 Security Architectures

Cybersecurity threats are getting increasingly more common these days. As a way to protect ones system from cyberattacks and data breaches, security-specific architectures[6] can be used. These architectures enable developers to implement their security goals and keep their digital assets safe by ultimately utilizing a set of security principles, methods and models. Every architect needs guidelines to work with. In the context of security architectures these rules are called *frameworks*. There exist many international framework standards with The Open Group Architecture Framework (TOGAF), Sherwood Applied Business Security Architecture (SABSA) and Open Security Architecture (OSA) being common examples. Software breaches can mean very expensive losses for a company, not only in financial aspects but also the reputation of a company suffers. There are many good reasons to utilize a strong security architecture like:[6]

- Fewer security breaches. A strong security architecture is able to close common weaknesses and reduces the risk of successful system attacks.
- Money savings. Proactive security measures are able to detect and fix vulnerabilities earlier in the product development cycle. The later an error is detected, the more money it can cost.
- Disciplinary measures mitigation. In the event of an attack it matters to which degree the business tried to reduce its risk and prevent vulnerabilities beforehand in order to get a favourable outcome according to the regulations.

After establishing a definition for IoT, giving examples for common vulnerabilities in IoT systems and describing popular design methods in the context of IoT development, the basis for this master's thesis has been created. The following analysis will showcase the landscape of today's research in the field of IoT patterns and the lack of existing security-specific solutions. By collecting and sorting all gathered patterns into an exhaustive catalogue, a well-documented and hierarchically organized list was created that will guide future projects in the development of more security-focused systems.

[5] https://privacypatterns.org, last accessed: 17.10.2022.
[6] https://www.dig8ital.com/post/what-is-security-architecture-and-what-do-you-need-to-know, last accessed: 19.10.2022.

3 Related Work

There are already many papers and articles that describe the relevance of patterns and architectures that focus specifically on IoT. But besides common design patterns and frameworks, security-specific patterns for IoT applications are still in their early stages of development and documentation and therefore rare to find. Here is a short excerpt to give an overview of important publications for the IoT development and to show the areas that are still lacking in research, e.g. IoT security patterns.

3.1 General Patterns for IoT Design

Design, privacy or security patterns are only a few examples of potential solutions to common development problems that IT experts face on a daily basis. The landscape of patterns and architectures that support the IoT market is huge and research can be found in many articles like the following.

Applying IoT Patterns to Smart Factory Systems. Patterns are a common way to describe abstract solutions to design problems and can be used to analyse and understand computer systems in an easier way. Reinfurt et al. [RFBL17] describe in their paper IoT-specific patterns that help to design IoT systems that can be applied to the domain of smart factory systems. These patterns cover different areas and operation modes like device communication and management as well as energy supply types.

IoT Architectural Styles. Other than design patterns, different architecture styles can be utilized when creating IoT systems. Muccini et al. [MM18] provide a number of abstract IoT reference architectures in their paper. By conducting a systematic mapping study a set of 63 papers was chosen out of over 2,300 potential works. The results of this study will help to classify existing and future approaches for IoT styles and patterns at the architectural level.

Landscape of Architecture and Design Patterns for IoT Systems. In order to get a better idea for the landscape of patterns and architectures that have accumulated over the years in research, Washizaki et al. [WYH+19, WOH+20] analysed the successes and failures of used patterns for IoT systems. Because of limited documentation and a lack of successfully executed implementations, there is a lot of room for improvement in the development of IoT-specific patterns and architectures.

3.2 Security-specific IoT Patterns

Unlike general frameworks that can be utilized by developers to simplify their work and help to solve problems on a more abstract level, domain-specific IoT patterns that also focus on security are difficult to find. Security objectives are a huge part of every computer system to protect the user's data and identity but, unfortunately, patterns that could prevent unauthorized access are lacking. There exist many ways an attacker can compromise IoT systems that are especially easy to exploit because of a vast variety of interfaces and connections that lead to vulnerabilities.

Architectural Patterns for Secure IoT Orchestrations. Fysarakis et al. [FSP+19] sketch the SEMIoTICS approach to create a pattern-driven framework that is based on already existing IoT platforms. Aiming to guarantee secure actuation and semi-automatic behaviour, the SEMIoTICS project utilizes patterns to encode dependencies between security, privacy, dependability and interoperability properties of smart objects.

Towards a Collection of Security and Privacy Patterns. Organized in an hierarchical taxonomy, Papoutsakis et al. [PFS+21] collect and categorize a set of security and privacy patterns. While giving the reader an overview of security- and privacy-related objectives that are relevant in the IoT domain, the goal of this paper is to match these properties to their corresponding patterns. This usable pattern collection should guide developers to create IoT solutions that are secure and privacy-aware by design.

A Decade of Research on Patterns and Architectures for IoT Security. Over the last three years, Rajmohan et al. [RNF20b, RNF20a, RNF22] published different papers that review the research work regarding IoT patterns and architectures for IoT security and privacy. Although there is a rise in number of publications, there is not yet an approach of applying architectures and patterns in a way that addresses security not only on the architectural but also on the network or IoT devices level.

Although the IoT research in academia is thriving and there exist a variety of design patterns, architectures and frameworks that can be utilized in the development of modern IoT devices, the lack of focus on security objectives is worrisome. Security patterns can not only guide developers to build safer systems but lessen the workload by taking already existing solutions for abstract problems. This master's thesis will make an important contribution in increasing the accessibility to these already existing IoT security patterns for future work in this field.

4 Methodology

This chapter introduces the research methodology and review protocol that is based on the systematic literature review (SLR) (in Section 4.1) and used in order to search for the needed primary studies to answer the following research questions in Section 4.1.1. The taxonomy that is needed to gather information from the collection of studies is then defined in Section 4.2.

4.1 Systematic Literature Review

According to Kitchenham et al. [KC07] SLR is an analysis of publications about a certain topic that collects and critically evaluates multiple research studies or papers in a systematic way. Its purpose is to provide an exhaustive summary of the available literature relevant to a specific research question in a transparent and reproducible way. The following points utilize the review protocol that is used to conduct the literature review in a strategic manner and consist of the research questions that should be answered, selection criteria the found papers need to fulfil and a search strategy on how to browse databases in order to find the most relevant publications.

4.1.1 Research Questions

This thesis aims to answer the following research questions (RQs):

RQ1 Which layer in the IoT World Forum Reference Model is covered by the least security patterns?

RQ2 Which security goal is addressed by security patterns the least and how can the coverage be improved?

RQ3 How many of the OWASP top ten most common IoT vulnerabilities are solved by utilizing these security patterns?

4.1.2 Selection Criteria

To minimize bias in the search and selection of our primary studies for this master's thesis, we predefined guidelines that need to be fulfilled for a study to be used in our analysis. This approach leads us to the following inclusion and exclusion criteria:

Inclusion Criteria:

IC1 A primary study must contain (one or more) security pattern that is applicable to an IoT system.

IC2 A primary study must target the IoT field, either in a general or specific application domain of IoT.

IC3 A primary study must discuss security objectives for system design, architecture or infrastructure.

Exclusion Criteria:

EC1 Papers not written in the English language are filtered out.

EC2 Papers that discuss design, privacy or misuse patterns as well as security architectures for the IoT domain are excluded.

EC3 Papers that are not peer-reviewed are also ruled out.

4.1.3 Search and Selection Strategy

This section focuses on the selection of appropriate primary studies that can serve as a basis to answer the aforementioned research questions. The goal is to collect as many primary studies as possible that are relevant for this thesis. The strategy to find papers that discuss security patterns is divided into two main parts: automatic and manual search. First we let a set of search engines find example studies in an automated approach with specific keywords. Here we also apply the mentioned inclusion and exclusion criteria to exclude any unwanted results ahead of time. Second we conduct a manual search in order to fill the gaps with any relevant scientific papers that were missed by the machine. In an attempt to avoid duplicates, the last stage of the search process consists of merging identical papers that were found by different search engines. The exact procedure is described in the following points and illustrated in Figure 4.1.

1. Database search:

As our primary search engines, we utilized five well known databases for scientific publications: IEEE Xplore[1], ACM Digital Library[2], ScienceDirect[3], Web of Science[4] and Scopus[5]. Because Scopus and ACM Digital Library already reference SpringerLink[6], we could exclude the former as well as Researchgate[7] and Google Scholar[8] which also include many non-peer-reviewed and non-English papers. Based on Kitchenham et al. [KC07] we created keywords that generate papers that are relevant for the answers to our research questions. This led us to the following search query that was used for our initial database search on May 21st, 2022:

(*"Internet of Things"* **OR** *"IoT"* **OR** *"Cyber Physical Systems"* **OR** *"Web of Things"*)
AND
(*"Security Pattern"* **OR** *"Security Design Pattern"*)

[1] https://ieeexplore.ieee.org, last accessed: 21.05.2022.
[2] https://dl.acm.org, last accessed: 21.05.2022.
[3] https://www.sciencedirect.com, last accessed: 21.05.2022.
[4] https://access.clarivate.com, last accessed: 21.05.2022.
[5] https://www.scopus.com, last accessed: 21.05.2022.
[6] https://link.springer.com, last accessed: 21.05.2022.
[7] https://www.researchgate.net, last accessed: 21.05.2022.
[8] https://scholar.google.com, last accessed: 21.05.2022.

For each search the query string needed to be slightly modified to fit each database's advanced search functionality and guidelines.

2. Manual search:

By utilizing the snowballing strategy that was introduced by Wholin and Prikladnicki [WP13], we searched manually for further literature that was missed by the automatic database inquiry. Following references of the already found papers, we looked for relevant publications that include further security design patterns that can be useful for our study. After a few iterations we found a set of papers that met the inclusion criteria of our SLR. After eliminating any duplicates of papers that were already discovered by the initial database search, we could add eleven more articles to our database search result. By including these manual publication findings, our RQs can be answered in a more thorough way.

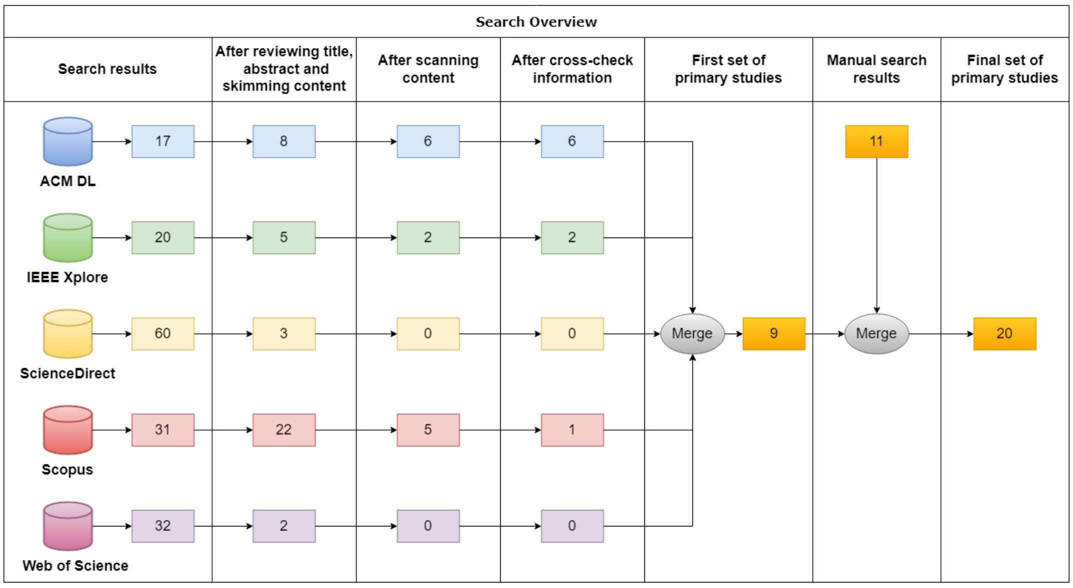

Figure 4.1: Overview of the search and selection procedure

4.2 Taxonomy of the Research Area

In this section the taxonomy of IoT security patterns is introduced. The main purpose of the taxonomy classifications is to simplify the extraction and comparison of patterns that are described in the primary studies. Subsection 4.2.1 defines a scheme on how to structure the pattern descriptions, then Subsection 4.2.2 depicts what the general architecture of an IoT system looks like and Subsection 4.2.3 shows which security goals are the main focus of the extracted patterns as the last part of the taxonomy classifications.

4.2.1 Pattern Description with POSA

All patterns in the catalogue will be listed utilizing the Pattern-Oriented Software Architecture (POSA) form, so that the descriptions of IoT security patterns in this master's thesis are presented in a unified form. Although POSA is a popular format, not all patterns in the analysed publications are presented in this way or do have a standardized format at all. Another very popular form is Gang of Four (GoF) which is used in the "Design

Patterns: Elements of Reusable Object-Oriented Software" book [GHJV94]. But we chose POSA because it is a well-accepted and widely used pattern description format that provides standard templates [Bun14] and is used in the book with the same name by Buschmann et al. [BMR+96]. Other pattern description formats can also be easily converted and mapped to the POSA form that describes design patterns with the following keywords: *Intent, Example, Context, Problem, Structure, Solution, Dynamics, Consequences, Implementation, Example Resolved, Known Uses, and Related Patterns.* Because of space limitations and to structure the pattern catalogue in a concise and clear way, we reduced the used keywords to the most important ones which are *Intent, Problem, and Solution.*

4.2.2 IoT World Forum Reference Model

In order to keep our pattern catalogue organized, we utilize the IoT architecture of the IoT World Forum Reference Model[9] (WFRM) for a layered hierarchy the found security patterns can be categorized in. This model is an attempt of the IoT World Forum to create a common model to guarantee interoperability between all IoT components. It is important to note that this model is not representative of a real IoT system but only an abstraction that was created in order to simplify IoT product development between vendors. A summary of each layer with its functionality is shown in Table 4.1.

L1 Physical Devices & Controllers. The outermost layer consists of the physical devices that are the "things" in the IoT architecture as well as the sensors and edge node devices connected to them. Almost any electronic device that has a network connection is either an IoT device or has one embedded. The most common types are:

- **Sensors:** Optical sensors, temperature sensors,...
- **Security devices:** Security cameras, audio recording devices, motion sensors,...
- **Smart wearable devices:** Watches, earbuds, glasses,...
- **Intelligent appliances:** Smart thermostats, intelligent refrigerators, connected televisions,...
- **Actuators**

L2 Connectivity. The communication and processing unit has a wide range in the IoT model. From edge node devices to the cloud up to the next layer, Edge Computing, this layer is responsible for the information transport in the IoT network. Implementations vary from case to case with single to multiple technology solutions. Wired, wireless or cellular as well as multi-tiered architectures are just a few alternatives, that can be used alongside Field Area Networks, that are suitable for a mix of private and public package transport.

L3 Edge Computing. The previously mentioned Edge Computing layer is also called "Cloud Edge" or "Cloud Gateway" computing. Its purpose is to interface the data and control levels to the higher cloud, SaaS or enterprises layers and therefore makes this layer essential in any IoT architecture model. This layer is not only able to route to software functions and convert protocols but also to execute logic to reduce the decision making time and decrease latencies in the execution phase.

[9] https://www.m2mology.com/iot-transformation/iot-world-forum, last accessed: 10.06.2022.

L4 Data Accumulation. For an efficient data stream there need to be storages that are responsible for queues between different architecture layers. Velocity, volume and variety should be continuously provided for incoming data in an IoT system. In order to be able to process and prepare the input stream for delivery to the next stage, this layer is implemented as an intermediate storage facility in the IoT WFRM. From simple solutions like the usage of SQL to more advanced distributed storage and processing systems like Hadoop and Hadoop File System, Mongo, Cassandra, Spark or other NoSQL versions, each IoT system has its own data management architecture.

L5 Data Abstraction. To get an overview of the incoming data, the Data Abstraction layer analyses and then organises information from different IoT devices, maps them into given schemata and prepares the extracted data for further upstream or downstream processing. For a simplified data flow the software frameworks for publish/subscribe or data distribution services are key elements. These architectures guarantee low latency high performance communication between Edge Computing, Data Accumulation, Applications and User Processes.

L6 Application. Just as its name suggests, this layer is where all important application logic is executed. To just name a few, there are IoT related applications, user patterns, statistical analysis, alarm systems, logistical processes, monitoring and optimization techniques as examples that come to mind.

L7 Collaboration & Processes. User interaction is the significant point in this layer of every IoT system. Data that was processed in lower layers is integrated into applications that are responsible for user engagement and economic value. The big challenge is to utilize IoT devices and its architecture model in such a way that it brings growth and value to the businesses and/or social good.

Table 4.1: Summary of IoT architecture layers[9]

No.	IoT Layer	Functionality
L7	Collaboration & Processes	User Interaction, Business Processes
L6	Application	Reporting, Control, Analytics
L5	Data Abstraction	Aggregation, Access
L4	Data Accumulation	Storage Facilities
L3	Edge Computing	Data Element Analysis, Transformation
L2	Connectivity	Message Transmission, Processing
L1	Physical Devices	The “Things”

4.2.3 Security Concerns

Each extracted IoT security pattern focuses on a specific security objective it tries to protect. The following paragraphs describe the most important security goals for IoT systems that should be covered by these security patterns for a secure software design. Hereby, the privacy goal is not further analysed in this master’s thesis because of the exclusion criteria that were introduced in Subsection 4.1.2. Specifically EC2 states that all papers that discuss privacy patterns are excluded from the analysis.

Confidentiality is the objective to ensure that data resources or information are protected against unauthorized disclosure. Encryption and setting passwords are the most common measures to enforce this security objective.

Integrity has the goal to protect data resources or information against unauthorized changes, tampering, destruction or loss. This security objective ensures that the data stays accurate and reliable.

Availability means that data resources or information are accessible to authorized users. It provides an assurance that a system and its data can be accessed by authenticated users when they are needed.

Authentication is the process of determining if an entity's identity is who he or she claims to be. Before a user can access stored information, he or she must prove their identity and permission to access the data.

Authorization is the process of determining if a user has permission to access or use a certain resource. It is usually enforced in combination with authentication by access control mechanisms.

Privacy is a subset of security that focuses on personal information. It describes the ability to protect sensitive data about personally identifiable information.

5 IoT Security Pattern Catalogue

This chapter lists all IoT security patterns that were collected during the previously explained search process in form of a catalogue for IoT developers. The pattern table 5.2 is divided into the seven layers of the World Forum Reference Model (WFRM). Each pattern corresponds to a specific layer and is described by its name, a list of vulnerabilities from the OWASP top ten it can (possibly) solve as well as security objectives that are either addressed by this particular pattern (+) or not mentioned in its original description (-).

For a more efficient use of space in the table that makes the information more concise but still easy to understand, we used specific abbreviations. Most of them were already previously introduced in this master's thesis. For a compact overview of the defined abbreviations, the following Table 5.1 can be utilized.

Table 5.1: Legend for pattern lookup table

WFRM Architecture Layers		**OWASP Top Ten Vulnerabilities**	
L1	Physical Devices & Controllers	T1	Weak, guessable or hardcoded passwords
L2	Connectivity	T2	Insecure network services
L3	Edge Computing	T3	Insecure ecosystem interfaces
L4	Data Accumulation	T4	Lack of secure update mechanisms
L5	Data Abstraction	T5	Use of insecure or outdated components
L6	Application	T6	Insufficient privacy protection
L7	Collaboration & Processes	T7	Insecure data transfer and storage
Security Objectives		T8	Lack of device management
C	Confidentiality	T9	Insecure default settings
I	Integrity	T10	Lack of physical hardening
A	Availability		
Ac	Authentication		
Az	Authorization		

Table 5.2: IoT security pattern lookup table

Layer	#	Pattern Name	Solution for										Security Concerns				
			T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	C	I	A	Ac	Az
	5.1.1	Hardware IoT Security		✓		✓	✓						+	-	-	+	-
	5.1.2	Secure IoT Thing		✓	✓				✓				+	+	+	-	-
L1	5.1.3	Secure Sensor Node			✓		✓		✓				+	+	+	+	+
	5.1.4	Security Segmentation			✓								-	-	-	+	+
	5.1.5	Trusted Platform Module					✓		✓				+	+	-	+	+
	5.2.1	Authenticated Channel		✓					✓				-	+	-	+	-
	5.2.2	Encrypted Channel		✓					✓				+	-	-	-	-
	5.2.3	Middleware Message Broker			✓				✓				+	-	-	-	-
	5.2.4	Middleware Self-contained Message							✓				+	-	-	-	-
	5.2.5	Orchestration of SDN Network Elements			✓				✓				+	-	-	-	-
	5.2.6	Outbound-Only Connection						✓	✓				+	-	-	-	+
L2	5.2.7	Password Based Key Exchange	✓						✓				+	-	-	-	-
	5.2.8	Safe Channel		✓					✓				-	+	-	-	-
	5.2.9	Secure Remote Readout							✓				+	+	-	+	-
	5.2.10	Signed Message							✓				-	+	-	+	-
	5.2.11	Symmetric Key Cryptography							✓				+	-	-	+	-
	5.2.12	Third Party Based Authentication							✓				+	+	-	+	-
	5.2.13	Trusted Communication Partner						✓	✓				+	-	+	+	+
	5.2.14	Web of Trust							✓				-	+	-	+	-
	5.3.1	Fog Computing		✓	✓				✓				+	+	+	+	+
L3	5.3.2	Fogxy		✓	✓				✓				-	-	+	+	+
	5.3.3	Secure Cloud-based IoT Architecture			✓				✓				+	+	+	+	+
	5.4.1	Encrypted Storage						✓	✓				+	-	-	-	-
L4	5.4.2	Redundant Storage							✓				-	-	+	-	-
	5.4.3	Safe Storage						✓	✓				-	+	-	-	-
	5.5.1	Alignment-based Translation Pattern			✓								-	-	+	-	-
L5	5.5.2	BlockBD			✓				✓				+	+	+	-	-
	5.5.3	Discovery of IoT Services			✓								-	-	+	-	-
	5.5.4	Flow-based Service Composition		✓					✓				+	-	-	-	-

Table 5.2: IoT security pattern lookup table

Layer	#	Pattern Name	Solution for										Security Concerns				
			T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	C	I	A	Ac	Az
L5	5.5.5	IoT Gateway Event Subscription			✓				✓				+	-	+	-	-
	5.5.6	IoT SSL Cross-Layer Secure Access			✓								+	+	-	+	+
	5.5.7	Middleware Message Translator			✓								-	-	+	-	-
	5.5.8	Middleware Simple Component											+	-	-	-	-
	5.5.9	D2D REST Request/Response			✓								+	-	+	-	+
	5.5.10	Server Sandbox									✓	✓	+	+	+	-	-
	5.5.11	Service Orchestration			✓				✓				+	-	-	-	-
	5.5.12	Translation with Central Ontology			✓								-	-	+	-	-
L6	5.6.1	Access Control to Physical Structures										✓	-	-	-	+	+
	5.6.2	Alarm Monitoring								✓			-	-	-	-	+
	5.6.3	Audit Log								✓			-	+	-	-	-
	5.6.4	Authenticated Session			✓								-	-	-	+	-
	5.6.5	Authorization Enforcer								✓			-	-	-	-	+
	5.6.6	Encrypted Processing							✓				+	-	-	-	-
	5.6.7	Fault Management								✓			-	-	+	-	-
	5.6.8	File Authentication							✓				-	-	-	+	+
	5.6.9	Matrix Authentication							✓				-	-	-	+	+
	5.6.10	Minefield					✓				✓		-	+	-	-	-
	5.6.11	Remote Authenticator/Authorizer							✓				-	-	-	+	+
	5.6.12	Role Based Access Control							✓				-	-	-	-	+
	5.6.13	Safe Processing							✓				-	+	-	-	-
	5.6.14	Secure Distributed Publish/Subscribe							✓				+	+	+	+	+
	5.6.15	Uptime								✓			-	-	+	-	-
L7	5.7.1	Account Lockout	✓										-	-	-	+	-
	5.7.2	Authentication Enforcer			✓								-	-	-	+	-
	5.7.3	Blacklist			✓					✓			-	-	-	+	+
	5.7.4	History-Based Authentication	✓										+	+	-	+	+
	5.7.5	Permission Control						✓		✓	✓		+	-	-	-	+
	5.7.6	Personal Zone Hub						✓					+	-	-	-	+

Table 5.2: IoT security pattern lookup table

Layer	#	Pattern Name	Solution for										Security Concerns				
			T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	C	I	A	Ac	Az
	5.7.7	Relays											-	-	-	-	+
L7	5.7.8	Single Access Point			✓								-	-	-	+	+
	5.7.9	Whitelist			✓					✓			-	-	-	+	+

5.1 Physical Devices & Controllers

The first set of patterns that are introduced in this section all belong to the Physical Devices & Controllers layer of the IoT WFRM. These patterns typically enhance the security aspect of the IoT devices themselves like the Secure IoT Thing 5.1.2 or their hardware components like the Trusted Platform Module 5.1.5 or Secure Sensor Node 5.1.3.

5.1.1 Hardware IoT Security [SID+18]

Intent. The goal of the Hardware IoT Security pattern is to enable IoT device vendors to keep their new devices up-to-date by adding cryptographic co-processors externally to the hardware. Furthermore, all existing devices should be prepared for future security challenges and demands.

Problem. The solution is constrained by the following forces:

- Offload cryptography. Separate business logic from security functionality to simplify updating and certification of devices.
- Costs. Secure device with resourceful cryptography as well as minimal costs and power consumption.
- Security "ages". The typical lifetime of an IoT device is much longer than the update cycle of a cryptographic function. Existing micro-controllers may not have the required resources for the new version.
- Software bugs. The implementation of an algorithm is never as secure as the algorithm itself because of physical limitations.
- Malicious attacks. A specialized processor is often required for the mitigation of physical attacks like power analysis or timing attacks.
- Provisioning. During production each device is assigned a private key. Only the manufacturer has access to the firmware the key is embedded in and must hereby be trusted.
- Time-to-market. Each change in the underlying cryptography leads to a re-certification and auditing process of the entire developed product.

Solution. Use exchangeable cryptographic co-processors to secure IoT devices.

5.1.2 Secure IoT Thing [BFAO21]

Intent. The Secure IoT Thing pattern's aim is to secure any (physical or virtual) entity that is connected to sensors and actuators.

Problem. The solution is constrained by the following forces:

- Threat handling. All identified threats from attackers must be mitigated or resolved.
- Diversity of things. Potentially malicious product owners as well as the devices themselves must be uniquely identifiable.
- Vulnerability. Low-cost devices typically lack computational power for strong security defences.

- Complexity. The complexity of a system directly correlates to its security level by increasing the attack surface.

Solution. In general, perfect security does not exist. By implementing basic security defences in every product, manufacturers would be able to counter a huge list of identified threats. Unfortunately, cost and performance goals are the reason a thorough security verification is impossible to achieve. But only when every security threat is eliminated, a device can be labelled as secure.

5.1.3 Secure Sensor Node [OFA20]

Intent. As part of an IoT system, the Secure Sensor Node is responsible for receiving, storing and sending data between different nodes in a secure way.

Problem. The solution is constrained by the following forces:

- Integrity, confidentiality, and availability. The sensor node has to make sure that transmitted data cannot be altered with and is at all times accessible to authorized entities.
- Auditability. Authorized entities should be able to reconstruct and monitor historical as well as ongoing events.
- Overhead. Processing times of sensed data should be kept as small as possible.
- Power consumption. Because of a known power constraint of IoT devices, the solution must use power economically.

Solution. As an extension of the Sensor Node pattern, this solution provides security features that enable the sensor node to communicate and store data in a secure way. Additionally, for auditing purposes a record of all occurring events is collected. Therefore, the implementation of additional encryption techniques, logging mechanisms as well as authentication and authorization controls are required.

5.1.4 Security Segmentation [FFYW20]

Intent. The purpose of the Security Segmentation pattern is to partition IoT devices and entities into subnetworks according to their security requirements.

Problem. The solution is constrained by the following forces:

- Heterogeneity. IoT devices are equipped with different levels of security features which need to be considered.
- Identity. A unique identification process is required to keep an overview of all participating devices in a specific network.
- Security level. Prioritization policies are needed to avoid conflicts between authorization rules from different sources.
- Attack surface. The complexity of a system and its variety of devices correlates to its increasing vulnerability.
- Compliance. Device heterogeneity hinders the enforcement of compliance rules.

- Threats. CPSs have an increased number of threats that may have safety consequences on the IoT system.

Solution. Divide the network into small units. Identified and more well-known devices are placed into the sensitive partitions. A combination of firewalls and VLANs (Virtual Local Area Networks) is used to achieve the segregation of devices.

5.1.5 Trusted Platform Module [MnF20]

Intent. The Trusted Platform Module (TPM) pattern is intended for verification of hardware and software as well as assuring trustability before execution.

Problem. The solution is constrained by the following forces:

- Attestation. For secure software execution attestation is used to verify that a software platform is legitimate.
- Secure cryptography. The generation of keys must be secure for a reliable encryption/decryption process.
- Secure signature generation. For message authenticity the secure generation of signatures is needed.
- Data integrity. Only authorized entities should be able to manipulate data.
- Secure processing. During the execution of software information should not be leaked or illegally modifiable.
- Dynamic root of trust (DRT). As a starting point for the “chain of trust” the DRT refers to the first reliable measurement of the chain.
- Authenticated boot. The implementation of the transitive “chain of trust” relies on the monitoring of the boot process.
- Sealed-bound key. These keys are bound to the platform state during their generation to ensure their origin.

Solution. The TPM is a hardware module that can attest hardware as well as software with its integrated cryptographic services. The verification process utilizes a set of cryptographic keys as well as the concept of “chain of trust”. Further applications of the TPM include the storage of secret keys and the execution of encryption or decryption respectively.

5.2 Connectivity

A very important part of every computer system is its transportation layer that is responsible for an efficient and safe data exchange in the network. The WFRM for IoT systems calls this the Connectivity layer. The goal of keeping connections between different devices or architecture layers safe is supported by a variety of patterns like Authenticated Channels 5.2.1, Password Based Key Exchange 5.2.7 or even Signed Messages 5.2.10.

5.2.1 Authenticated Channel [PFS+21]

Intent. The Authenticated Channel pattern ensures that the communication routes of an IoT system are trustworthy and reliable even for private conversations.

Solution. To achieve this goal, an authenticated channel should require mutual authentication of all communication partners. Also, the secret key should be verifiable by at least one peer to confirm the common secret that is used for a safe channel. Lastly, no communication partner should be able to retrieve old session keys, even when long-term secret keys, like certificates, are known. Therefore, forward secrecy is a major key point of ensuring attackers cannot eavesdrop private conversations.

5.2.2 Encrypted Channel [PFS+21]

Intent. The goal of the Encrypted Channel pattern is to keep sent information confidential during transmission across networks.

Problem. Public networks and channels are easy to be intercepted and eavesdropped by malicious third parties.

Solution. The easiest way to achieve confidentiality is encryption. By utilizing the typical TCP handshake and exchanging cryptographic information, communication partners are able to establish an encrypted channel between them.

5.2.3 Middleware Message Broker [TWG+18]

Intent. The IoT Artifact's Middleware Message Broker is a pattern whose purpose is to manage the communication between IoT artifacts.

Problem. In order to avoid exposing internal operations of different independent components in the middleware, multiple interfaces that enable point-to-point connections should not be implemented. The usage of point-to-point interfaces can also lead to problems regarding the management quality of service requirements, the matching of security constraints and dynamic reconfiguration.

Solution. The Message Broker acts as a common messaging interface and therefore solves the mentioned problem by hiding the internal operations of components, using logical names for the communication partners and functioning as a middle-man in the message transmission process. For a successful message delivery of the broker each message must contain a label and the information to send as payload.

5.2.4 Middleware Self-contained Message [TWG+18]

Intent. The IoT Artifact's Middleware Self-contained Message pattern requires each message to consist of the complete information needed for execution of a specific action.

Problem. Middleware components should only need to store the minimal required amount of resources. Therefore, messages within must contain all information that is needed for their interpretation and execution. Regardless of when the event or command is interpreted, the message should obtain complete information on how to execute it.

Solution. To identify different messages, the middleware component processes and routes them based on a distinct set of types. Each message that travels downstream can have an answer that is routed in the reverse direction. This enables conversations to be distinguished by matching downstream travelling messages with upstream ones.

5.2.5 Orchestration of SDN Network Elements [TWG+18]

Intent. This pattern's main controller aims to orchestrate all SDN elements (virtual switches) of a network. Hereby, the orchestrator manages the transmission of flow and control messages between different network topologies and configurations.

Problem. The main issue of domain-focused IoT deployments is their isolated nature.

Solution. One way to fix this problem is to move the communication from the device layer upwards into a higher part of the architecture model. The network layer, for instance, can implement this pattern which is also used in virtual SDNs, where a single orchestrator monitors all elements (virtual resources or instances) of the network. This single virtual LAN consists of virtually connected networks which are located in different locations. By utilizing this main controller the physical separation of networks disappears.

5.2.6 Outbound-Only Connection [RBF+17]

Intent. An Outbound-Only Connection pattern allows, like the name suggests, only communication that is initiated from the device itself while blocking all incoming connection requests. This prevents attackers from sending malicious messages and infecting the receiver.

Problem. The solution is constrained by the following forces:

- Security. All unsolicited messages must be ignored.
- Functionality. While blocking malicious requests, the original functionality of the device should not be affected.
- Constraints. Because of resource limitations, IoT devices are constrained in their solutions for communication control.

Solution. A solution for this problem would be to only allow the devices themselves to initiate communication. Furthermore, devices should only accept incoming messages that are responses to previously sent connection requests.

5.2.7 Password Based Key Exchange [SOP+16]

Intent. The goal of the Password Based Key Exchange pattern is to secure communication with a common secret that is ideally "short" and easy to remember.

Problem. The solution is constrained by the following forces:

- Limited access. It is difficult to securely load or store a private key file on a device.
- User experience. The original functionality of the device should not be affected by the implemented security mechanisms.

- Eavesdropping. Expect that the integrity of messages will be violated by malicious third parties.

Solution. A common secret between the communication parties is used to generate session key pairs. These random session keys secure the communication channel further.

5.2.8 Safe Channel [PFS+21]

Intent. The purpose of the Safe Channel pattern is to guarantee integrity during message transmission on the transport layer by utilizing certificates.

Problem. Public communication channels are lacking security and are therefore open for interceptions by malicious third parties who can tamper with sensitive information.

Solution. As an example, the file channel integrity tool in Apache Fume is an efficient distributed system that specializes on the transmission, collection and aggregation of large log data from different sources. Also, the Transmission Control Protocol (TCP) supports the detection of data tampering by implementing a (non-cryptographic) checksum.

5.2.9 Secure Remote Readout [URZ15]

Intent. The Secure Remote Readout pattern supports a secure implementation of the remote readout functionality.

Problem. Measurements that are sent from the gateway to the remote entity need to be protected from malicious third parties.

Solution. A dedicated hardware (Security Module) is in charge of securing information that is transmitted from the gateway over insecure WAN. En-/Decryption, digital signatures, key generation and secure key storage are all cryptographic functionalities the security module provides. The digitally signed and encrypted measurements are exchanged over a Transport Layer Security (TLS) connection between the gateway and the remote entity.

5.2.10 Signed Message [PFS+21]

Intent. The purpose of the Signed Message pattern is to guarantee the authenticity, integrity as well as non-repudiation of all exchanged messages.

Solution. The sender of a message is associated with it by utilizing their digital signature during the generation and/or exchange process.

5.2.11 Symmetric Key Cryptography [SOP+16]

Intent. The purpose of Symmetric Key Cryptography is to hinder malicious third parties from eavesdropping or intercepting private message exchanges.

Problem. The solution is constrained by the following forces:

- Computational complexity. The applied security protocol should not affect the required computational overhead.
- Resource constraints. IoT devices with limited storage capacity may be unable to store long keys that are needed for asymmetric cryptography.

Solution. Symmetric cryptography utilizes a common secret that is shared between communication partners to encrypt exchanged messages. To authenticate the communicating parties before a trustworthy connection can be established, a handshake is required.

5.2.12 Third Party Based Authentication [SOP+16]

Intent. The Third Party Based Authentication pattern aims to establish trust between parties that are unknown to each other. The authenticated communication channel is then secure against the intrusion of malicious attackers who try to eavesdrop or intercept the communication.

Problem. The solution is constrained by the following forces:

- User experience. The functionality of a system should not be affected by the implemented security mechanisms.
- Computational complexity. Compared to symmetric cryptography, asymmetric cryptography, that uses public/private key pairs, is more time consuming.

Solution. The authentication process of communication parties can be handled by public key cryptography based on certificates. Then a session key is generated and used for safe communication between the parties.

5.2.13 Trusted Communication Partner [RBF+17]

Intent. The Trusted Communication Partner pattern suggests to limit the range of communication partners of a device to only trusted sources. Unknown devices may pose a security risk and try to access the device or launch a denial of service attack. Therefore they should be blocked for their connection requests.

Problem. The solution is constrained by the following forces:

- Uninvited communication. Connection requests from unknown sources should be blocked.
- Functionality. Implemented security mechanisms should not affect the operation of a system.
- Flexibility. It should not matter if communication partners are static or frequently changing.

Solution. To connect your device to trusted sources, implement a list of trusted parties. The device only allows incoming messages from the listed communication partners. If a connection request comes from an unknown source, the attempt should be blocked and investigated.

5.2.14 Web Of Trust [SOP+16]

Intent. The Web of Trust represents a structured way to establish trust in a network of communication parties without a central authority.

Problem. The participants do not want to rely on an outside party to authenticate the communication partners.

Solution. Only by verifying the fingerprint themselves, can each communication party build trust in the key individually. A central authentication authority is not needed in this case.

5.3 Edge Computing

As a way to connect the data and control layers of an IoT system to the higher cloud, the Edge Computing layer of the WFRM architecture functions as a kind of interface. Patterns, that contribute to this cloud infrastructure, are for example Fog Computing 5.3.1 or also Fogxy 5.3.2.

5.3.1 Fog Computing [SFI16]

Intent. Between the cloud and edge devices there is a virtualized platform called fog computing. The platform allows the cloud to communicate with the devices in the network and to provide computation, storage as well as network services.

Problem. The solution is constrained by the following forces:

- Large number of nodes. A large amount of data is produced by the nodes in the network that must be aggregated and analysed.
- Latency. Because of the applications' geographical distance from the cloud data centres, data transmission is sensitive to latency.
- Mobility. Edge devices are typically mobile but this should not affect the efficiency of the IoT system.
- Location awareness. Some applications need information about the geographical location of their device.
- Heterogeneity. Different IoT components, like routers, switches, access points, end user devices with their own protocols and interfaces, need to be manageable in a uniform way.
- Transparency. Limitations of a device's storage, communication or computational power need to be transparent to the user.
- Big data analytics. Central analysis in data centres of the huge amount of data from edge devices is not feasible.
- Cloud support. For permanent storage and long-term computations cloud data centres may still be necessary.
- Scalability/Flexibility. Constant change demands a dynamic approach to resource and device allocation.

- Multi-tenancy. Resource sharing is a recommended way to support multiple applications.
- Multiplicity of providers. Distributed IoT systems need to be able to orchestrate consistent policies across multiple providers.
- Security. Access control to data from different devices, the fog or the cloud needs to be implemented for identity authentication and authorization.
- Filtering. In order to avoid inefficient bandwidth usage and security issues, filtering of (mostly unnecessary) data is recommended.

Solution. Fog Computing can be described as a platform that is closer to the devices themselves but at the same time provides cloud computing-like services like storage, computation and networking. In addition, fog computing offers benefits like low latency, bandwidth efficiency, location awareness, security and filtering services. While most of the data is immediately processed at the device to improve response times, aggregated data can be send to the cloud for further analysis. In general, the fog structures the communication between edge devices and the cloud.

5.3.2 Fogxy [STB18]

Intent. The purpose of the Fogxy pattern is to provide an architectural solution for applications whose availability, real-time or low latency requirements cannot be accomplished by utilizing cloud-only implementations.

Problem. The solution is constrained by the following forces:

- Low latency & real-time interaction. The unavoidable communication overhead and geographical distribution make a central data centre unsuitable to accommodate the real-time, short latency requirements of an IoT application.
- Standardization, heterogeneity, and interoperability. There is no standard for fog computing which ranges from embedded devices to virtualized platforms. Devices and components are distributed across the system and the interoperability of different providers must be guaranteed.
- Resource limitation. Because of their specific purpose devices in the fog layer contain limited storage, computing power and energy resources.
- Geographical distribution. Unlike the cloud, which resides in central data centres, fog nodes are widespread even to remote locations without internet connection.
- Missing & inadequate infrastructure. The insufficient cloud infrastructure as well as higher network speed and availability requirements cause an increased network load and possible restrictions.
- Interplay with the cloud. For an efficient collaboration cloud components should handle computationally or memory intensive tasks while embedded devices perform real-time critical ones.
- Access control. To avoid security problems, an authentication and authorization process must regulate which components can use services and which cloud data can be accessed.
- Location awareness. In order to efficiently perform task distribution and service provisioning, devices must be traceable.

Solution. By decoupling heterogeneous smart objects from centralized components, Fogxy implements real-time and availability requirements. System components that are located in different tiers can be distributed along a cloud-thing continuum. The decentralized approach enables smart objects and their services to facilitate real-time critical applications.

5.3.3 Secure Cloud-based IoT Architecture [Fer20]

Intent. The Secure Cloud-based IoT Architecture tries to overcome system threats and simplify the management of security. The implemented defence mechanisms include authentication, authorization, security logger, secure channel and firewalls.

Problem. The solution is constrained by the following forces:

- Quantity of things. Security management of a huge number of things is quite complex.
- Diversity of things. Different vendors and possible owners make controlling IoT devices difficult.
- Vulnerability of things. Because of limited resources the security defences of IoT devices are rather weak.

Solution. By utilizing a layered hierarchy, security protection for IoT data assets and communication channels can be defined.

5.4 Data Accumulation

As a storage facility between architecture layers, the Data Accumulation layer is responsible for processing transmission queues in the IoT WFRM for further delivery. The storage of data can have a lot of vulnerabilities that are addressed by patterns to help strengthen the confidentiality, availability and integrity properties. Common examples are the Encrypted Storage 5.4.1, Redundant Storage 5.4.2, and Safe Storage 5.4.3.

5.4.1 Encrypted Storage [KED+06]

Intent. To ensure that stolen data from a storage is useless to an attacker, the Encrypted Storage pattern acts as a second line of defence. It strengthens the protection mechanisms that already exist, such as firewalls and other server defences, and keeps sensitive data hidden from unauthorized users.

Problem. The loss of sensitive customer data can ruin the image of a company and it is very hard to recover from adverse publicity in modern society. But because the storage of private data, like credit card numbers, is sometimes unavoidable for certain services, the defences to protect user information have to be improved.

Solution. To avoid attackers from simply reading sensitive data, that is stored in a storage facility, critical data is encrypted before it is ever committed to disk. This way sensitive information has to be decrypted in memory before is can be accessed by anyone.

5.4.2 Redundant Storage [PFS+21]

Intent. The purpose of the Redundant Storage pattern is to safely manage the failure of cloud storage devices.

Solution. A solution would be the usage of a distributed redundant array of independent drives (RAID) or other redundant data system. In the case of device failure parity devices can be utilized to restore the lost data. The same applies to the recovery of a failed parity device.

5.4.3 Safe Storage [PFS+21]

Intent. The aim of the Safe Storage pattern is to guarantee that the contents of the storage remain complete, accurate and reliable. This means that the storage facility ensures the integrity of all sensitive data regardless of the length of its storage time nor its access frequency.

Solution. A few examples on how to implement this kind of integrity properties are Hash Checks, the Server Sandbox or Minefield patterns.

5.5 Data Abstraction

All data, that is accumulated from different layers of the WFRM architecture, gets sorted and prepared for further processing in the Data Abstraction layer. By aggregating and analysing the data, that is incoming from different IoT devices, a smooth transmission of information between the layers is guaranteed. A few examples for patterns, that provide these types of interfaces, are BlockBD 5.5.2, REST Request/Response 5.5.9 or the Server Sandbox 5.5.10.

5.5.1 Alignment-based Translation [TWG+18]

Intent. The Alignment-based Translation pattern is responsible for the semantic translation of RDF messages that are transmitted between IoT artifacts. The translation process is based on the sets of correspondences that are defined between artifacts' ontologies.

Problem. Because of different technologies, that are used in developing artifacts, their interoperation is hindered by a syntactic as well as semantic discrepancy.

Solution. Since other formats are uniformly transformable into RDF, this solution assumes RDF messages. Because the semantics of a message depend on the artifact, interoperability can only be achieved by mapping between URIs, parts of the RDF structure, transformations etc. For an easy way to submit messages, that need to be translated and published, the translator should implement the needed interfaces.

5.5.2 BlockBD [MFFMS19]

Intent. The aim of BlockBD is to improve security goals like traceability and veracity of stored data by implementing Blockchain technologies in a typical Big Data ecosystem. Thus, enhancing the overall quality of the algorithms' results.

Problem. The solution is constrained by the following forces:

- Speed. The velocity of the obtained insights or sometimes even real-time analysis is often critical in a Big Data system.
- Efficiency. The power consumption of the server clusters in a Big Data system, especially in ad-hoc solutions, is high enough that it has to be considered.
- New security problems. New technological solutions in Big Data systems most likely also lead to new security challenges.

Solution. BlockBD utilizes the implementation of an additional Blockchain layer that acts as a distributed ledger system. The layer is based on the already used nodes of the Big Data ecosystem and manages the stored data.

5.5.3 Discovery of IoT Services [TWG+18]

Intent. The Discovery of IoT Services pattern functions as a registry of IoT services for artifacts.

Problem. In order to notice and use an IoT service, the different platforms need to present these functionalities in an accessible and discoverable way.

Solution. For easy access a service catalogue is implemented that shows all registered IoT services.

5.5.4 Flow-based Service Composition [TWG+18]

Intent. The Flow-based Service Composition pattern's goal is to manage the interoperability of different IoT artifacts by modifying the execution flow.

Problem. Applications and services are defined as networks of "black box" processes that communicate over predefined connections. But there exists no pattern that specifies the execution flow of multiple IoT services.

Solution. This approach arranges a sequential execution flow for IoT services by connecting the black boxes and linking the output of one service to the input of another.

5.5.5 IoT Gateway Event Subscription [TWG+18]

Intent. The IoT Gateway Event Subscription pattern splits a gateway into two parts in order to achieve more flexibility in the data transmission process. The physical part is responsible for network access and communication protocols, whereas the virtual one manages the remaining gateway operations and services. This method makes it also possible to decouple data providers from their identities.

Problem. In order to enable communication between heterogeneous IoT devices, an asynchronous channel between the communication partners has to be implemented that allows data to be published, filtered and consumed.

Solution. This pattern utilizes the IoT gateway as a subscription mechanism and middleman between IoT artifacts. Thus, the gateway performs protocol conversion if needed. Like the publish/subscribe pattern, senders publish messages without knowledge of their receivers and subscribers receive only messages they expressed interest in. Therefore, messages need to contain the subscription information.

5.5.6 IoT SSL CROSS-Layer Secure Access [TWG+18]

Intent. The IoT SSL CROSS-Layer Secure Access pattern tries to implement external interfaces of every layer of the IoT system to ensure secure communication channels.

Problem. Layered architectures in an IoT system consist of many interfaces that increase the attack surface.

Solution. As a solution to secure the access of each architecture layer and its exposed REST APIs, the Secure Sockets Layer (SSL) pattern is implemented. This way only authenticated entities are able to access the external interfaces.

5.5.7 Middleware Message Translator [TWG+18]

Intent. The Middleware Message Translator's job is to transform an IoT artifact's middleware internal message format to any platform's own data formats.

Problem. In order to enable message transmissions between IoT artifacts, the middleware first has to syntactically and semantically translate any sent messages.

Solution. The message translator is able to convert messages between proprietary data models and the middleware component's own data format.

5.5.8 Middleware Simple Component [TWG+18]

Intent. The purpose of the Middleware Simple Component pattern is to ensure that the single responsibility principle is employed for each IoT artifact.

Problem. An issue can be, that in more complex systems different components may need to perform specific functions.

Solution. For simplified operations the responsibilities of components should be divided recursively until the needed granularity is achieved. This way testing and debugging can also be performed more efficiently.

5.5.9 REST Request/Response [TWG+18]

Intent. This pattern is a request/response solution for gateway connections in the device-to-device layer.

Problem. In order for the IoT gateway to receive data and execute orders, it needs to be able to communicate with IoT artifacts. Therefore, the artifact's middleware must be able to connect to the gateway and enable a message exchange between them.

Solution. This approach utilizes a Request/Response pattern that enables IoT artifacts to connect through a HTTP/REST API. The communication works by sending a request message to a replier system and after processing the response is returned.

5.5.10 Server Sandbox [KED+06]

Intent. The aim of the Server Sandbox pattern is to isolate the web server in order to protect it from damage that can be caused by undiscovered bugs in the server software.

Problem. Because it is an impossible task to ensure that a server is secure against all possible threats, it is important to protect the software in a manner that keeps the damage of a compromised system minimal.

Solution. The Server Sandbox strictly restricts the privileges that components of web applications have during runtime in order to minimize the possession of privileges that are unnecessary or can be easily abused to compromise the system. By removing the user access to all global privileges and replacing it with specific user and group privileges, the risk of allowing hackers easy access to the operating system is decreased. In addition, it is necessary to physically harden the system that is hosting the server.

5.5.11 Service Orchestration [TWG+18]

Intent. The Service Orchestration pattern manages the interactions between diverse, heterogeneous IoT services.

Problem. In order for heterogeneous IoT artifacts to be able to work together, many different services have to be orchestrated. It is important for the message flow to not only facilitate that sent data arrives at its destination, but also that specific actions need to be triggered on the way. A common issue is that these processes/actions are rarely reused but rather duplicated.

Solution. The main idea is to define a set of IoT nodes, e.g services and interfaces that are controlled by an internal element that assigns specific tasks. By managing all processes it is possible to orchestrate these diverse IoT services in a unified way.

5.5.12 Translation with Central Ontology [TWG+18]

Intent. The Translation with Central Ontology pattern focuses on the communication between IoT artifacts by semantically translating RDF messages that are based on a common data model.

Problem. A modularized central ontology on the basis of IoT and domain ontologies has to be created in order to achieve semantically common translations.

Solution. When the central ontology is based on already existing standards, advantages like better scalability and easier long-term maintenance are guaranteed. In addition, the translator should provide an interface for submitting messages to translate and publish the results.

5.6 Application

The Application layer of the WFRM architecture specifically focuses on application logic that manages, monitors, analyses and otherwise optimizes IoT related applications. The Audit Log 5.6.3, Authorization Enforcer 5.6.5 or Safe Processing 5.6.13 are a few examples that fall into this category of application-focused patterns.

5.6.1 Access Control to Physical Structures [FBDDLP07]

Intent. The Access Control pattern manages the authentication and authorization of access to physical structures which include alarm monitoring, relays and time schedules.

Problem. The solution is constrained by the following forces:

- Automatic access control. Each user is allowed access during a specific time of the day which needs to be automatically revocable.
- Automatic activation. Devices can be automatically enabled in a time-based manner.
- Time representation. Each day of the week needs to be represented in a uniform way.
- Access restriction. Depending on its identity or other characteristics, each visitor's access is restrictable.
- Boundaries. A clear definition of boundaries for each unit of access is needed.
- Variety. Each visitor has different access needs.
- Emergency. In case of an emergency, there need to be plans and policies on how to manage the usually closed units.
- Logging. Is is necessary to track each access time frame of a visitor.

Solution. In this approach the structure of the access control system is represented by a Role Based Access Control pattern. Furthermore, Alarm Monitoring and Relay patterns are implemented in order to manage time schedules. This way not only access control times can be monitored but also automatic actions are easier to facilitate.

5.6.2 Alarm Monitoring [FBDDLP07]

Intent. The Alarm Monitoring pattern ensures warnings are sent in case of malicious attacks in a physical system.

Problem. The solution is constrained by the following forces:

- Types. There exist physical and logical alarms.
- Neglect. Alarms can be ignored.
- Logical alarms. There are two states logical alarms can be set to: reset and alarm.

- Physical alarms. There are four states physical alarms can be set to: reset, alarm, cut or short. Hereby, the last two states are caused by either faulty wiring or tampering.
- Logging. Each alarm's state needs to be constantly monitored.
- Update. Interested parties need to be updated when an alarm's state changes.

Solution. This solution implements the alarm entity and uses generalization to further separate the characteristics of a logical from a physical alarm. Additional information is included in the alarm to store the exact time of the event. For regular updates of system changes the Observer pattern is integrated.

5.6.3 Audit Log [LL17, PFS+21]

Intent. The purpose of the Audit Log pattern is to monitor a system and record any significant events with detailed information. These records are then stored as encrypted log files on the server.

Solution. There already exist many different applications that can be used to monitor an IoT system. By default UNIX systems contain a variety of standard log files that are directed to the /var partition. Tools like NetCool can also be used on the network level to aggregate data from different sources such as conventional text log files or UNIX syslog streams.

5.6.4 Authenticated Session [PFS+21]

Intent. The Authenticated Session pattern manages the authenticated online identity of a user when browsing on the internet.

Solution. By utilizing web sessions, the user can access multiple protected pages without having to authenticate themselves more than once. The session keeps track of the current access time and expires after a specified time of inactivity.

5.6.5 Authorization Enforcer [PFS+21]

Intent. The aim of the Authorization Enforcer pattern is to control who is allowed to access specific restricted resources in a system.

Solution. This pattern specifies for each active user which and how different resources can be accessed.

5.6.6 Encrypted Processing [PFS+21]

Intent. The Encrypted Processing pattern ensures that sensitive data stays protected during processing.

Problem. Typically data is decrypted before processing it which makes it vulnerable for tampering by unauthorized users.

Solution. Common approaches to implement this pattern are homomorphic functions and secure multi-party computation. The second one is based on additive secret sharing which involves the distribution of a common secret between the communication partners.

5.6.7 Fault Management [PFS+21]

Intent. The definition of Fault Management is the handling of abnormal operation of a system. Its purpose is to correct the faulty behaviour and restore its original functionality.

Solution. In order to correct any faulty behaviour of a system, specific steps which are also called the “flow of fault management” can be followed:

- Compile alarms.
- Keep customers content by taking quick action.
- Refine and connect alarms.
- Analyse and test the system to identify bugs.
- Discover a method to rectify the issue.
- Confirm that the bug vanished.
- Determine if the current fault management function is efficient.

5.6.8 File Authentication [AA18]

Intent. The File Authentication pattern manages files in a system in which multiple user types have different privileges to access them.

Solution. An approach would be to group different users that want access to an application together and to create a home directory for an authenticated user. Then this directory can be spread over the internet to allow the other users access to the application.

5.6.9 Matrix Authentication [AA18]

Intent. The Matrix Authentication pattern specifies which privileges each user has to access the stored data.

Problem. In order to maintain a robust system, collected data needs to be accessed by multiple different applications.

Solution. By utilizing a two-dimensional table, each user is assigned its own privileges to access the stored sensitive data.

5.6.10 Minefield [KED+06]

Intent. The goal of the Minefield pattern is to confuse the attacker during a break-in by aggressively modifying the standard components of a system in a way that threat detection becomes easier.

Problem. Because attackers have access to the same commercial-off-the-shelf software as developers do, it is possible for them to easily infiltrate the familiar environment and avoid detection, even accelerating the process with already existing scripted tools.

Solution. There are a variety of customizations that the Minefield pattern can take advantage of to modify a common system and trick the attacker. First, it is possible to make scripted attack tools unusable by breaking their compatibility with the system. Second, alarm messages about an intruder can be sent to the user without alerting the attacker itself. Third, any method to make the attacker uncomfortable in the newly modified setting can be enough to fend off a breach.

5.6.11 Remote Authenticator/Authorizer [AA18]

Intent. The Remote Authenticator/Authorizer pattern is utilized to authenticate or authorize resources that are shared remotely over the internet.

Solution. By implementing a single entry point, this pattern is able to transparently control and authorize server access of any remote user.

5.6.12 Role Based Access Control [AA18]

Intent. The aim of Role Based Access Control is to authorize access to resources based on specific user roles in a diverse system where interoperability is crucial.

Solution. This pattern is implemented by categorizing the application resources based on their importance in the system. In an extended version, each modified security policy also is provided with different administration roles.

5.6.13 Safe Processing [PFS+21]

Intent. The Safe Processing pattern ensures that data integrity is guaranteed during and after an application processes sensitive information.

Solution. By implementing the Minefield pattern or integrating any type of integrity check into a system, safe data processing can be made possible.

5.6.14 Secure Distributed Publish/Subscribe [FYW20]

Intent. The Distributed Publish/Subscribe pattern ensures that in an IoT system the publication of events can be securely decoupled from the subscribed users themselves.

Problem. The solution is constrained by the following forces:

- Loose coupling. Subscribers should not be connected to publishers and vice versa.
- Location transparency. The locations of subscribers/publishers are hidden.
- Security degree. Varied protection requirements for different data sensitivity.
- Attack surface. Keep number of vulnerabilities as low as possible.

- Heterogeneity. Diverse IoT devices increase attack surface.
- Identity. Exact identification of participating network devices is needed.
- Compliance. Regulations must be followed which may cause system restrictions.
- Overhead. Keep computations for event distribution low.
- Power consumption. Because of power constraints, IoT devices should be economic.

Solution. The publish/subscribe functionality can be utilized by implementing this pattern. In addition, secure communication channels, access control mechanisms, security logging and digital signatures can be integrated to adapt the system to the required security levels.

5.6.15 Uptime [PFS+21]

Intent. The Uptime pattern can be used to monitor the availability of a specific resource over a given period of time and collect satisfaction or violation evidence in relation to the desired value.

Solution. There are two ways the uptime of a resource can be measured: The first method involves counting the absolute time a system has been running since the initial start. A second approach uses the total running time and divides it by the time the system has been active to obtain the uptime percentage.

5.7 Collaboration & Processes

The main point of the Collaboration & Processes layer in the WFRM architecture is the user interaction with the IoT application that can be utilized to achieve economic value and engagement. In order to ensure a safe environment and monitor who is engaged with the platform, IoT devices can be secured using a Blacklist 5.7.3 combined with a Whitelist 5.7.9 or by implementing Permission Control 5.7.5 mechanisms.

5.7.1 Account Lockout [PFS+21]

Intent. The Account Lockout pattern restricts the user to a limited number of password attempts before the account gets locked.

Problem. Attackers often use dictionaries to guess the password of a user's account.

Solution. This authentication approach can be implemented by following these steps:

- Login screen requires username and password.
- User requests a specific resource by entering its credentials.
- If username is valid, the mediator retrieves account information.
- Mediator evaluates if account is still locked.
- Mediator checks validity of password. If it's not valid, failed login attempts are increased and the last failed login time is set.

- After valid password was entered, resource request is granted and number of failed login attempts is set to 0.

5.7.2 Authentication Enforcer [PFS+21]

Intent. The Authentication Enforcer pattern deals with the question on how to identify an entity and verify that it is who it claims to be.

Solution. When a user requests access to a specific resource, the Authentication Enforcer applies its authentication protocol. After successful authentication, an explicit or implicit Proof of Identity is generated. Therefore, the implementation of the authentication process includes the requirements definition, approach selection and information creation.

5.7.3 Blacklist [RBF+17, PFS+21]

Intent. A Blacklist is a register that lists all identifiers of communication partners that are denied access privileges due to previous abusive behaviour.

Problem. The solution is constrained by the following forces:

- Intervention. Stop already existing abuse.
- Prevention. Prevent future abuse.
- Identification. Reliable identification of abusers.
- Control. Manual adjustment of control mechanisms to define abusive behaviour.
- Simplicity. Easy understanding and usage.
- Efficiency. Normal functionality of IoT devices should not be affected.

Solution. All identifiers of abusers are written on the Blacklist and can be modified via an administrative interface. Any user that is on the Blacklist and wants the privilege to access a resource is immediately blocked. If the communication partner is not on the list, he is allowed to proceed.

5.7.4 History-Based Authentication [CDE+14]

Intent. The purpose of History-Based Authentication is to prevent unauthorized users who disguise as legitimate ones from accessing resources they are not supposed to see.

Problem. The solution is constrained by the following forces:

- Physical token. Tokens can be lost, impaired, taken or duplicated.
- Biometry. Biometric authentication approaches mostly lack maturity and applicability.
- Secure information. Authentication information needs to be hard to steal or replicate.
- Passwords. Regularly changing passwords add security but also mean extra work to create and communicate new authentication information to authorities.
- Complete. A more complete authentication means it is harder to bypass it.

- Proportionality. A task's resources should be proportional to its prevalence and urgency level.
- Disguise. Impersonation of authenticated users must be impossible for authentication authorities.
- Tampering threats. Before giving access to resources, data integrity must be verified.

Solution. The implementation of this pattern requires authentication to be based on what an entity knows, which is in this case its own history. For a successful authentication a set of questions must be answered by the requesting entity. Authenticators must be equipped with decent capabilities, knowledge and authority in order to fulfil their duties. Also by aggregating information about an entity from other depended-on entities, the authenticator's knowledge about a user's history is widened and the authentication process is aided by the collective memory.

5.7.5 Permission Control [RBF+17]

Intent. The Permission Control pattern ensures that device owners can control which functionality or data a server or other communicating entity is able to obtain after their first connection.

Problem. The solution is constrained by the following forces:

- Choice. All data rights are granted by the device owner.
- Granularity. There should be different permission levels.
- Enforcement. Each choice of the user must be strictly followed.
- Simplicity. The user interface must be easy to understand in order to be used correctly.
- Updates. After changes the system has to ensure that only the newest set of choices is enforced.

Solution. For a good Permission Control implementation make sure to ask the user for each device or entity, that is connecting for the first time to the backend server, which information or functionality is allowed to be shared. These choices must be stored on the server and be confirmed once a change in the system occurs.

5.7.6 Personal Zone Hub [RBF+17]

Intent. The Personal Zone Hub is a control centre for device owners to manage permissions and data sharing between multiple gateways and clouds.

Problem. The solution is constrained by the following forces:

- Ownership. The owner of personal data should be the one to grant access.
- Manageability. It may be impractical to perform access control per device.
- Trust. It is difficult to trust third parties with personal data administration.
- Decentralization. In order to spread responsibilities and avoid vendor lock-in, it is better to utilize more than one party for data management.

Solution. The best approach is to create a trusted, permanently addressable Personal Zone Hub as a central control unit for all devices, services, apps and other data. This way the user can share access to parts or all content that is managed by the zone.

5.7.7 Relays [FBDDLP07]

Intent. The Relays pattern handles a system with electronically controlled switches.

Problem. The solution is constrained by the following forces:

- Distinction. There exist two types: door relays and auxiliary relays.
- Relays can be set to two states: on or off.
- Activation period of a relay can be limited or indefinite.

Solution. For a simple implementation a relay entity can be used. Further characteristics can then be added by generalizing the relay entity into door and auxiliary relays respectively.

5.7.8 Single Access Point [PFS+21]

Intent. The Single Access Point is, like the name suggests, the only entry point into a system that grants access to authorized users.

Problem. The more entry points a system has, the easier it is for attackers to cause damage.

Solution. A simple implementation of the Single Access Point involves checking the security policies when a user is requesting entry into the system. This way access control is enforced and the overall system security is increased.

5.7.9 Whitelist [RBF+17]

Intent. The Whitelist is a tool to identify all trusted communication partners of a system and block the privileges of outsiders.

Problem. The solution is constrained by the following forces:

- Prevention. Prevent future abuse.
- Identification. Reliable identification of abusers.
- Control. Manual adjustment of control mechanisms to define abusive behaviour.
- Efficiency. Normal functionality of IoT devices should not be affected.

Solution. All identifiers of communication partners are written on the Whitelist and can be modified via an administrative interface. Any user that is on the Whitelist and wants to access a resource is granted the privilege. If the communication partner is not on the list, the request is denied.

6 Evaluation

Before any of the research questions could be answered, we searched for security patterns that are IoT-specific and already exist in literature. With the help of this newly created pattern catalogue we are now able to answer the previously stated RQs by following these steps:

RQ1 Compare the number of security patterns of each layer in the IoT WFRM.

RQ2 List the important security goals for an IoT system and check for each pattern which goals it protects. Compare the coverage of the different security goals.

RQ3 List the OWASP top ten most common security vulnerabilities within IoT and check if each vulnerability is solved by at least one pattern.

6.1 Data Set of IoT Security Primary Studies

The foundation for the security pattern catalogue in Chapter 5, which was created specifically for this master's thesis, can be found in Table 6.1. This table showcases in chronological order the collected data set of primary studies that are focused on IoT security patterns.

Table 6.1: Overview of primary IoT security pattern studies

Year	**Author**	**Title**	**Paper #**
2021	Fernández et al.	A Pattern for a Secure IoT Thing	[BFAO21]
2021	Papoutsakis et al.	Towards a Collection of Security and Privacy Patterns	[PFS+21]
2020	Fernández et al.	Abstract and IoT security segmentation patterns	[FFYW20]
2020	Fernández et al.	Secure Distributed Publish/Subscribe (P/S) pattern for IoT	[FYW20]
2020	Fernández et al.	A Pattern for a Secure Cloud-Based IoT Architecture	[Fer20]
2020	Muñoz et al.	TPM, a Pattern for an Architecture for Trusted Computing	[MnF20]
2020	Orellana et al.	A Pattern for a Secure Sensor Node	[OFA20]
2019	Moreno et al.	BlockBD: A Security Pattern to Incorporate Blockchain in Big Data Ecosystems	[MFFMS19]

Table 6.1 continued from previous page

2018	Ali et al.	Applying security patterns for authorization of users in IoT based applications	[AA18]
2018	Schuß et al.	IoT Device Security the Hard(Ware) Way	[SID+18]
2018	Seitz et al.	Fogxy: An Architectural Pattern for Fog Computing	[STB18]
2018	Tkaczyk et al.	Cataloging design patterns for internet of things artifact integration	[TWG+18]
2017	Lee et al.	A case study in applying security design patterns for IoT software system	[LL17]
2017	Reinfurt et al.	Internet of Things Security Patterns	[RBF+17]
2016	Sinnhofer et al.	Patterns to Establish a Secure Communication Channel	[SOP+16]
2016	Syed et al.	A Pattern for Fog Computing	[SFI16]
2015	Ur-Rehman et al.	Secure Design Patterns for Security in Smart Metering Systems	[URZ15]
2014	Ciria et al.	The History–Based Authentication pattern	[CDE+14]
2007	Fernández et al.	Security Patterns for Physical Access Control Systems	[FBDDLP07]
2005	Kienzle et al.	Security patterns repository, version 1.0	[KED+06]

6.2 RQ1: IoT World Forum Reference Model Categorization

In order to answer RQ1 *“Which layer in the IoT WFRM is covered by the least security patterns?”*, we went through each paper in our data set in Table 6.1 and assigned each pattern to its corresponding layer in the model. The pattern distribution for each WFRM layer is illustrated with a corresponding pie chart in Figure 6.1.

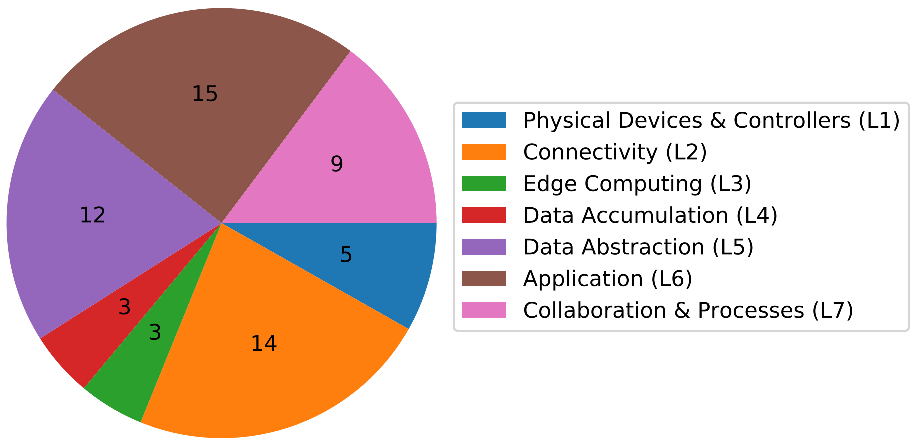

Figure 6.1: The WFRM layer distribution of IoT security patterns

According to the calculated distribution of IoT security patterns that are attributed to the different layers of the WFRM, most patterns can be almost equally found in the Application (L5) and Connectivity (L2) layers. With 15 and 14 patterns respectively these two layers take almost half of the total of 61 IoT security patterns that were found during our search process. The Edge Computing (L3) and Data Accumulation (L4) layers on the other hand are with three patterns each on the lower end of the pattern coverage. When looking at the reasoning for this occurrence, we can say that Edge Computing can be seen as an own technology infrastructure that not all models consider a part of IoT. If we specifically searched for Edge functionality in publications, our success rate in finding such patterns would definitely be significantly higher. But because IoT was the main focus of our research topic, only a few publications that specified IoT as well as Edge technology at the same time could be found. The reason for Data Accumulation (L4) being under-represented with IoT security patterns in our study is definitely worth analysing further. Maybe because a secure storage is a big part of any computer system, mobile or stationary devices, there are only a few patterns that exist to date that specifically try to solve security problems in IoT devices. In conclusion, for a better distribution of patterns in the WFRM architecture the development of IoT security patterns for the Data Accumulation layer is encouraged.

6.3 RQ2: Protected Security Goals

With the second RQ we want to answer the question which security goals are covered by the patterns in the catalogue of Chapter 5 and how the coverage of the rest can be improved. Like for RQ1 we looked at each paper from the data set in Table 6.1 and assigned the specific security goals to each of them. The total number of times each security objective is covered by a pattern is shown in Table 6.2.

Table 6.2: Security objectives addressed by IoT security patterns

Security Objective	**Pattern Count**
Confidentiality	30
Integrity	20
Availability	18
Authentication	27
Authorization	24

With a total of 30 publications the confidentiality objective is the goal that is covered the most in our data set. It is a security requirement that is surely easier to achieve than the others, because any system should protect their sensitive data. Thus, even if an IoT system is built without any security aspects in mind, the probability that confidentiality is ensured is pretty high. Therefore, it is usual that there exist many patterns that guarantee this objective. Availability, however, is the security goal with the least amount of coverage. The objective to ensure that a system is accessible for as long and often as possible can be quite challenging. One other reason for this could be, that when thinking about security, availability is not the most obvious choice compared to confidentiality or integrity. But by keeping the aspect of replication in mind to ensure availability when developing new IoT technologies, IoT architectures will be more secure in the future.

Figure 6.2 compares the coverage of different security objectives in each layer of the WFRM architecture. The outlier in the bar plot is definitely the Data Accumulation (L4) layer. With only three security goals being covered and authentication and authorization being absent entirely, one sees clearly the lack of security solutions for storages of IoT devices. When analysed further one could say, that the reason for authentication being neglected lies in these mechanisms usually being implemented in higher layers of the IoT architecture. Interesting to mention is also the lack of availability support in the Collaboration & Processes (L7) layer. But because this layer is focused on user interactions and not the applications of the system themselves, it makes sense that availability is not a priority here.

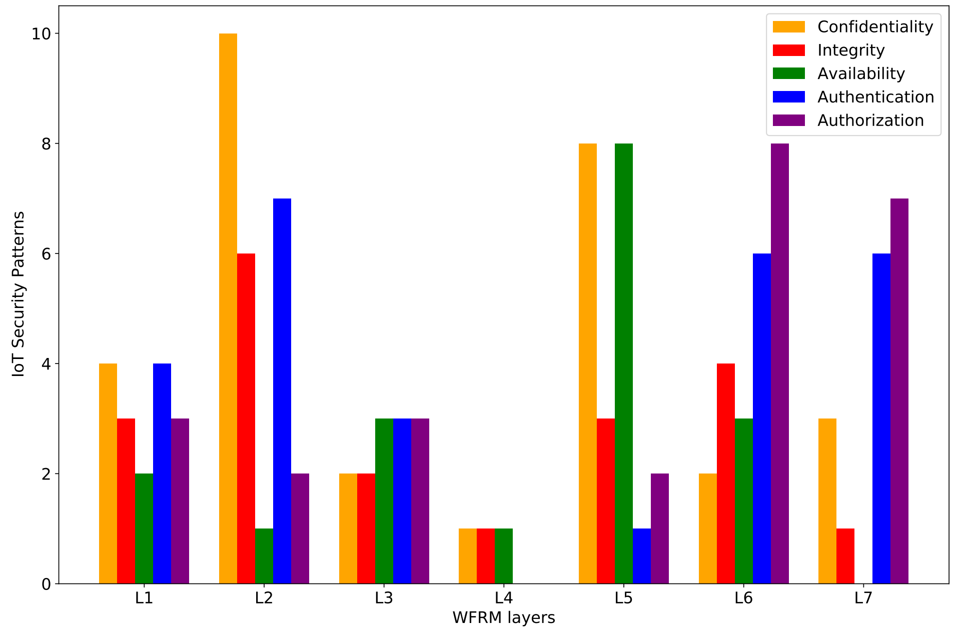

Figure 6.2: IoT security patterns with their security concerns per WFRM layer

6.4 RQ3: Solutions for Common Vulnerabilities

Looking at the OWASP top ten IoT vulnerabilities there are countless of new issues that arise every year while developing IoT systems. In RQ3 we now want to know if any of the security patterns in our pattern catalogue in Chapter 5 are able to solve some of these common mistakes developers tend to make in IoT environments. The following Figure 6.3 shows which common vulnerabilities can be solved with the found IoT security patterns.

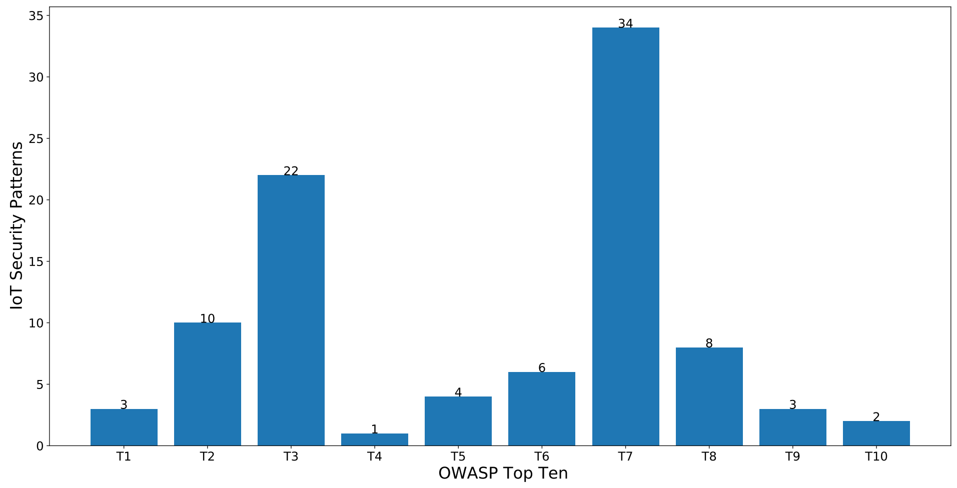

Figure 6.3: Potential pattern solutions for the OWASP common vulnerabilities

It seems that each common vulnerability from the OWASP top ten can be (partially) solved by at least one IoT security pattern from our catalogue. The easiest one to fix is apparently T7 which focuses on insecure data transfer and storage. At least 34 different IoT security patterns that we found have a solution that could remove parts of the vulnerability and improve the handling of data in IoT devices. On the other hand T4 is apparently the hardest one to solve with only one pattern addressing this issue. If we look into its description, the problem is the lack of ability to securely update the IoT device. This is a very specific issue that also is highly dependent on the hardware of the device and its user interface. For a better update management of IoT devices further research in terms of security patterns is highly recommended in this area.

When we now look into the WFRM layer distribution of the IoT security patterns for each individual OWASP vulnerability, the pie charts in Figure 6.5 mirror the results of the previous bar plot. While the pie charts for T2, T3 and T7 (see Figures 6.5(b), 6.5(c), and 6.5(g) respectively) show the most diverse range of pattern solutions from up to six different layers, the one for T4 in Figure 6.5(d) has only patterns from the Physical Devices & Controllers (L1) layer.

🟦 Physical Devices & Controllers (L1)
🟧 Connectivity (L2)
🟩 Edge Computing (L3)
🟥 Data Accumulation (L4)
🟪 Data Abstraction (L5)
🟫 Application (L6)
■ Collaboration & Processes (L7)

Figure 6.4: Legend for pie charts

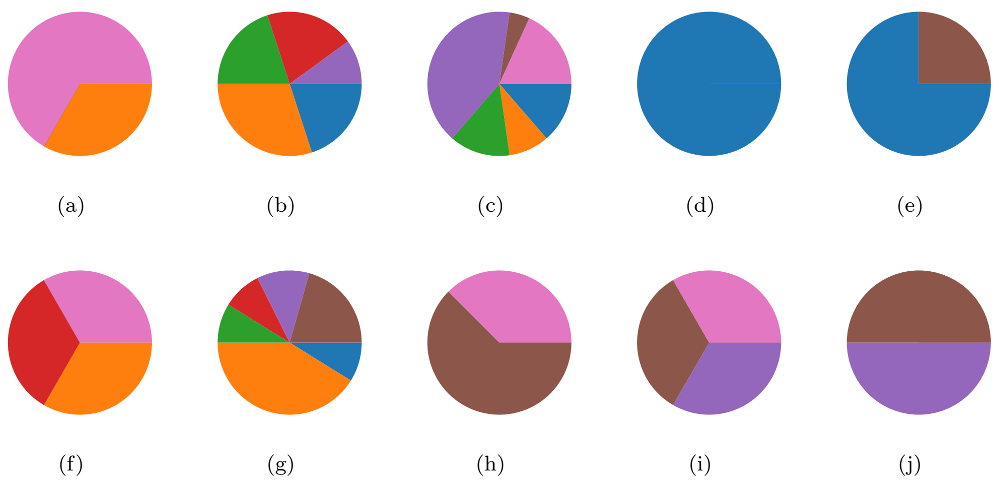

Figure 6.5: Layer distribution of security patterns that address the OWASP top ten with one pie chart per vulnerability: (a) T1; (b) T2; (c) T3; (d) T4; (e) T5; (f) T6; (g) T7; (h) T8; (i) T9; and (j) T10.

Therefore, we see a connection between the found IoT security patterns for a specific OWASP vulnerability and the number of layers the various solutions are from. This assumption is confirmed by the following Figure 6.6, which consists of a scatter plot that showcases the correlation between the pattern quantity and their layer distribution with a value of 0.86808368. This correlation coefficient always ranges from -1 to 1 and indicates the strength of the relationship between two variables. A value between 0.7 and 1 shows a strong connection and the positive sign indicates, that more patterns as solutions for a specific vulnerability also mean a more diverse distribution of WFRM layers for these IoT patterns.

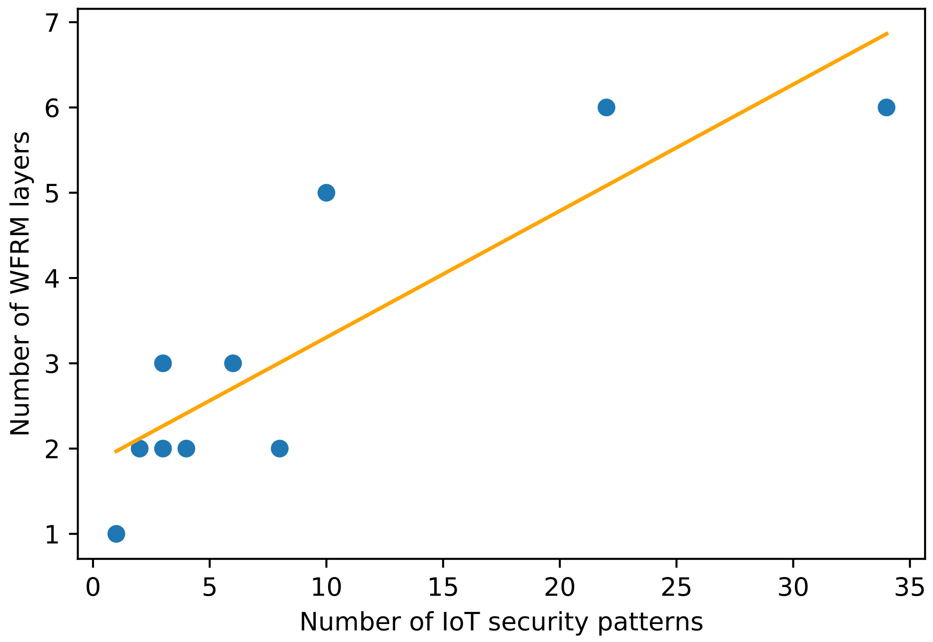

Figure 6.6: The correlation between the number of patterns and WFRM layers for each OWASP vulnerability.

6.5 Use Case Application

Before we discuss the findings of our research work, let's see how the created pattern catalogue can be used in a real life situation. The pattern collection was designed to easily function as a checklist. To showcase the pattern selection process, we describe the following IoT home scenario. It is important to note, that in this example we assume that the system is only used by the owner of the house and no further security measures were taken than the ones that were already integrated into the system.

This smart home contains different connected devices that are distributed in the living room, bedroom, kitchen, bathroom and entrance of the house. All electronic devices run on the custom firmware Tasmota[1] and include: Four RGB LED bulbs[2], six Shelly[3] that function as light controllers, and five smart plugs[4]. In addition, a Raspberry PI with HomeBridge[5] allows the integration of HomeKit into the network that controls the following devices: Two televisions[6], four Sonos ZonePlayers[7], a Ring camera and doorbell[8].

After we established the use case example, we can inspect each pattern of the catalogue in Chapter 5 and check its applicability in the given context. The evaluation results can be found in Table 6.3 with (-) indicating this pattern is not suitable for this system, (o) being used in cases where the given system has already implemented similar security features this pattern would provide, and (+) marking recommended patterns to implement to further optimize the system design.

[1]https://tasmota.github.io/docs, last accessed: 25.02.2023.
[2]https://www.tasmota.shop/gb/lights/7-led-rgb-bulb-wifi-e27-15w.html, last accessed: 25.02.2023.
[3]https://www.shelly.cloud/en-de/products/product-overview/1xs1, last accessed: 25.02.2023.
[4]https://www.amazon.de/dp/B07D73S72W, last accessed: 25.02.2023.
[5]https://homebridge.io, last accessed: 25.02.2023.
[6]https://plugins.hoobs.org/plugin/homebridge-samsung-tizen, last accessed: 25.02.2023.
[7]https://plugins.hoobs.org/plugin/homebridge-zp, last accessed: 25.02.2023.
[8]https://plugins.hoobs.org/plugin/homebridge-ring, last accessed: 25.02.2023.

Table 6.3: Use case applicability of patterns

#	Pattern	Rating	Explanation
5.1.1	Hardware IoT Security	+	Exchangeable cryptographic co-processors to secure IoT devices.
5.1.2	Secure IoT Thing	+	Secure any entity that is connected to sensors/actuators, e.g Raspberry PI.
5.1.3	Secure Sensor Node	-	System does not include sensor nodes.
5.1.4	Security Segmentation	+	IoT devices are divided into subnetworks.
5.1.5	Trusted Platform Module	+	Attestation of Raspberry PI with integrated cryptographic services.
5.2.1	Authenticated Channel	o	Mutual authentication of communication partners and forward secrecy.
5.2.2	Encrypted Channel	o	TLS handshake and exchange of cryptographic information.
5.2.3	Middleware Message Broker	o	HomeBridge controls the flow of messages between IoT devices.
5.2.4	Middleware Self-contained Message	+	Messages should be “pure and complete” representations of events/commands.
5.2.5	Orchestration of SDN Network Elements	-	Only required when an IoT SDN is employed.
5.2.6	Outbound-Only Connection	+	Blocks incoming malicious connection requests.
5.2.7	Password Based Key Exchange	+	Common secret is used to generate session key pairs.
5.2.8	Safe Channel	o	Use certificates to guarantee integrity during message transmission.
5.2.9	Secure Remote Readout	+	Security Module encrypts measurements before transmitting.
5.2.10	Signed Message	o	Use digital signatures during the message generation/exchange process.
5.2.11	Symmetric Key Cryptography	+	Handshake and common secret are exchanged between communication parties.
5.2.12	Third Party Based Authentication	+	Combination of asymmetric cryptography and session keys.
5.2.13	Trusted Communication Partner	+	List trusted communication partners and block unknown connection requests.
5.2.14	Web of Trust	-	Tasmota uses a central self-signed certificate authority.
5.3.1	Fog Computing	-	No cloud-based system.
5.3.2	Fogxy	-	No cloud-based system.
5.3.3	Secure Cloud-based IoT Architecture	-	No cloud-based system.
5.4.1	Encrypted Storage	+	Critical data is encrypted before it gets committed to disk.
5.4.2	Redundant Storage	-	No cloud-based system.
5.4.3	Safe Storage	+	Guarantee integrity of stored data.
5.5.1	Alignment-based Translation Pattern	o	HomeBridge enables interoperability between different platforms.
5.5.2	BlockBD	-	No Big Data system.
5.5.3	Discovery of IoT Services	-	No usage of different IoT services.
5.5.4	Flow-based Service Composition	-	No usage of different IoT services.
5.5.5	IoT Gateway Event Subscription	o	HomeBridge sends notifications on updates.

Table 6.3: Use case applicability of patterns

#	Pattern	Rating	Explanation
5.5.6	IoT SSL Cross-Layer Secure Access	o	Only authenticated entities are able to access the external interfaces.
5.5.7	Middleware Message Translator	o	HomeBridge enables interoperability between different platforms.
5.5.8	Middleware Simple Component	+	Universally applicable pattern to achieve the best component decomposition.
5.5.9	REST Request/Response	o	HomeBridge API is used to connect to different IoT devices.
5.5.10	Server Sandbox	+	Isolate server to protect it in case the system gets compromised.
5.5.11	Service Orchestration	-	No usage of different IoT services.
5.5.12	Translation with Central Ontology	o	HomeBridge enables interoperability between different platforms.
5.6.1	Access Control to Physical Structures	-	No physical structures need to be accessed.
5.6.2	Alarm Monitoring	o	Alarm functionality is included in HomeBridge.
5.6.3	Audit Log	o	HomeBridge has a rolling log screen.
5.6.4	Authenticated Session	-	System runs on a local server with no internet requirements.
5.6.5	Authorization Enforcer	-	Only relevant if system is used by users with different roles.
5.6.6	Encrypted Processing	+	Integrity of data with e.g homomorphic functions or secure multi-party computation.
5.6.7	Fault Management	+	Smart handling of any faulty behaviour of the system.
5.6.8	File Authentication	-	Only relevant if system is used by users with different privileges.
5.6.9	Matrix Authentication	-	Only relevant if system is used by users with different privileges.
5.6.10	Minefield	+	Modify Raspberry PI to confuse attackers and simplify threat detection.
5.6.11	Remote Authenticator/Authorizer	-	System runs on a local server with no internet requirements.
5.6.12	Role Based Access Control	-	Only relevant if system is used by users with different roles.
5.6.13	Safe Processing	+	Guarantee integrity during data processing with e.g integrity checks.
5.6.14	Secure Distributed Publish/Subscribe	o	HomeBridge sends notifications on updates.
5.6.15	Uptime	o	HomeBridge measures and displays the server availability.
5.7.1	Account Lockout	o	Login via password authentication.
5.7.2	Authentication Enforcer	+	Authentication process that creates a proof of identity.
5.7.3	Blacklist	+	Identification of abusers who are not granted access to the system.
5.7.4	History-Based Authentication	+	Authentication is based on the user's own history.
5.7.5	Permission Control	+	User can control which data is shared with the server.
5.7.6	Personal Zone Hub	-	No cloud-based system.
5.7.7	Relays	-	No switches in the system.

Table 6.3: Use case applicability of patterns

#	Pattern	Rating	Explanation
5.7.8	Single Access Point	+	Only one entry point into the system with HomeBridge UI.
5.7.9	Whitelist	+	Identification of trusted communication partners.

While there are many patterns that are not suitable to be implemented in this kind of smart home scenario, like BlockBD 5.5.2 or the Web of Trust 5.2.14, just as many are already integrated into the system, e.g Uptime 5.6.15 or Account Lockout 5.7.1. Nevertheless by going through the catalogue and analysing each pattern individually, we found 25 patterns that can be used to optimize the security measures in this smart home example. Spread across all layers of the WFRM architecture, one can choose from a variety of patterns that include Symmetric Key Cryptography 5.1.1, Server Sandbox 5.5.10 or even simpler solutions like a combination of a Blacklist 5.7.3 and a Whitelist 5.7.9.

This use case demonstrates that our pattern catalogue can act as an easy guideline to improve the security of a system. The evaluation process can be as straightforward as using the catalogue as a checklist while other approaches could include the sorting by WFRM layers or even by security objectives. Regardless of which method one uses, every system has vulnerabilities that can benefit from even the most obvious or trivial security features, like a Single Access Point 5.7.8.

7 Discussion

In this last chapter we point out the threats to validity of our research work (see Section 7.1) as well as suggest areas in the IoT security field that need more investigation in the future in Section 7.2. Lastly a final conclusion to close off this master's thesis is given in Section 7.3.

7.1 Threats to Validity

Like for any type of research work, there exist specific limitations that must be considered when looking at the context of a dissertation. Internal as well as external factors that cannot be influenced by the researcher can lessen the validity of the analysis and its conclusions. These threats also exist in this master's thesis and can be listed as the following points:

Lacking Data Set. The process of automated and manual searching, collecting and scanning publications that fit into the theme of this master's thesis can be quite challenging. We have to decide which patterns have a focus on IoT security and are truly pieces of a solution to a larger problem and not entire architectures by themselves. Therefore the creation of a complete and structured pattern catalogue from scratch is an impossible task and prone to errors and missing patterns.

Indefinite Categorizations. Defining different patterns and putting them into specific categories can be a difficult task. When assigning these patterns to the different layers of the IoT WFRM, we did it with the best of our ability. But these categories can be interpreted in different ways and a pattern can be attributed to more than one layer. So there are many possibilities on where to draw the lines.

Outdated OWASP Top Ten. To have an up-to-date discussion on the topic of IoT security patterns, we tried to take the newest research into account in our analysis. But this is not always possible. In the case of the referenced OWASP top ten list on the common IoT vulnerabilities, the ranking was last updated in 2018. So the possibility that the rankings have changed in the last couple of years is huge because of how fast technology evolves these days. This has to be taken into consideration, when we analyse how these vulnerabilities can be solved with our pattern catalogue and which risks still exist in IoT systems today.

7.2 Future Work

A collection of security patterns in the IoT field is a good start to get an overview of the current state of the art. But there are many more ways in which researchers and developers can advance the secure IoT development and utilize the advantages of standardization.

Pattern Catalogue Expansion. The IoT security pattern catalogue in Chapter 5 cannot be called complete in any way. There are surely more security patterns that can be modified into the IoT context as well as other types of patterns that can make the implementation of secure IoT systems easier. A few examples would be privacy patterns, misuse patterns or anti-patterns. Therefore, the expansion of the security pattern catalogue for IoT is definitely a topic for further research.

Industry Cooperation. Technology and science are ever-evolving, therefore the need for different types of patterns for common problems will always exist and require new and modern solutions. The best way to develop new security patterns, that are optimized for applicability and usage in real-world situations, is a cooperation with the industry. Only when academia combines its theories and ideas with the practical problems of the corresponding industry, we will find the best solutions to solve common issues in the world of IoT.

IoT Security Architectures. The last point would be the usage of existing security patterns to model architectures that are able to protect IoT systems from cyberattacks. By combining specific security patterns we will be able to create improved IoT systems in the future.

7.3 Conclusion

The aim of this analysis was to identify the shortcomings of the state-of-the-art research in the field of IoT security and to provide a guide for future development of secure IoT systems.

In the beginning we defined, which IoT devices belong into the IoT spectrum and in which areas of life these are found. Regardless if you look into the healthcare, banking or industry sector, IoT applications exist everywhere they can be useful. Even a small home can benefit from the safety features of IoT gadgets like motion detectors, security cameras or smart locks.

After gaining a better understanding of the topic, we further inspected various threats in the IoT field that can be caused by cyberattacks and harm the safety of its users. Attackers can infiltrate a system by using techniques like man-in-the-middle, denial of service or even ransomware. In addition, we looked at a list of IoT vulnerabilities, such as unsecure passwords or system components, and how to overcome these threats by utilizing secure-by-design architecture.

After introducing the IoT WFRM and an overview of the most important security objectives, our goal was to create a comprehensive catalogue of IoT security patterns. Although one can never say such a collection is complete, the catalogue includes all findings we could make under the given search criteria. Furthermore, these patterns allowed us to gain knowledge about the following questions:

- Which layer in the IoT WFRM is covered by the least security patterns?
- Which security goal is addressed by security patterns the least and how can the coverage be improved?

- How many of the OWASP top ten most common IoT vulnerabilities are solved by utilizing these security patterns?

With a catalogue that consists of 61 patterns, it was surprising that almost half of these belonged to only two layers. Also, the lack of security goals that were covered in the Data Accumulation layer is alarming. On the other hand every vulnerability, that was published by OWASP, was addressed by at least one pattern, which is a positive discovery.

To sum up, with these findings we built a foundation for future endeavours to broaden the spectrum of IoT security patterns. Thus, this pattern catalogue can be used as a starting point for secure system design or further academic research, that will expand the current number of entries in our pattern collection. With appropriate industry cooperation the development of applicable patterns, that improve the security of IoT systems and at the same time protect its users, can be achieved.

List of Figures

4.1 Overview of the search and selection procedure 11

6.1 The WFRM layer distribution of IoT security patterns 42
6.2 IoT security patterns with their security concerns per WFRM layer 44
6.3 Potential pattern solutions for the OWASP common vulnerabilities 44
6.4 Legend for pie charts 45
6.5 Layer distribution of pattern solutions for OWASP top ten 45
6.6 The correlation between the number of patterns and WFRM layers for each OWASP vulnerability. 46

List of Tables

2.1 Internet of Things Top Ten (2018) taken from OWASP Wiki[0] 4

4.1 Summary of IoT architecture layers[0] 13

5.1 Legend for pattern lookup table 15
5.2 IoT security pattern lookup table 16
5.2 IoT security pattern lookup table 17
5.2 IoT security pattern lookup table 18

6.1 Overview of primary IoT security pattern studies 41
6.2 Security objectives addressed by IoT security patterns 43
6.3 Use case applicability of patterns 47
6.3 Use case applicability of patterns 48
6.3 Use case applicability of patterns 49

Bibliography

[AA18] Ishfaq Ali and Muhammad Asif. Applying security patterns for authorization of users in iot based applications. In *2018 International Conference on Engineering and Emerging Technologies (ICEET)*, pages 1–5, Feb 2018.

[BFAO21] Eduardo B. Fernández, Hernan Astudillo, and Cristian Orellana. A pattern for a secure iot thing. In *26th European Conference on Pattern Languages of Programs*, EuroPLoP'21, New York, NY, USA, 2021. Association for Computing Machinery.

[BMR$^+$96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture - Volume 1: A System of Patterns.* Wiley Publishing, 1996.

[Bun14] Michaela Bunke. On the description of software security patterns. In *Proceedings of the 19th European Conference on Pattern Languages of Programs*, EuroPLoP '14, New York, NY, USA, 2014. Association for Computing Machinery.

[CDE$^+$14] José Carlos Ciria, Eladio Domínguez, Inés Escario, Ángel Francés, María Jesús Lapeña, and María Antonia Zapata. The ¡i¿history-based authentication¡/i¿ pattern. In *Proceedings of the 19th European Conference on Pattern Languages of Programs*, EuroPLoP '14, New York, NY, USA, 2014. Association for Computing Machinery.

[FBDDLP07] Eduardo B. Fernandez, Jose Ballesteros, Ana C. Desouza-Doucet, and Maria M. Larrondo-Petrie. Security patterns for physical access control systems. In Steve Barker and Gail-Joon Ahn, editors, *Data and Applications Security XXI*, volume 4602, pages 259–274, 07 2007.

[Fer20] Eduardo B. Fernández. A pattern for a secure cloud-based iot architecture. In *Proceedings of the 27th Conference on Pattern Languages of Programs*, PLoP '20, USA, 2020. The Hillside Group.

[FFYW20] Eduardo Fernández, E Fernandez, Nobukazu Yoshioka, and Hironori Washizaki. Abstract and iot security segmentation patterns. 01 2020.

[FSP$^+$19] Konstantinos Fysarakis, George Spanoudakis, Nikolaos Petroulakis, Othonas Soultatos, Arne Bröring, and Tobias Marktscheffel. Architectural patterns for secure iot orchestrations. In *2019 Global IoT Summit (GIoTS)*, pages 1–6, 2019.

[FYW09] Eduardo B. Fernández, Nobukazu Yoshioka, and Hironori Washizaki. Modeling misuse patterns. In *Proceedings - International Conference on Availability, Reliability and Security, ARES 2009*, pages 566–571, oct 2009.

[FYW20] Eduardo Fernández, Nobukazu Yoshioka, and Hironori Washizaki. Secure distributed publish/subscribe (p/s) pattern for iot. 02 2020.

[GHJV94] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education, 1994.

[KC07] Barbara Ann Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 07 2007.

[KED$^+$06] Darrell M. Kienzle, Matthew C. Elder, Ph. D, Ph. D, David Tyree, and James Edwards-hewitt. Security patterns repository, version 1.0, 2006.

[LL17] Wen-Tin Lee and Po-Jen Law. A case study in applying security design patterns for iot software system. In *2017 International Conference on Applied System Innovation (ICASI)*, pages 1162–1165, May 2017.

[MFFMS19] Julio Moreno, Eduardo B. Fernandez, Eduardo Fernandez-Medina, and Manuel A. Serrano. Blockbd: A security pattern to incorporate blockchain in big data ecosystems. In *Proceedings of the 24th European Conference on Pattern Languages of Programs*, EuroPLop '19, New York, NY, USA, 2019. Association for Computing Machinery.

[MM18] Henry Muccini and Mahyar Tourchi Moghaddam. Iot architectural styles. In Carlos E. Cuesta, David Garlan, and Jennifer Pérez, editors, *Software Architecture*, pages 68–85, Cham, 2018. Springer International Publishing.

[MnF20] Antonio Muñoz and Eduardo B. Fernandez. Tpm, a pattern for an architecture for trusted computing. In *Proceedings of the European Conference on Pattern Languages of Programs 2020*, EuroPLoP '20, New York, NY, USA, 2020. Association for Computing Machinery.

[OFA20] Cristian Orellana, Eduardo B. Fernandez, and Hernán Astudillo. A pattern for a secure sensor node. In *Proceedings of the 27th Conference on Pattern Languages of Programs*, PLoP '20, USA, 2020. The Hillside Group.

[PFS$^+$21] Manos Papoutsakis, Konstantinos Fysarakis, George Spanoudakis, Sotiris Ioannidis, and Konstantina Koloutsou. Towards a collection of security and privacy patterns. *Applied Sciences*, 11(4), 2021.

[RBF$^+$17] Lukas Reinfurt, Uwe Breitenbücher, Michael Falkenthal, Paul Fremantle, and Frank Leymann. Internet of things security patterns. In *Proceedings of the 24th Conference on Pattern Languages of Programs*, PLoP '17, USA, 2017. The Hillside Group.

[RFBL17] Lukas Reinfurt, Michael Falkenthal, Uwe Breitenbücher, and Frank Leymann. Applying iot patterns to smart factory systems. In *Proceedings of the 11th Advanced Summer School on Service Oriented Computing*, pages 1–10. IBM Research Division, 2017.

[RNF20a] Tanusan Rajmohan., Phu Nguyen., and Nicolas Ferry. A systematic mapping of patterns and architectures for iot security. In *Proceedings of the 5th Inter-*

national Conference on Internet of Things, Big Data and Security - IoTBDS,, pages 138–149. INSTICC, SciTePress, 2020.

[RNF20b] Tanusan Rajmohan, Phu H. Nguyen, and Nicolas Ferry. Research landscape of patterns and architectures for iot security: A systematic review. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 463–470, 2020.

[RNF22] Tanusan Rajmohan, Phu Hong Nguyen, and Nicolas Ferry. A decade of research on patterns and architectures for iot security. *Cybersecurity*, 5:1–29, 2022.

[SFI16] Madiha H. Syed, Eduardo B. Fernandez, and Mohammad Ilyas. A pattern for fog computing. In *Proceedings of the 10th Travelling Conference on Pattern Languages of Programs*, VikingPLoP '16, New York, NY, USA, 2016. Association for Computing Machinery.

[SID+18] Markus Schuß, Johannes Iber, Jürgen Dobaj, Christian Kreiner, Carlo Alberto Boano, and Kay Römer. Iot device security the hard(ware) way. In *Proceedings of the 23rd European Conference on Pattern Languages of Programs*, EuroPLoP '18, New York, NY, USA, 2018. Association for Computing Machinery.

[SOP+16] Andreas Daniel Sinnhofer, Felix Jonathan Oppermann, Klaus Potzmader, Clemens Orthacker, Christian Steger, and Christian Kreiner. Patterns to establish a secure communication channel. In *Proceedings of the 21st European Conference on Pattern Languages of Programs*, EuroPlop '16, New York, NY, USA, 2016. Association for Computing Machinery.

[STB18] Andreas Seitz, Felix Thiele, and Bernd Bruegge. Fogxy: An architectural pattern for fog computing. In *Proceedings of the 23rd European Conference on Pattern Languages of Programs*, EuroPLoP '18, New York, NY, USA, 2018. Association for Computing Machinery.

[TWG+18] Rafal Tkaczyk, Katarzyna Wasielewska, Maria Ganzha, Marcin Paprzycki, Wieslaw Pawlowski, Pawel Szmeja, and Giancarlo Fortino. Cataloging design patterns for internet of things artifact integration. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2018.

[URZ15] Obaid Ur-Rehman and Natasa Zivic. Secure design patterns for security in smart metering systems. In *2015 IEEE European Modelling Symposium (EMS)*, pages 278–283, 2015.

[WOH+20] Hironori Washizaki, Shinpei Ogata, Atsuo Hazeyama, Takao Okubo, Eduardo B. Fernandez, and Nobukazu Yoshioka. Landscape of architecture and design patterns for iot systems. *IEEE Internet of Things Journal*, 7(10):10091–10101, 2020.

[WP13] Claes Wohlin and Rafael Prikladnicki. Systematic literature reviews in software engineering. *Information and Software Technology*, 55(6):919–920, 2013.

[WYH+19] Hironori Washizaki, Nobukazu Yoshioka, Atsuo Hazeyama, Takehisa Kato, Haruhiko Kaiya, Shinpei Ogata, Takao Okubo, and Eduardo B. Fernandez. Landscape of iot patterns. In *2019 IEEE/ACM 1st International Workshop on Software Engineering Research Practices for the Internet of Things (SERP4IoT)*, pages 57–60, 2019.