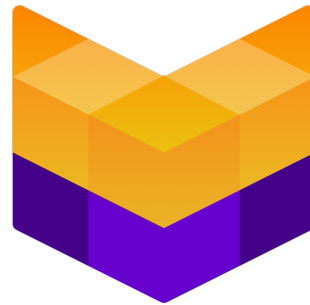


Espen Gudmundsen
Sigrid Marita Kvamme

Data Protection Fortification

A Data-centric Threat Modeling Method for
Development Teams to Assess the Risk of Data
Tampering and Support Secure Handling of
External Data Sources

Master's thesis in Computer Science
Supervisor: Daniela Soares Cruzes
Co-supervisor: Tosin Daniel Oyetoyan
June 2022



Espen Gudmundsen
Sigrid Marita Kvamme

Data Protection Fortification

A Data-centric Threat Modeling Method for
Development Teams to Assess the Risk of Data
Tampering and Support Secure Handling of External
Data Sources

Master's thesis in Computer Science
Supervisor: Daniela Soares Cruzes
Co-supervisor: Tosin Daniel Oyetoyan
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

Systems rely on data for purposes such as situational awareness, environmental monitoring, energy monitoring, and critical industrial monitoring. These systems often rely on sensory data generated by Internet of Things (IoT) devices, which have many security challenges due to their low-resource constraints, heterogeneity, and deployment in hostile environments. Systems consuming this data must therefore be designed with security measures to detect and prevent damage from data tampering attacks. However, there has been a lack of research on methods that can aid practitioners in understanding risk in data.

In a pre-study to this thesis, we performed a systematic literature review on security testing of IoT. In this work, we found a lack of focus on security testing in early phases of development and that monitoring data is a potential way to detect attacks. However, first knowing how and what to monitor is essential. In this master thesis, we develop a method for practitioners to reflect on the risk in data and improve the security of handling data from IoT. We also investigate the potential for a data monitoring system to detect data tampering. We employ a framework for carrying out design science research (DSR), using interviews and focus groups as our data generation methods.

The main contribution of this study is a data-centric threat modeling method named Data Protection Fortification (DPF) that development teams can use during planning to assess the security risk of a data source. This research further provides insight into practices reported by companies for handling data securely. We also contribute to research with a conceptual architecture for a data monitoring system for detecting attempts at tampering with data.

DPF was developed and tested in collaboration with two companies in the industry, one student organization, and two security experts. Results show that participants have a positive attitude towards using DPF and that it has the potential to become a communication tool for security between developers and stakeholders. Furthermore, the method proved useful in other contexts beyond data with origin in IoT devices.

Sammendrag

Datasystemer avhenger av data for å kunne bidra til en bedre situasjonsforståelse, som tillater overvåking av ulike faktorer innen miljø, energi og i industrielle produksjonsmiljøer. Slike datasystemer avhenger ofte av sensoriske data generert av enheter i tingenes internett (IoT). Disse enhetene har mange sikkerhetsutfordringer knyttet til begrensede ressurser, produktulikheter, og deres fysiske lokasjon som gjør de sårbare for fysisk skade fra både vær og mennesker. Datasystemer som benytter seg av datakilder fra IoT må derfor inneha sikkerhetsmekanismer som kan oppdage og stoppe trusler som kan komme med dataene. Til tross for dette, finnes det lite forskning på metoder som kan bidra til å vurdere risikoen ved bruk av en datakilde.

I en forstudie til denne masteroppgaven utførte vi et systematisk litteratursøk på sikkerhetstesting av IoT. Her avdekket vi at fokuset på sikkerhetstesting i tidlige utviklingsfaser er mangelfull, og at overvåking av data er en potensiell nytenkende måte å oppdage angrep på. Det er imidlertid essensielt å først finne ut hvordan data kan overvåkes, og hvilke data man bør rette fokus mot. I denne masteroppgaven utvikler vi en metode som kan brukes for å reflektere over risikoen i data og forbedre sikkerheten ved håndtering av data fra IoT. Vi undersøker også om det finnes potensiale for å oppdage data som har blitt tuklet med ved å benytte et dataovervåkingssystem. Vi bruker et rammeverk for å gjennomføre design science research (DSR), og samler inn data gjennom å holde intervjuer og fokusgrupper.

Hovedbidraget fra denne masteroppgaven er en datasentrisk metode for trusselmodellering kalt Data Protection Fortification (DPF). Denne metoden kan bli brukt av utviklerteam for å vurdere sikkerhetsrisikoen i en datakilde og komme opp med relevante sikkerhetsmekanismer, som et ledd i utformingen av et sikkert design. I denne studien bidrar vi også med innsikt i praksiser for sikker datahåndtering funnet i ulike bedrifter. Vi bidrar også til forskning med en konseptuell arkitektur for et dataovervåkingssystem for å oppdage forsøk på tukling med data.

DPF ble utviklet og testet i samarbeid med to bedrifter i industrien, en student organisasjon, og to sikkerhetseksperter. Resultatene viser at deltakerene har en positiv innstilling til bruk av metoden, og at den har potensiale for å bli et verktøy for kommunikasjon om sikkerhet mellom utviklere og ikke-utviklere i et utviklingsprosjekt. Metoden har vist seg å være relevant også for andre datakilder utover data fra IoT.

Acknowledgement

We would like to thank our supervisor, Professor Daniela Soares Cruzes, and our co-supervisor, Associate Professor Tosin Daniel Oyetoyan, for their fantastic guidance and support throughout this year. Without them, this master thesis would not be possible.

We extend our gratitude to Thor-André Aresvik, who gave us the opportunity to work with Equinor and provided us with valuable support throughout the project period. We would like to express our appreciation to Vidar, Terje, and Bjørnar at Equinor for the support and the good discussions. If not for them, this thesis would have ended up very differently. We would also like to thank the rest of the teams participating in our study, who offered to share insight into how they worked with us. We are thankful for valuable feedback from the two anonymous security experts who evaluated our method.

This work would not be possible without support from our loved ones at home and friends who provided understanding and mental support throughout the project. Finally, thanks to our fellow students, Andreas Rimolsrønning and Ola Plassen, for all the precious interruptions and late-night work sessions.

The good memories and experiences from this time will forever stay with us.

Contents

Abstract	iii
Sammendrag	v
Acknowledgement	vii
Contents	ix
Figures	xiii
Tables	xv
Acronyms	xvii
1 Introduction	1
2 Background and Related Work	3
2.1 Vulnerability and Risk Management	3
2.1.1 Bug	3
2.1.2 Vulnerability	3
2.1.3 Risk	4
2.1.4 Threat	6
2.2 Security Activities in Development Phases	6
2.3 Threat Modeling	6
2.3.1 Step 1: Model System	7
2.3.2 Step 2: Find Threats	8
2.3.3 Step 3: Address Threats	8
2.3.4 Step 4: Validate	9
2.3.5 STRIDE	9
2.3.6 Asset-Oriented Threat Modeling	10
2.3.7 Threat Modeling in Agile Software Development Projects . .	11
2.3.8 Evaluating Threat Modeling Using Flow	12
2.4 Protection Poker	13
2.4.1 Calculating Risk	13
2.4.2 Performing Protection Poker	14
2.5 Security Testing	16
2.5.1 Security Testing of Data-intensive Systems	17
2.6 Data Monitoring and Anomaly Detection	19
2.6.1 Detecting Dynamic Threats Using Audit Hooks	19
2.6.2 Audit Hooks in Industry	20
2.7 Identified Gap in Research	21
3 Research Methodology	23

3.1	Research Objective	23
3.2	Research Questions	23
3.3	Context and Participants	24
3.3.1	Equinor	24
3.3.2	Webkom	25
3.3.3	Ryde	26
3.4	Approach for Research	26
3.5	Method for Thesis Development	27
3.5.1	Explicate Problem	29
3.5.2	Define Requirements	30
3.5.3	Design and Develop Artefact	30
3.5.4	Demonstrate Artefact	32
3.5.5	Evaluate Artefact	32
3.6	Data Collection Methods	33
3.7	Data Analysis	34
3.8	Research Paradigm and Bias	35
3.9	Ethics	36
3.10	Data Storing	36
3.10.1	Security Measures	36
4	Design of Data Protection Fortification	37
4.1	Preparations for Data Protection Fortification	39
4.1.1	When to Perform Data Protection Fortification	39
4.1.2	Select Data Source	39
4.1.3	Prepare Data Fields	39
4.1.4	Participants	39
4.1.5	Tool Recommendations	40
4.2	How to Perform Data Protection Fortification	41
4.2.1	Discuss Data Source	42
4.2.2	Identify Data Fields	42
4.2.3	Prioritize Data Fields	42
4.2.4	Plot Fields on Risk Matrix	44
4.2.5	Evaluate Security Measures	45
4.2.6	Outcome	49
4.2.7	Evaluate and Review	50
4.3	Data Protection Fortification in Practice	50
4.4	Data Protection Fortification in Practice - Alternative	55
4.5	Discussion Questions for Data Protection Fortification	60
4.5.1	General Questions About the Data Source	60
4.5.2	Security Implications for the Data Source	62
4.5.3	Evolution of Questions	64
5	Results	67
5.1	RQ1: What Are the Practices Reported by Companies for Securely Handling Data Coming From Devices or Other Sources?	67
5.1.1	Data Access	69

5.1.2	Data Validity and Monitoring	70
5.1.3	Data Velocity	72
5.1.4	Data Origin	72
5.1.5	Data Processing	72
5.1.6	Data Schema	73
5.1.7	Data Usage	73
5.1.8	Fault Mitigations	74
5.2	RQ2: How Can Data-centric Threat Modeling Support Teams in Identifying Security Risks and Promote Secure Design in Handling Data?	74
5.2.1	TAM Evaluation of DPF	75
5.2.2	Evaluation of the Activities in DPF	75
5.2.3	Verbal Evaluation of DPF	75
5.2.4	Observations and Lessons Learned from DPF	78
5.2.5	Evaluating DPF with Security Experts	80
5.3	RQ3: How Can Data Monitoring Using Audit Hooks Identify Attacks in Data With Increased Risk?	82
5.3.1	Audit Hook	84
5.3.2	Analysis Capabilities	85
5.3.3	Ingest	86
5.3.4	Monitoring Dashboard	86
6	Discussion	87
6.1	Implications to Research	87
6.2	Implications to Practice	93
6.2.1	Recommended Practices for Securely Handling Data	94
6.3	Threats to Validity and Limitations	96
6.4	Lessons Learned	97
7	Conclusion	99
	Bibliography	101
A	DPF Cheat Sheet	107
B	DPF Preparation Meeting Guide	113
C	DPF Data Extraction Form	115
D	NSD Notification Form	121
E	On Security Testing of IoT Systems: A Systematic Literature Review	127

Figures

2.1	Classic risk matrix	5
2.2	Extending the risk matrix	5
3.1	Design Science Research method framework	27
3.2	Design Science Research: Thesis Development	28
4.1	Overview of the flow of using Data Protection Fortification	37
4.2	Steps in Data Protection Fortification	41
4.3	Risk matrix used in DPF	44
4.4	Method for security measures evaluation	46
4.5	Calibrating using Mentimeter	52
4.6	Estimating using Mentimeter	53
4.7	A risk matrix as shown in Mentimeter.	54
4.8	Evaluating security measures in Miro	55
4.9	Estimation table and risk matrix drawn on a white-board	56
4.10	Answering sheet for participants during estimation	57
4.11	Example of answers in DPF with only physical tools	59
5.1	Reported practices for handling a connection URL	70
5.2	Reported practices for validation and monitoring	71
5.3	Reported practices for use of data schema	73
5.4	TAM evaluation results	76
5.5	Evaluation of most useful part in DPF	76
5.6	Example of requirement from DPF taken into to development	83
5.7	Monitoring system architecture using audit hooks	84

Tables

2.1	Development phases and related security activities	6
2.2	Description of STRIDE and examples of mitigations	10
2.3	Security properties and their relation to STRIDE	17
2.4	Approaches for security testing of IoT systems	18
2.5	Findings from our pre-study and their relevance to this thesis	19
3.1	TAM inspired questions used to evaluate DPF	34
4.1	Desired properties of DPF	38
4.2	Security measures for the <i>String</i> data type	47
4.3	Security measures for the <i>Number</i> data type	48
4.4	Security measures for the <i>Date</i> data type	49
4.5	Security measures for handling the data source	49
4.6	Discussion questions for the first step in DPF	65
5.1	Practices reported by the development teams	68
5.2	Overview of DPF sessions held	74
5.3	Observations and lessons learned from DPF	79
6.1	Ability to experience flow in DPF	87
6.2	How DPF address challenges reported in literature	89

Acronyms

API Application Programming Interface. 78, 81

AR action research. 27, 29

CAR canonical action research. 27

CVE Common Vulnerabilities and Exposures. 4

DFD Data Flow Diagram. 89

DPF Data Protection Fortification. iii, v, xiii, xv, 1, 2, 24, 31–44, 46, 49–51, 55, 57, 58, 60, 65, 67, 74–82, 85, 87–94, 96–100

DSL domain-specific language. 89, 91

DSR design science research. iii, v, 23, 26–29, 31

GDPR General Data Protection Regulation. 26, 50, 81

ICS Industrial Control Systems. 81

IoT Internet of Things. iii, v, 1, 2, 17, 19–21, 23, 24, 26, 29, 38, 44, 60, 62, 64, 72, 73, 81, 91, 95, 96, 99, 100

IS information systems. 27

NIST National Institute of Standards and Technology. 16

SaaS Software as a Service. 20

SAST Static Application Security Testing. 6, 16, 30

SDK Software Development Kit. 20

SDLC software development life cycle. 6, 7

SLR systematic literature review. 1, 7, 17, 19, 29, 30, 99

SSDLC secure software development life cycle. 1

Chapter 1

Introduction

Data has become a critical asset for decision-making in different domains such as smart grids, smart city and smart ocean [1–3], where data from Internet of Things (IoT) systems is becoming increasingly prevalent [4]. However, ensuring that the data has not been tampered with by an adversary that is aiming to influence decisions is difficult, as IoT comes with several challenges. IoT devices have limited resources and are restricted to light-weight encryption mechanisms and authentication algorithms. Furthermore, they are typically deployed in unattended environments where an attacker can get physical access and take over the device [5–7], and potentially make it send maliciously altered data [5, 7, 8] or even injection attacks through the data. Therefore, systems consuming such data must be designed with security measures to detect and handle such threats. However, the focus of security professionals has traditionally been on securing functions and systems, not data [9], resulting in a lack of methods that can aid practitioners in understanding risk in data.

With the goal of uncovering approaches for development teams to start doing security testing of IoT systems, we wrote a systematic literature review (SLR) on state-of-the-art security testing approaches for Internet of Things (IoT) [10]. From this, we discovered a need for more research on approaches applicable during earlier phases of development, especially during planning and design. Among the approaches found, one approach in particular showed promise for bringing focus on the data used in IoT systems and the importance of not assuming trust in such data. The approach describes a data monitoring system for detecting threats in data, using a data-capturing concept referred to as audit hooks [11].

In this thesis, we build on the work of our SLR, together with findings in recent literature on threat modeling, to develop a threat modeling process focusing on data from IoT, that is applicable during the design phase as part of a development team’s secure software development life cycle. The method, which we have named Data Protection Fortification (DPF), was developed in tandem with performing field testing in the industry. The goal of DPF was initially to help determine what IoT data should be monitored for threats using audit hooks. However, after several iterations of development, it has expanded to include elicitation of

other security measures on data. The results from testing show that DPF is helpful for development teams to identify higher-risk data fields and shows potential for usage in other contexts beyond data with origin in IoT systems.

The research questions for this thesis are the following:

- **RQ1** What are the practices reported by companies for securely handling data coming from devices or other sources?
- **RQ2** How can data-centric threat modeling support teams in identifying security risks and promote secure design in handling data?
- **RQ3** How can data monitoring using audit hooks identify attacks in data with increased risk?

The contributions of this thesis are two-fold. In the first part, we develop a method for development teams to (1) share knowledge about a data source and discuss possible security implications, (2) collaborate on identifying higher-risk data fields, and (3) evaluate security measures that can be implemented to reduce the risk of data tampering. In the second part, we use our DPF method to identify practices for data handling reported by the development teams and come up with possible requirements for a data monitoring system extending the work of Shrestha and Hale [11] for detecting possible tampering of data. We answer this part with the data collected from performing Data Protection Fortification with various development teams.

The thesis is organized into seven chapters. Chapter 1 has introduced the problem and the motivation for this thesis. Chapter 2 aims to explain relevant theoretical concepts and related work which has influenced this thesis. Chapter 3 dives into the research activities and the method for conducting it, also explaining the research context and problem statements of this thesis. Chapters 4 and 5 presents the result, namely the DPF method and the synthesis of data collected during the DPF sessions with various teams. Chapter 6 discusses the results' implications for practice and research. We also discuss lessons learned from developing the DPF method. Finally, Chapter 7 concludes the thesis and also presents directions for future work.

Chapter 2

Background and Related Work

In this chapter, we present theory and related work in fields relevant to this thesis. First, we introduce key terms within the fields of vulnerability and risk management. We then outline the different software development phases and relevant security activities found within these. We introduce the method of threat modeling and STRIDE, as well as related work on threat modeling. Following this, we describe Protection Poker, on which we initially base our proposed threat-modeling method. Then we give security properties and their relation to security testing. Relevant findings from the pre-study on security testing on data-intensive systems are presented and followed up with literature relevant to the proposed data monitoring system. Finally, we summarize the identified gap in research that guides the work in this thesis.

2.1 Vulnerability and Risk Management

In this section, we define some common terms and concepts in vulnerability and risk management.

2.1.1 Bug

A bug is a synonym for a fault [12]. A fault may be introduced in a system's specification or code, while a bug is usually associated with a fault introduced in the code during development. Both have the potential to cause an error or failure in the system. Hence, faults are considered a cause, and system failures are an effect resulting from activating or executing a fault. Landwehr *et al.* [13] characterizes an inadvertently introduced security flaw in a program as a bug, violating the systems security requirements.

2.1.2 Vulnerability

According to NIST [14], a vulnerability is "a weakness in the computational logic (e.g., code) found in software and hardware components that, when exploited,

results in a negative impact to confidentiality, integrity, or availability". In simpler terms, a vulnerability is an error that an attacker can exploit [15] and may be introduced at any time during the software development process. To reduce the risk of unintentionally introducing previously known vulnerabilities, MITRE corporation has for over 20 years disclosed information about known vulnerabilities to the public through its Common Vulnerabilities and Exposures (CVE) program¹.

2.1.3 Risk

Within the domain of computer security, NIST defines *risk* as a "measure of the likelihood and the consequence of events or acts that could cause a system compromise, including the unauthorized disclosure, destruction, removal, modification, or interruption of system assets" [16]. Ni *et al.* [17] give the following characteristics to a risk:

- A risk is harmful.
- A risk is uncertain; we may not know if it will occur or not.
- A risk is abrupt and not anticipated.
- A risk can not be completely eliminated, only controlled, mitigated, transferred, or evaded.

Using the NIST definition of risk, Equation (2.1) shows how risk can be calculated if the likelihood of it occurring and the consequence is known. The resulting "risk number" is often used to assess risks.

$$\text{Risk} = \text{Likelihood} \times \text{Consequence} \quad (2.1)$$

Risk assessment is a structured process that is a key step in most risk management frameworks [17]. It involves processes for identification, classification, estimation, and prioritization of risks. Many tools and techniques have been created to aid practitioners in performing and documenting risk assessments across various domains. One such tool, the Risk Matrix Approach (RMA), is a widely adopted tool for conducting risk assessments. Ni *et al.* [17] discusses several limitations to RMA, such as the typical four-square pattern found in many RMA variants. One such variant, the "4-T strategy", is shown in Figure 2.1, where risks are categorized into four groups based on their risk value (X, Y). For risks that fall close to or at the border of one "category", it is unclear what action should be taken. To remedy this, Ni *et al.* [17] propose drawing hyperbolas into the matrix as shown in Figure 2.2 and then coloring the areas between these to indicate the threshold categories.

¹<https://cve.mitre.org/>

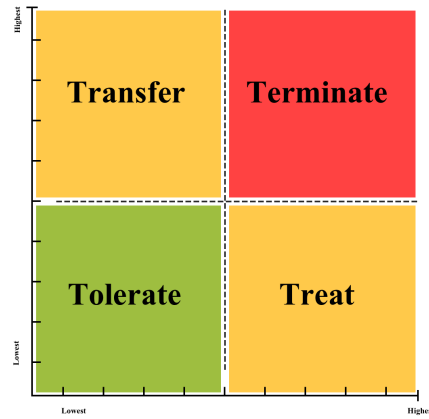
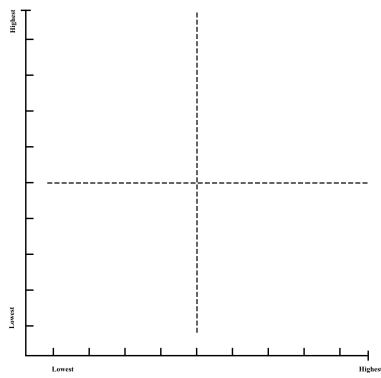
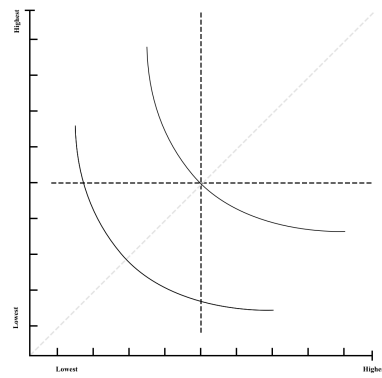


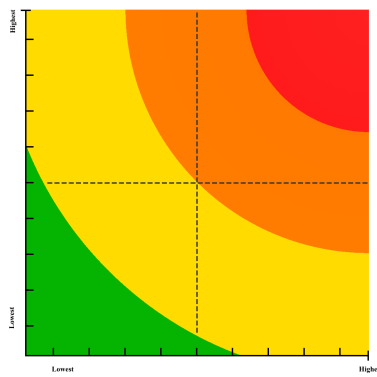
Figure 2.1: Classic risk matrix using the 4-T (Tolerate, Treat, Transfer, Terminate) strategy to categorize risks.



(a) Traditional risk matrix



(b) Adding hyperbolas to (a)



(c) Adaptation of (b)

Figure 2.2: Adding hyperbolas to a risk matrix as recommended by Ni *et al.* [17].

2.1.4 Threat

NIST [18] defines a threat as "any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service". A threat is also defined as "the potential for a threat-source to successfully exploit a particular information system vulnerability" [18].

2.2 Security Activities in Development Phases

Bachmann and Brucker [19] describe the security activities and security testing methods that practitioners can apply in the different phases of development: (1) during planning and design, (2) during application development, (3) when executable in a test environment, and (4) during system operation and maintenance. Table 2.1 shows examples of such activities and techniques mapped to the development phases. The techniques used in (3) can also be used in (4) to check if the system is still secure, in addition to passive security testing techniques such as monitoring the behavior of the system or analyzing system logs [19].

Table 2.1: Development phases mapped to security activities and security testing techniques.

Development phase	Example security activities
During Planning and Design	Threat modeling, architecture security reviews
During Application Development	SAST, manual code reviews, static binary code analysis
Executable in a Test Environment	Penetration testing, vulnerability scanners, fuzz testing
System Operation and Maintenance	Monitoring system, intrusion detection system

2.3 Threat Modeling

Threat modeling is recognized as an essential activity in software security. The activity aims to elicit security requirements based on existing knowledge about the system, typically by using abstractions (e.g., models, functional requirements) to help think about possible risks and threats [20, 21].

Threat modeling is beneficial to perform for both small and large complex systems, both those already in production or existing only on paper [20]. Since taking action on new security requirements is more costly when a system is already in production, it is recommended to perform threat modeling in the early phases of development. At the start of a project, threat modeling can help fortify security in the overall system architecture. During a project, integrating a more focused threat modeling as part of the organization's software development life

cycle (SDLC), using new requirements or functionality as the scope, can help identify security requirements for these.

As the activity has grown in popularity and adoption, many techniques have come to exist to aid in performing threat modeling, such as STRIDE covered in Section 2.3.5. Numerous processes on how to perform threat modeling also exist in the literature. They can be categorized by their area of focus and level of granularity as software-centric, asset-centric, attack-centric, and data-centric [22–24].

Tuma *et al.* [22] did a systematic literature review (SLR) where 26 threat analysis approaches for secure software design were reviewed, and this study was later extended with 22 approaches by Håkonsen and Ahmadi [23] who also added a focus on agile.

Main findings from Håkonsen and Ahmadi [23] which are relevant to this thesis include: (1) the majority of methodologies were not adapted to agile development and had no clear definition of done, (2) generally poor descriptions on how to integrate the techniques into the workflow of practitioners, and (3) there is generally a lack of documentation on how to perform the threat modeling except for the publications. The authors call for research on improving all of these aspects. The authors also found that techniques that can support practitioners by providing support in terms of examples and templates seemed promising for an agile context and for ease of adoption, where these approaches were also often lightweight and had the potential for collaboration. Furthermore, only a minority of the approaches were useful to developers, and none focused on data.

Regardless of which threat modeling process one chooses to follow, we echo Myagmar *et al.* [20] in that a threat model can not simply be created based on ideation and brainstorming sessions. There is a need for a structured process to identify risks systematically.

In the following section, we explain one structured process inspired by OWASP's guide on application threat modeling [25], and Shostack [21]. The output from each step aims to answer the four questions summarized in Shostack's four-step framework for threat modeling [21], as shown below:

1. **Model System** - *What are you building?*
2. **Find Threats** - *What can go wrong?*
3. **Address Threats** - *What should you do about the things that can go wrong?*
4. **Validate** - *Did you do a decent job of analysis?*

2.3.1 Step 1: Model System

This step involves modeling the system we are building and generating documentation about the system's intended use. OWASP [25] has compiled a list of what we typically want to document in this step. This includes:

- Creating use cases to understand how the application is used.
- Identifying entry points to see where a potential attacker could interact with the application.

- Identifying assets, i.e., items or areas in which an attacker would be interested.
- Identifying trust levels representing the access rights the application will grant to external entities.

2.3.2 Step 2: Find Threats

Different techniques exist to help in discovering possible threats to the system. One such technique, STRIDE (covered in more detail in Section 2.3.5), is a structured process that systematically helps one go through different threat categories. STRIDE can be applied to the use-cases identified and documented in the previous step and help keep discussions focused. Other techniques for finding threats include generating "attack/threat trees" and "misuse cases". Both are typically documented in diagrams and show how attackers can target specific functions to attain their goals and how the system provides countermeasures to defend against the attacks (or highlighting a lack of such). Regardless of which technique is used to discover threats, all threats to the system should be documented (e.g., enumerated in a list) so that we can address them in the next step.

2.3.3 Step 3: Address Threats

In short, we can either mitigate, eliminate, accept or transfer a threat. A short description of each is given below:

- **Mitigate:** Propose security measures that will make it harder for an attacker to take advantage of a threat. For instance, use HTTPS instead of HTTP to make it less likely that someone will be able to tamper or intercept client-server communication.
- **Eliminate:** Remove the threat by also removing the entry point which enables the threat. This typically means removing the functionality associated with the threat.
- **Transfer:** Transfer the responsibility of handling a threat by moving the risk to someone else. For instance, letting an external product handle sensitive information, such as user authentication or payment procedures on behalf of the system.
- **Accept:** This just means to accept that the threat exists and that there are possible risks associated with this threat.

Mitigation should be the "go-to" action to address a threat, as it is the easiest and best for your customers [21] (although mitigating threats can be hard work).

Ranking threats

In OWASP [25], threat analysis is first done by ranking the threats. A general approach is to look at the risk factors for a threat. To help us in this work, we

can use a risk assessment model such as DREAD to rank threats by assigning a numeric "risk score" to them.

Providing Threat Mitigations

In this step, we try to devise countermeasures or mitigations to defend against the threat. For this, we can use checklists that include common mitigation strategies for threats categorized using STRIDE. Of course, not all threats can be mitigated with these strategies, so developing specific mitigations may require both security expertise and domain knowledge about the system one is building. We have included a checklist of common mitigation techniques categorized using STRIDE in Table 2.2.

Documenting Mitigations and Countermeasures

After ranking the threats and providing possible mitigations, it is important to begin working on implementing the actual mitigations. Shostack [21] recommends documenting the identified threats as bugs, thereby ensuring that they are tracked and prioritized. Also, documenting relevant test cases to verify that the threat has been mitigated is recommended. It is important that threat modeling is not just an exercise done on paper [21], and by doing this, we create actionable tasks for the development team to handle.

2.3.4 Step 4: Validate

The final step of threat modeling is to review the overall process. It is recommended to assess each step in the order they were performed, beginning with the model and continuing with the identification and addressing of threats. To help in this, Shostack [21] propose a checklist for common things to look for in each step.

2.3.5 STRIDE

STRIDE is a mnemonic created by Microsoft for six major categories of threats that can be used in threat modeling. It stands for spoofing of user identity (S), tampering with data (T), repudiation (R), information disclosure (I), denial of service (D), and elevation of privilege (E) [21, 26, 27]. Table 2.2 describes each of these threat categories and gives some examples of common mitigations². The mitigation techniques for tampering given by OWASP and Shostack [21] are focused on preventing tampering of data from happening in the first place (e.g., of files, network packets), and as a following, the mitigation techniques include measures like using hashes, digital signatures, and tamper-resistant protocols. However, within the scope of this thesis, we are more concerned with protecting the system when

²https://owasp.org/www-community/Threat_Modeling_Process

we assume the data could have been tampered with to include false or malicious inputs. Therefore we instead show example mitigations here for data validation.

For practitioners wishing to apply STRIDE to their systems, OWASP provides a reference sheet³.

Table 2.2: Description of STRIDE with examples of mitigations [21].

Attack category	Description	Mitigations
Spoofing	Pretending to be someone you are not.	Appropriate authentication using strong protocols, where credentials and authentication tokens are encrypted during transit and storage.
Tampering	Modifying something you are not supposed to modify.	Proper validation of data (e.g., data type, format, white-listing, semantic) and having no security decision based upon parameters that can be manipulated.
Repudiation	Being able to claim that you did not do something.	Use digital signatures, leave audit trails for key events and sensitive operations, and protect your logs.
Information Disclosure	Exposing information to people who are not authorized to see it.	Use strong authorization for accessing resources, encrypt the data, and transport it using privacy-enhanced protocols.
Denial of Service	Attacks that prevent a system from providing a service.	Appropriate authentication and authorization, filter requests, throttle suspicious hosts and maintain quality of service with elastic resources.
Elevation of Privilege	When a user or process can do things they should not be able to do.	Run processes with the least privileges, separate data and code, validate data, and check for vulnerable third-party dependencies.

2.3.6 Asset-Oriented Threat Modeling

During threat modeling, activities for threat enumeration and asset identification are often carried out in collaborative brainstorming sessions. Without guidance or formalized processes to drive these activities, the outcome of these activities is highly dependent on the knowledge or creativity of the participants. Messe *et al.* [28] propose a structured asset identification process to be used within the asset identification phase of threat modeling. A reference model for assets was created to help practitioners reason about the assets. The structured process aims to facilitate collaborative threat enumeration and asset identification activities.

³https://owasp.org/www-pdf-archive/STRIDE_Reference_Sheets.pdf

Definition of an asset Existing definitions of what an asset is are too vague to be used in the context of asset-oriented threat modeling. Instead, Messe *et al.* [28] propose three derivatives of an asset in their asset reference model. Domain Asset (DA) is a type of asset that has a business value for the business stakeholders. DA appear as system artifacts in system architecture models. Assets that have value for security experts, and have threats associated with them, are considered Vulnerable Assets (VA). VA appears as system artifacts in attack pattern descriptions. Both DA and VA may have assets that exist solely within each type, for instance, a DA that is not vulnerable to an attack. The third type of asset, a Vulnerable Domain Asset (VDA), is a DA which is also identified as a VA, which should have controls implemented to mitigate possible attack surfaces. During asset identification, VDA's are something an attacker wants to attack that has value for the business stakeholders and are discovered when a VA matches a DA.

For this thesis, the definitions of assets could also apply to fields in a data source to identify which fields have a business value and the possibility of containing vulnerabilities. The reference model could be used to identify and discuss the data source's fields. The fields considered both to be "vulnerable" and to have "domain value" could be prioritized when evaluating security measures for the data source.

2.3.7 Threat Modeling in Agile Software Development Projects

Bernsmed *et al.* [29] study how agile development teams have adopted threat modeling as part of their development process. Four development teams with different levels of security expertise and developer competence were studied, and challenges, good practices, and experiences were collected from each team. The most difficult challenge reported is integrating threat modeling in the existing development process and changing developers' mindsets into focusing on security. The recommendations for good practices and challenges from Bernsmed *et al.* [29] listed below can help improve future threat modeling methods.

Challenges Developers lack of motivation. It is difficult to make developers attend threat modeling sessions because they feel valuable development time is lost. Not knowing what to do with the outcome of threat modeling is also reported as an issue. One observation was that development teams do threat modeling for compliance reasons, not because they see its potential value.

Best Practices Including different roles and stakeholders as participants in threat modeling sessions is recommended. Regular time intervals for the sessions and using checklists as part of making a clearly defined process also help guide the sessions.

Recommendations Create lists of the following items that can be used during the threat modeling session:

- Assets involving infrastructure personnel (databases, servers, cloud infrastructure).
- Secrets, such as cryptographic keys. Also, specify how these are used in the system.
- Services and systems that send or receive data from the product.

For this thesis, the challenges and best practices reported are useful to guide the development of a method for assessing threats that come in through the data. Particularly for generating tangible outcomes from the activity that is useful to developers (and not only for compliance purposes), providing checklists to guide the process, which is also recommended by Myagmar *et al.* [20].

2.3.8 Evaluating Threat Modeling Using Flow

Flow is a widely accepted model to capture what makes an experience enjoyable [30]. Shostack [21] claims that flow is the most important test of a methodology for threat modeling. Without experiencing flow, practitioners may feel bored or anxious and become less efficient in finding threats. In turn, this makes them less likely to adopt the method [21]. Evaluation using flow has also been adopted in other disciplines, such as Gameflow [31], which is used for evaluating player enjoyment in games. The elements that contributes to flow are given below (from Shostack [21]):

- **The activity is intrinsically rewarding** Doing the activity because doing it feels enjoyable, not because you get rewarded for doing it.
- **People become absorbed in the activity** Effortless involvement in the activity, with little interruptions or waiting involved [31].
- **A loss of the feeling of self-consciousness** When in a flow experience, we become less aware of the information we use to represent to ourselves who we are [30].
- **Distorted sense of time** The experience of losing track of time when performing an activity, or feeling that time stands still.
- **A sense of personal control over the situation or activity** Exercising control of a difficult situation to overcome a challenge or task at hand [30].
- **Clear goals** The task or activity must have a clear, attainable goal for the person to be able to concentrate on achieving it. Shostack [21] mention that many approaches to threat modeling fail to include clear goals. One such example is "find all possible security problems", of which the possibility for achievement is debatable. Hence, goals should have attainable criteria for how they can be achieved.
- **Concentrating and focusing** The task at hand requires attention from the person so that only information relevant to the task is allowed into awareness. The more concentration required by the task at hand, the more absorbing the task will be [31].
- **Direct and immediate feedback** Feedback provides the possibility to assess the current progress on a task, and also to let a person know if they have

done sufficient work on the task (e.g., give feedback if the threat model is in a "good" state).

- **Balance between ability level and challenge** Tasks should match the skill level of the person performing it. If a person does not have the right skills to perform an activity, the activity is rendered meaningless [30].

We will use the elements in flow to evaluate the method we develop, as recommended by Shostack [21].

2.4 Protection Poker

The Protection Poker game is a security activity for agile development teams proposed by Williams *et al.* [32]. The goal is to enable teams to perform a security risk assessment for the following product increment. It is an informal form of misuse-case development and threat modeling based on the participants' discussions. The tangible output from the activity is an estimated security risk "score" for each requirement or functionality included in the following product increment that can help guide the prioritization of required security efforts. The activity stems from another collaborative estimation activity, Planning Poker, which many development teams use to estimate the relative size of the requirements included in the following product increment.

Protection Poker should be performed during the start of each new iteration. Ideally, the extended team should be present (developers, project managers, product owners, testers, and security experts) so that the discussions during the activity can benefit from the diversity of knowledge and perceptions. SINTEF [33] suggests that to keep the game focused, one person should be a moderator. Someone should also be assigned the responsibility to note down any interesting points from the discussions during the activity, such as security problems and vulnerabilities, likely attacks, and possible ways to mitigate the attacks.

2.4.1 Calculating Risk

The *security risk* of a requirement is calculated by the equation in Equation (2.2). The *ease points* reflects how the new, enhanced or corrected functionality described by a requirement, affects the *ease of attack* of the system. The *value points* reflects the *value of the asset that could be exploited with a successful attack*.

$$\text{Security Risk} = \text{Ease Points} \times \text{Value Points} \quad (2.2)$$

The equation reflects the hypothesis that the probability of a successful attack increases with the value of the asset available through the functionality and the ease of attack of the functionality. The more attractive an asset is to an attacker, the more time an attacker will be willing to spend to devise an attack; thus, the risk increases.

There are different variants of which values are available to the team when voting on the estimates. Williams *et al.* [32] suggests using nine possible values

when voting on the estimates (1, 2, 3, 5, 8, 13, 20, 40, 100), while another variant by SINTEF [33] suggests using ten values that are more evenly spread out (<10, 20, 30, 40, 50, 60, 70, 80, 90, 100).

Calibration To get the best results, Williams *et al.* [32] recommends that the team calibrates the ease- and value-points at the start of a project. SINTEF [33] suggests calibrating before running Protection Poker for a new system since the numbers that determine asset value or system exposure (i.e., ease of attack) can vary between different development projects.

The calibration involves assigning the lowest and the highest value to the requirements that are the most difficult to attack and the easiest to attack, respectively. Since the value of a particular asset (e.g., database table) does not change based on the different requirements that use that asset, a relative value can be determined for a given system so that the team can reuse these value estimates.

The calibration is important to prevent the team from falling for the temptation of giving everything a high value in high-risk projects [33], since "when everything is very valuable, then nothing is very valuable" [34]. If there is little spread in the risk estimations, it would be tough to make prioritizations within the project [33]. The calibration may change over time as the system changes along with the iterations [34].

2.4.2 Performing Protection Poker

In the following sections, we briefly describe the process of performing the Protection Poker, as described by Williams *et al.* [32, 34] and SINTEF [33]. We have structured the activity into three steps: the opening discussion, risk estimation, and security requirements elicitation.

Opening Discussions

The activity starts with selecting the requirements that will most likely be included in the following work iteration. Someone with expert knowledge about the requirements then explains the requirements to the team. The team is then encouraged to think and discuss their opinions on different security aspects, such as possible threat actors, attack surfaces, the potential for misuse, and the value of the assets handled. Examples of questions proposed by Williams *et al.* [32] when discussing the security implications include:

- *Who would want to attack the system?*
- *What could an attacker do if they got a hold of the data stores that this new requirement accesses - and stole, deleted, or corrupted the data?*
- *What damage could an insider do through this functionality, particularly if he or she could bypass the user interface?"*

When the discussion quiets down, the team moves on to estimating risk values for each requirement.

In Williams *et al.* [34], it is also suggested to create a checklist of security issues to consider in this discussion, possibly extending these example questions. This brings a possibility for both knowledge sharing and discovering hidden assumptions within the team. As stated by Williams *et al.* [32], "Protection Poker shows promise to improve not only software security but also the entire development team's security knowledge."

Risk Estimation

For each requirement, the team votes on ease points and value points using a specialized card deck to calculate the security risk according to Equation (2.2). The points are given relative to the calibrated endpoints and the values for the previously assessed requirements. The team can vote on either ease points or value points first.

The ease points and value points are estimated iteratively for each requirement. The general procedure is as follows: 1) the team discusses, 2) the team votes, showing their answers simultaneously, 3) the ones with the highest and lowest cards start a new discussion by explaining their choices to the group, 4) a revote is done if needed. The steps repeat until there is consensus in the team about each number estimate [33, 34].

Williams *et al.* [32] mention that misunderstandings and new perspectives and perceptions can often be revealed while discussing differences of opinion, and that it may take two or three votes before the team reaches a consensus. If the team cannot reach a consensus on their own, the moderator can suggest a value [33].

The *value points* for a requirement is the sum of all of the estimated values of the assets that the new requirement protects or uses [32]. The estimation discussion starts by having everyone collaborate on identifying all the assets that are created or touched by the requirement under consideration [33]. If any assets do not already have a value point estimate from a prior calibration, the team first assigns these through discussion, and voting [32]. SINTEF [33] suggests that the discussions revolve around the asset's value for various actors (e.g., customers, users, the business, attackers) and the consequences if the asset is compromised. When all the assets have values, the sum is noted down as the value points of the requirement.

When estimating *ease points*, the team first discusses the change in ease of attack of a system related to a new requirement. SINTEF [33] suggests discussing how the requirement influences how the system can be attacked and the ease of performing these attacks. In addition, consider what level of access the attacker can get (e.g., full access, read-only) and if they can affect the availability of any assets. Williams *et al.* [32] mention that these discussions might lead the team to revise a requirement on the spot (also updating the documentation) to reduce the ease-point estimate. The final estimate from the vote is noted down as the ease points of the requirement.

For Protection Poker to be an effective technique for estimating security risks, it

is necessary to have a culture where diverse opinions are valued. No one should be coerced into agreeing with the estimates of team members who are held in higher regard since disagreements and misunderstandings often indicates a security risk [34].

After the value points and ease points have been estimated for all of the requirements, the security risks can be calculated according to Equation (2.2) and put into a table.

Security Requirements Elicitation

The team can use the list of calculated risks to prioritize their security efforts when implementing the requirements. Decisions on how to address the security for each requirement should be documented. If any specific security activities or functionalities are required, they should be included where other requirements are documented (e.g., in the backlog) [33]. Security measures might include measures such as developing more formal misuse cases or threat models, doing security inspections, redesigning the architecture to be more secure, using static analysis tools (SAST), or doing security testing [32].

2.5 Security Testing

Security testing should be carried out regularly to protect systems and test their resilience to attacks from adversaries. NIST defines security testing as "testing that attempts to verify that an implementation protects data and maintains functionality as intended" ⁴. Felderer *et al.* [26] further describes security testing as testing that verifies and validates software system requirements related to security properties by identifying if the properties are correctly implemented for a given set of assets. Table 2.3 gives a description of each security property as defined by NIST [35], and show how each security property relate to STRIDE.

⁴https://csrc.nist.gov/glossary/term/security_testing

Table 2.3: Security properties and their relation to STRIDE [27, 35].

Property	Description	STRIDE
Confidentiality	Assurance that information is not disclosed to unauthorized persons, processes, or devices. It covers data in storage, during processing, and in transit.	Information Disclosure
Integrity	The property that data has not been accidentally or maliciously altered or destroyed since it was created, transmitted or stored. It includes ensuring information non-repudiation and authenticity.	Tampering with Data
Availability	Ensuring timely and reliable access to and use of information services and data for authorized users.	Denial of Service
Authenticity	The property of being genuine and being able to be verified and trusted, and confidence in the validity of a transmission, a message, or message originator. Verifying the identity of a user, process, or device.	Spoofing Identity
Authorization	The process of verifying that a requested action or service is approved for a specific user, process, or device. Providing the correct access privileges.	Elevation of Privilege
Non-Repudiation	Protection against an individual who falsely denies having performed a certain action, and provides the capability to determine whether an individual took a certain action, such as creating information, sending a message, approving information, or receiving a message.	Repudiation

2.5.1 Security Testing of Data-intensive Systems

In the specialization project that preceded this thesis, we wrote a systematic literature review (SLR) on state-of-the-art security testing approaches for IoT systems, where 1335 papers were filtered, and 21 papers were included [10]. Table 2.4 shows an overview of the identified approaches. For each approach, we reviewed the type of security testing reported, at what point in the development process they could be used, what skills were needed, and how the approach output was reported. We also reviewed their possibilities for automation, how easy they were to adopt, and how they would fit in an agile setting.

As IoT is data-intensive, we expected to find a focus on data among the approaches. However, approaches considering data were mainly concerned with setting up test beds for analyzing network traffic data [6, 36] to discover potential security flaws. One approach for data monitoring showed promise in detecting attacks coming through the data (see Section 2.6.1).

Table 2.4: Approaches for security testing of IoT systems included in Kvamme and Gudmundsen [10]. The paper used as inspiration for developing the data monitoring system is highlighted with an asterisk (*).

Code	Title	Authors	Approach for security testing
P01	A Dynamic Analysis Security Testing Infrastructure for Internet of Things	Wang <i>et al.</i> [36]	DAST using network capturing methods
P02	Attack Surface Modeling and Assessment for Penetration Testing of IoT System Designs	Mahmoodi <i>et al.</i> [37]	Penetration testing and attack-surface modeling
P03	Automated and on-demand cybersecurity certification	Karagiannis <i>et al.</i> [38]	Automated security audits to certify components
P04	AVRS: Emulating AVR microcontrollers for reverse engineering and security testing	Pucher <i>et al.</i> [39]	Fuzzing of AVR firmware
P05	Bug detection in embedded environments by fuzzing and symbolic execution	Vijtiuk <i>et al.</i> [40]	Symbolic execution and fuzzing of software
*P06	Detecting Dynamic Security Threats in Multi-Component IoT Systems	Shrestha <i>et al.</i> [11]	Multi-component threat analysis using audit hooks
P07	EcoFuzz: Adaptive energy-saving greybox fuzzing as a variant of the adversarial multi-armed bandit	Yue <i>et al.</i> [41]	Coverage-based gray-box fuzzing
P08	Finding Sands in the Eyes: Vulnerabilities Discovery in IoT with EUFuzzer on Human Machine Interface	Jiaping <i>et al.</i> [42]	Black-box mutation-based fuzzing
P09	FIRM-AFL: High-Throughput Greybox Fuzzing of IoT Firmware via Augmented Process Emulation	Zheng <i>et al.</i> [43]	Coverage-based gray-box fuzzing
P10	Hybrid Firmware Analysis for Known Mobile and IoT Security Vulnerabilities	Sun <i>et al.</i> [44]	Binary code similarity analysis
P11	Integrating Threat Modeling and Automated Test Case Generation into Industrialized Software Security Testing	Marksteiner <i>et al.</i> [45]	Automated security testing through risk-analysis-enriched threat modeling
P12	IoT Testbed Security: Smart Socket and Smart Thermostat	Bettayeb <i>et al.</i> [46]	Easy-to-setup testbed for security assessment
P13	Iotverif: Automatic Verification of SSL/TLS Certificate for IoT Applications	Liu <i>et al.</i> [47]	Constructing model from runtime communication and checking it for vulnerabilities
P14	Mirage: towards a Metasploit-like framework for IoT	Cayre <i>et al.</i> [48]	Framework for security audits and penetration testing
P15	P2IM: Scalable and hardware-independent firmware testing via automatic peripheral interface modeling	Feng <i>et al.</i> [49]	Fuzzing with randomized input of microcontroller firmware
P16	Penetration Testing of Intrusion Detection and Prevention System in Low-Performance Embedded IoT Device	Zitta <i>et al.</i> [50]	Penetration testing
P17	Requirements and Recommendations for IoT/IIoT Models to automate Security Assurance through Threat modeling, Security Analysis and Penetration Testing	Ankele <i>et al.</i> [51]	Using metadata from common diagrams and models to automate security assurance
P18	Security Testbed for Internet-of-Things Devices	Siboni <i>et al.</i> [6]	Generic testbed performing standard and advanced security tests
P19	Techniques to Improve Reliability in an IoT Architecture Framework for Intelligent Products	Coman <i>et al.</i> [52]	Securing code and device at many layers
P20	Threat Analysis for Wearable Health Devices and Environment Monitoring Internet of Things Integration System	Tseng <i>et al.</i> [53]	Threat modeling and penetration testing
P21	Z-Fuzzer: Device-agnostic fuzzing of Zigbee protocol implementation	Ren <i>et al.</i> [54]	Improved grammar-based fuzzing through coverage heuristics

Table 2.5 shows the most relevant findings from this SLR. We use these to identify requirements for a method to do data-centric threat modeling that can be used during the development of IoT systems.

Table 2.5: Findings from Kvamme and Gudmundsen [10] and their relevance to this thesis.

Aspect	Finding	Relevance to thesis
Application	Very few approaches were applicable in early phases of development	The method should be possible to perform in early development phases.
Skills needed	Developers could perform 50% of the approaches, but only 24% had an output they were likely to understand. In contrast, 86% could be performed and understood by security-trained developers	The method should be performed by developers, and its result should be usable by developers. The method should help share security knowledge among the participants.
Output	More than half of the approaches (57%) do not specify the output from their approach	The method should generate clear and actionable output.
Automation	62% of the approaches could be partly automated, and 29% could be highly automated.	The steps in the method and the output should have the potential for automation.
Agile	None of the approaches mentioned how they fit into an agile workflow	The method should help the team inspect security efforts and encourage close collaboration on embedding security into the product.

2.6 Data Monitoring and Anomaly Detection

This section introduces the related work on data monitoring using a concept called "audit hooks" [11], discovered our SLR on security testing for IoT [10]. We also describe the results of searching in grey literature to find data monitoring similar to the one presented by Shrestha and Hale [11] in commercially available monitoring tools.

2.6.1 Detecting Dynamic Threats Using Audit Hooks

Shrestha and Hale [11] presents a technique for performing dynamic security analysis in an IoT environment using data captured from multiple sources and perspectives (web traffic, Bluetooth, application data) collected through three types of "audit hooks". An *audit hook* is a mechanism placed in code to capture data, such as a function parameter, enabling real-time access to data often only available within a process or a function. The data captured from the audit hooks are usually passed to a (centralized) auditor service [11], which integrates the data to enable

security analysis. The authors recommended adding audit hooks to methods related to critical or sensitive data assets. Risk assessments or design documents can help determine where to place the audit hooks.

The authors implemented audit hooks in a previously developed IoT testbed and successfully identified attacks by analyzing the data captured from the audit hooks. Four multi-class classification algorithms (Support Vector Machine (SVM), logistic regression, Naïve Bayes, and K-NN) were evaluated for training a model on identifying an attack based on captured data. The SVM algorithm resulted in the best average precision and recall of attacks, with an average precision of no less than 86%.

Relevant to this thesis is the general idea of monitoring the system through the use of audit hooks placed in code. We will extend this work by developing a monitoring system using audit hooks in code to transmit data received from an external source to analysis. The monitoring system could include the attack detection algorithm in its analysis capabilities.

2.6.2 Audit Hooks in Industry

An extensive effort was made to search for commercial software products similar to the audit hooks proposed by Shrestha and Hale [11]. Though many Software as a Service (SaaS) products exist for monitoring log data, such as Panther⁵, few introduce concepts similar to "audit hooks" implemented in code for sending data to the monitoring system while the data is still in the execution environment.

DataDog⁶, a cloud-based SaaS product primarily aimed at consuming telemetry data (such as application metrics, logs, and trace-data), comes close with their implementation of an agent running in parallel with the system under monitoring. The running application communicates with the DataDog service through the DataDog agent. An SDK is provided in multiple programming languages, enabling communication from within the execution environment (i.e., inside the application code). The type of data expected to be sent and monitored in DataDog is performance- or usage-related statistical data, where the data is generated by the system (e.g., a metric for current CPU usage of the running application or an increment of a metric). The anomaly detection feature available in DataDog can monitor statistical data using different configurations, such as abnormal user fluctuation and application failures, and then alerting this to the system maintainers.

In contrast to the type of data expected in DataDog, we are in this thesis interested in data that comes from external sources (e.g., human user input, sensor readings). We expect this data to have a lower level of trust and could contain mistakes or threats. Even though the data type is generally different, some of the algorithms used in DataDog could still be relevant. While the anomaly detection feature is not meant to detect vulnerabilities from within the data (e.g., malicious input) or if a data source has been tampered with, the algorithms used for moni-

⁵<https://www.panther.com/>

⁶<https://www.datadoghq.com/>

toring can still be relevant for discovering data tampering over time. The "robust" algorithm⁷ supported by DataDog monitoring seems promising for discovering data tampering where there are subtle level shifts in the data.

2.7 Identified Gap in Research

None of the threat modeling approaches found by Håkonsen and Ahmadi [23] focused on data. For data-intensive systems, data-centric threat modeling may provide a novel perspective for security. Also, there is a need for research to document how their threat modeling approaches can be adopted in practice, as well as create methods that are usable by developers [23].

The granularity level used for assets in traditional threat modeling is usually on whole databases, or tables [24]. For this thesis, we focus on a lower granularity level by focusing on the data source itself and, more specifically, the data fields within the data source. Focusing on data itself poses different implications for security, such as how the data is transported into the systems and the origin of the values generated in each data attribute.

Most of the approaches we found in our pre-study for analyzing IoT data are based on capturing network traffic [10]. With the audit hooks [11], the capturing of data happens within the boundaries of the application code of receiving systems. We found no similar concepts to this mentioned in the literature, and few results when searching for commercial tools for monitoring supporting this.

⁷<https://docs.datadoghq.com/monitors/create/types/anomaly/>

Chapter 3

Research Methodology

In this chapter, we describe the research methodology used in this thesis. First, we present the overall purpose of this thesis and then describe our research questions. Next, we present the context of the collaborating company and the participating case teams. We present our method for developing the thesis by applying design science research (DSR), the research paradigm, and the data collection methods and data analysis. Finally, we describe how we consider ethics in this project and security measures for protecting the collected data.

3.1 Research Objective

This study aims to understand the security threats associated with handling data from external sources from the point of view of a developer, in the context of a development team offering services associated with the data. We use the insights from this study to propose an architecture for a data monitoring system with threat intelligence capabilities.

In this thesis, we focus on security threats that have their origin in tampering of data, which impacts data integrity. Other security properties, such as availability, could also be affected due to tampering. Sensor data that comes from IoT devices are especially relevant, as these devices come with many challenges, which makes securing them difficult [5, 6, 52]. We limit the scope to tampering of textual and numeric data since these most data sources have these data types.

3.2 Research Questions

The research questions for this thesis have evolved throughout the project, and their final form as as following:

- **RQ1** What are the practices reported by companies for securely handling data coming from devices or other sources?
- **RQ2** How can data-centric threat modeling support teams in identifying security risks and promote secure design in handling data?

- **RQ3** How can data monitoring using audit hooks identify attacks in data with increased risk?

3.3 Context and Participants

In order to answer the research questions we have posed, we conducted a study with Equinor, Ryde, and a student organization at NTNU. These organizations vary greatly in their scale and use different kinds of data to develop different products. In order to collect data from and develop and evaluate a threat modeling method (see Chapter 4) in different contexts, we included different kinds of organizations in our data collection. Before the publication of the thesis, the participants got access to the relevant parts to check that the content related to themselves was correct and permitted to be published.

We describe the demography of the teams participating in this study in the following sections.

3.3.1 Equinor

The primary collaborating company for this thesis is Equinor, which is an international energy company with over 21 000 employees in almost 30 countries¹. The company works closely with its suppliers. They have around 2000 developers, where around two-thirds are suppliers, and the size of the teams is often small. These developers are responsible for developing and maintaining over 5000 applications built on a wide technology selection from the past 30 years.

The anticipated use of IoT data was central in finding collaborating companies for this thesis. Equinor has many teams concerned with handling data from sensors deployed on a variety of equipment onboard oil and gas installations. The department from Equinor we have collaborated with in this thesis is most often not concerned with IoT data in their services. Instead, they are concerned with various data sources that mainly originate from other external companies. Despite not being concerned with IoT data, their handling of external data sources seemed like an interesting case. In addition, we also had contacts within this department, making it easier to get prioritized by the development teams when researching as externals.

The participants from Equinor were selected in the process of snowball sampling, where the initial participants put us in contact with other people they thought might be relevant for us to talk to or included the people they deemed relevant in the DPF sessions. In the following sections, we describe the primary collaborating teams in Equinor.

¹<https://www.equinor.com/en/about-us.html>

Team Atlas

Team Atlas results from an initiative to offer a data platform that integrates various data sources related to marketing and midstream processing, which were previously spread across many analyst and data scientist environments within Equinor. The collaborating team consists of about 8 members with various skills in development and infrastructure, platform architecture, and machine learning, and follows a loose implementation of agile. Their services are used across several data-product teams in the extended Atlas team of around 80 members and other users of the platform. Team Atlas is responsible for providing this data platform to any team wanting to access third-party (often licensed) data sets while also ensuring storage and availability of these data sets. The team has been around since 2020 but was restructured in the second half of 2021.

Team EurekaML

The EurekaML team maintains a platform that acts as a toolbox for their users to process data or create machine-learning models. The primary goal of the data processed in this platform is to give gas trade recommendations to support market analysts in their decision-making. They get their data from the data platform provided by the Atlas team and carry the superset of all the data which their users need. The team consists of around 6 members working with the platform's development and assisting users of the platform in their development. The team works in sprints, using a Kanban board, and has been around in its current form since the second half of 2021.

Team AR/VR

The AR/VR team visualizes 3D models of Equinor-governed installations, such as oil platforms and ships, and makes them available to different environments and users in Equinor. The delivered software enables users to interact with the models and augments them with relevant data from various sources. The team consists of about 15 members of different roles, including developers, designers, testers, and tech leads, and uses elements from Kanban, Scrum, and DevOps in their development process. The AR/VR team has been around since 2017, with most of the team members having more than 3 years of experience on the team. They create several different products for internal use, including mobile and web apps and virtual and augmented reality products, which employees use in the office or on platforms.

3.3.2 Webkom

Webkom² is the committee with the technical responsibility for the student organization Abakus at NTNU. The committee consists of 17 student members, of which

²<https://github.com/webkom>

all are developers with varying years of experience and development skills, such as web development and infrastructure development. They have weekly standups and work iteratively but do not follow any specific agile methodology. The team develops and maintains multiple open-source systems, including, but not limited to, the Abakus website, the committee admissions system, a voting system used by multiple student associations, in addition to providing tech support for the users.

The data handled by Webkom will, in many cases, contain personally identifiable information about students. Hence they must be confident that their current handling of the data is secure and following GDPR regulations. The data source is mostly students' input data, using online forms or websites created by Webkom.

3.3.3 Ryde

Ryde is a Norwegian-based company focusing on delivering micro-mobility services in various cities in Norway and Sweden. Since their start in 2019, they have become one of the market leaders of micro-mobility services in many Norwegian cities, operating a fleet of several thousand electric scooters. Customers access the electric scooters through a mobile application that communicates with the electric scooter's embedded IoT system. The application development team resides in a non-english speaking country and consists of 3 developers and one user experience designer. In addition to the client application, the development team also maintains several internal applications to support functions such as customer service and mechanics/operations (performing maintenance and distribution of electric scooters).

Ryde processes data generated from users and the IoT devices to bring value to business decisions at Ryde. Examples include better understanding their current operating areas, detecting erroneous functions in their services, and customer usage patterns. Because of their relevance to IoT, and also because we had contacts within the company, we decided to include them as a collaborating company. The participants from Ryde were the chief operating officer (COO) and a local leader for operations.

3.4 Approach for Research

In this thesis, we employ a framework for carrying out design science research (DSR) projects as described by Johannesson and Perjons [55], which is similar to the methodology proposed by Peffers *et al.* [56]. We chose this methodology because it has a well-defined structure and aligns well with our goal of creating new artifacts to improve on a practical problem of general interest. The chosen approach for research will help us to: (1) develop a data-centric threat modeling method for companies to evaluate security requirements related to their current or intended handling of data coming into their systems, (2) evaluate the need for implementing security measures for such data which carries increased risk,

and (3) provide recommendations on a monitoring system for identifying threats related to tampering.

The five primary activities of DSR are, as shown in Figure 3.1, (1) explicate the problem, (2) define requirements, (3) design and develop artifact, (4) demonstrate artifact, and (5) evaluate the artifact, and several different research strategies can be used to carry out each activity. The framework consists of several activities that are not temporally ordered but are based on input-output relationships, reflecting the iterative manner of working on design science research projects.

The thesis started as an action research (AR) project due to a wish to work closely with practitioners to solve a relevant practical problem while simultaneously contributing to scientific knowledge. As a result, our thesis draws on several of the principles of canonical action research (CAR), as described by Davison *et al.* [57]. CAR is a prominent form of action research in the field of information systems (IS). It is an iterative approach consisting of five stages: (1) diagnosis, (2) planning, (3) intervention, (4) evaluation, and (5) reflection. When the product of an AR project is an artifact, it becomes very similar to a design science project [55].

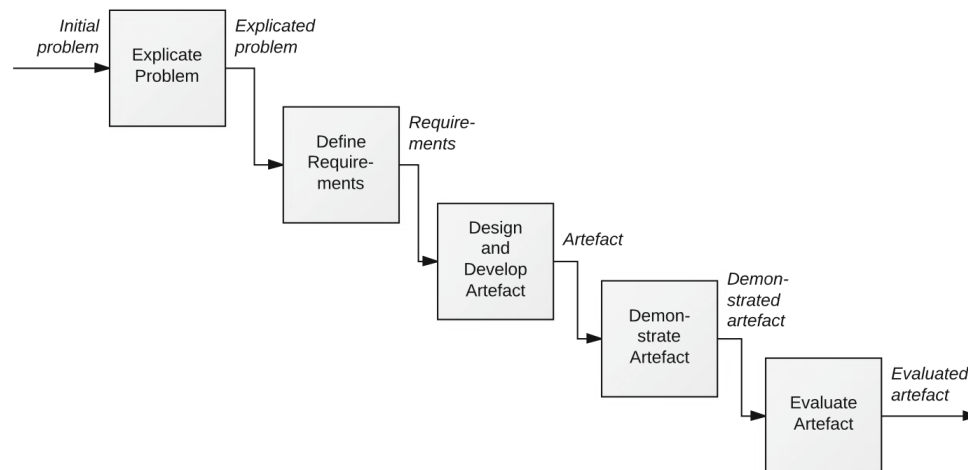


Figure 3.1: The activities in a method framework for design science research projects.

Illustration from Johannesson and Perjons [55]

3.5 Method for Thesis Development

In Figure 3.2 we illustrate how we used the DSR framework to drive the develop of this thesis. The figure captures a timeline of the activities we performed and their input/output relationships.

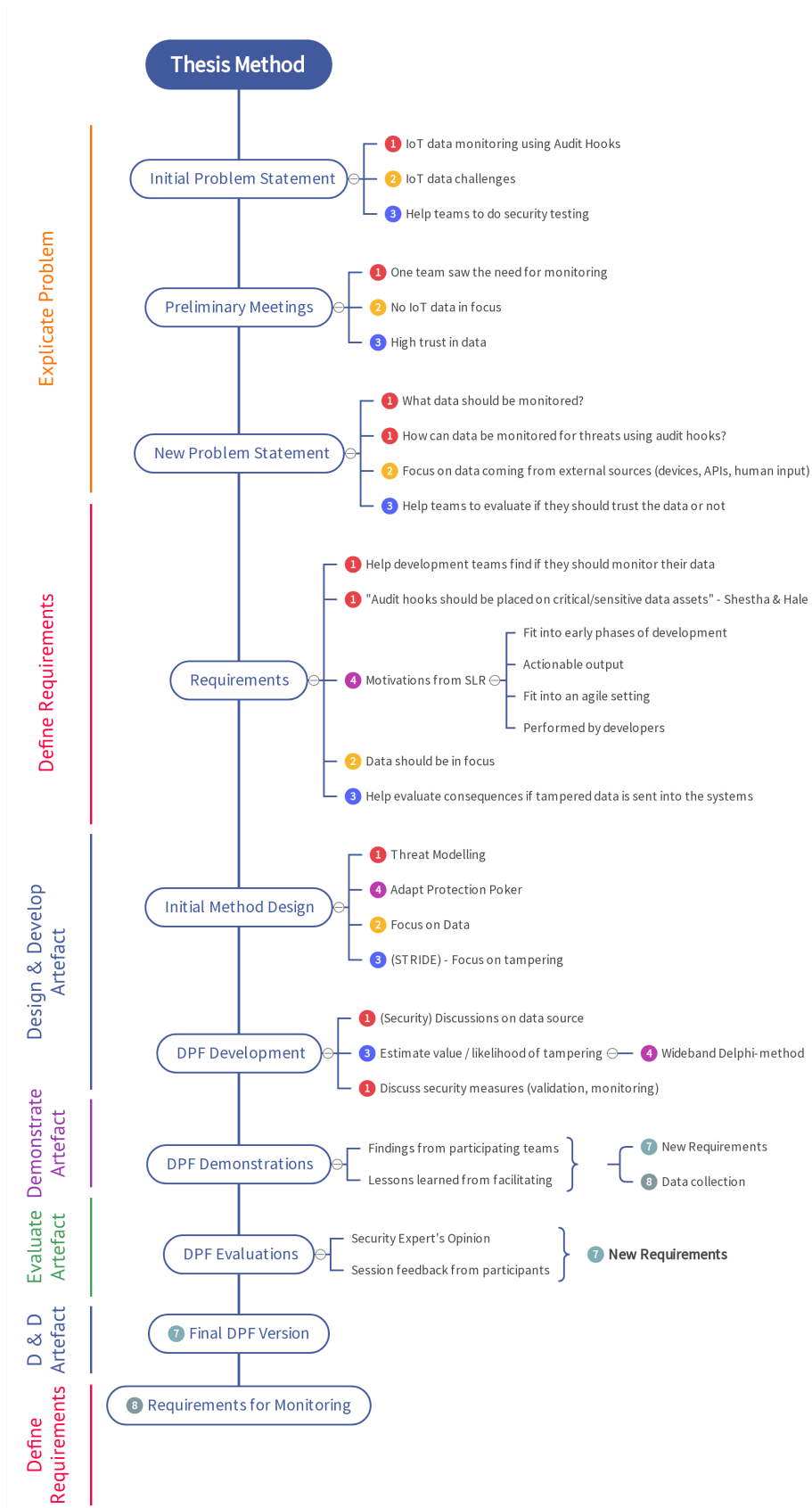


Figure 3.2: The activities from design science research used for development of this thesis.

We acknowledge that an illustration can not truly capture the iterative nature of developing a method. We often had to move back and forth between eliciting new requirements, designing, and even revisiting our problem statements to develop the method proposed in this thesis. Despite this, the illustration helps get an overview of the main output from the activities and their rough order. In the text, we organize the steps by their respective activity in DSR.

3.5.1 Explicate Problem

Explicating the problem involves investigating and analyzing a practical problem of general interest. In this phase, we lay the foundation for the focus of our thesis. The following sections describe the activities performed and their outcome in more detail.

Initial Problem Statement

Based on what we had learned from writing our systematic literature review [10], three problem statements had spawned for our thesis. Firstly, we wanted to focus on IoT data and its challenges. Secondly, we wanted to build on the approach for monitoring data proposed by Shrestha and Hale [11]. Finally, we had an overarching goal that we wanted to help development teams do security testing. As illustrated in Figure 3.2, many of the initiatives in later activities are descendants of these three problem statements (enumerated as (1), (2), and (3) in the figure).

Preliminary Meetings

As part of the first "diagnosis" phase in AR and the "explicate problem" activity in DSR, we held unstructured meetings with five development teams within one department in Equinor during the first few weeks of the project. From these meetings, we gained insight into what the development teams were working on, what kind of security testing they were doing, and which security threats the teams were concerned with. We also had a meeting with a security expert in Equinor who works with their application security, where we gained insight into how they work on security and whether a data monitoring system could be interesting.

An important bi-product from some of the meetings included recommendations of other possible stakeholders within Equinor who might be interested in our study. Though most of these stakeholders were organized in different units within Equinor, they could still provide valuable insight and were relevant for us to contact.

Key insights from the preliminary meetings include:

- The massive scale of development makes enforcing global security guidelines difficult since the context of the applications varies greatly, and consequently, so does the need for protecting the systems. Therefore the implementation of security guidelines is mostly left to the teams.

- Threat-modeling is promoted to developer teams as a good way to start building a security mindset, but few of the teams we spoke with had tried it.
- All teams had integrated a commercial SAST and SCA tool to catch security flaws in their code repositories, but a few teams were unsure how to best deal with its findings.
- The services delivered by the teams often revolved around handling large amounts of data from various sources.
- Several teams mentioned a high level of trust in the incoming data they used in their products.

New Problem Statement

Based on the insight gained from the preliminary meetings and from searching for related works, we narrowed down the focus of our research to encompass security in handling data coming from sources external to the teams. We also wanted to explore methods that could help teams determine if they should monitor their data and how to do this. Finally, we also wanted to challenge the inherent "trust" in the data handled by the development teams. The problem statement was further developed throughout the project and is expressed through the research objective defined in Section 3.1 and the related research questions in Section 3.2.

3.5.2 Define Requirements

When defining requirements, one outlines a solution to the problem and elicits requirements for an artifact. In this thesis, two artifacts were relevant to define requirements for: (1) a data-centric threat modeling method and (2) a data monitoring system, although the focus in this thesis is on the first artifact. Though gathering of requirements for this thesis was an iterative process, they initially emerged based on the results from the previous "explicate problem" phase and motivations from our SLR as described in Section 2.5.1. The requirements include that data should be in focus and that we should develop a way to help development teams evaluate the consequences of tampered data and how they could protect their systems against it.

3.5.3 Design and Develop Artefact

In this phase, we did the design and development of the two artifacts delivered in this thesis. This activity aims to produce an artifact that addresses the explicated problem and fulfills the requirements.

DPF Development

The first artifact is a method for development teams to do data-centric threat modeling. We develop the method through several iterations of development, demon-

stration, and evaluation, the latter two of which are described further in later sections.

Initial Method Design Initially, we designed a method for teams to do threat modeling on their data sources to evaluate the need for data monitoring. A few factors had to be considered when developing a method for the development teams. First, the development teams had little time to learn and go through more elaborate threat modeling activities, such as STRIDE. Second, we wanted to focus on the data sources used by the development teams, while we had previously learned that each team's knowledge about the data sources varied greatly. These factors and motivations from our SLR influenced the options we had for designing a method that would allow us to collect the data we needed to answer our research questions while bringing value to the teams participating. With this in mind, and after looking for existing methods in the literature that could be suitable for our case, we adapted Protection Poker (see Section 2.4) to focus on the data sources used by a system. We also looked in the literature to learn more about the challenges of similar methods.

Final DPF Version The method evolved and improved from the initial version after demonstrating and evaluating the method with different teams in accordance with design science research. The final DPF method is as described in Chapter 4, and is the result of several cycles of improvements. When further developing the method, we used insights from requirements that emerged from evaluations from participants in the sessions, the opinions of security experts, and our lessons learned from facilitating the activity. We also used literature and acknowledged sources to strengthen the theoretical backing of the different parts of the method. Although the method had its origins in Protection Poker, it diverged enough to become a standalone method.

Data Monitoring System

The second artifact is a data monitoring system. Some requirements for the proposed data monitoring system were based on the original idea proposed by Shrestha and Hale [11], as well as some insights from interviews with a collaborating team. Most of the proposed requirements for monitoring were developed from experiences reported by the participating teams during the DPF sessions. We also presented the idea of monitoring data using "audit hooks" to a security expert to get opinions used to develop the requirements.

Due to the scope and limited time of the project, this artifact did not reach the maturity needed for demonstration and evaluation.

3.5.4 Demonstrate Artefact

When demonstrating an artifact, one aims to prove its feasibility by showing that it can address the explicated problem. It was an iterative process along with developing and evaluating the method, where we tested DPF in multiple real-life case studies. A data extraction interview guide was prepared for each team and was journaled in during our sessions. In addition, the digital version of DPF produced data through the use of Mentimeter, where participants vote and give feedback. The lessons learned from facilitating, such as how the teams responded to the discussion questions, were used to generate new requirements for the method. In tandem, we also collected data to analyze the team's current handling of data. We describe the data analysis of the data generated during the demonstrations in Section 3.7.

3.5.5 Evaluate Artefact

We evaluate the DPF method in order to determine how well it fulfills the requirements and to what extent it can address the research problem. In this thesis, we do a formative evaluation to improve the artifact further. After each demonstration, the participating team joined for an evaluation of DPF. During the first demonstrations, we only collected oral feedback from the participants. We found that participants found it easier to give more thorough feedback this way than the planned anonymous textual feedback. We asked the participants open-ended questions to discuss potential improvements to the activity. In the final two sessions, we decided to complement the oral feedback by including a questionnaire, using the Technology Acceptance Model (TAM) [58] as an inspiration for the questions. Adding this evaluation allowed us to better measure the participants' attitude towards using DPF.

Besides evaluating DPF with the teams, we also got two security experts in separate meetings to evaluate its potential and the validity of the questions and security measures presented to the teams in the activity.

Many new requirements sprung from these evaluations, which we use to develop DPF further. Section 3.7 describes the data analysis of the data generated from evaluations.

Evaluation of DPF with Participants

In order to evaluate and further develop Data Protection Fortification (DPF), we evaluated the activity with the teams after each session. By analyzing the results from the evaluation, we can also assess the potential of the method to be viable for teams in practice. Furthermore, we can learn what kind of data the method is viable for and how the method needs to be adapted to be applicable in different contexts. Limitations of our evaluation are discussed in Section 6.3.

In all the sessions, we collected oral feedback by asking questions such as:

- *What went well?*

- *What did not go well?*
- *What did you learn (about security, the data source, the system)?*
- *How could the activity be adapted to fit your team better?*

The participants were also encouraged to comment on other things that came to mind.

As a part of the learning process, we improved our evaluation in the last two sessions by supplementing the oral evaluation questions with more quantitative evaluation questions using Mentimeter. For this, we included a more structured evaluation inspired by the Technology Acceptance Model (TAM)[58, 59] for collecting quantitative data, as well as asking each participant to order the different steps in DPF by how useful they felt it was.

In Table 3.1 we have listed the statements included in the digital questionnaire we used for the evaluation. We evaluate the participants' perceived ease of use, perceived usefulness, and acceptance of the method. Each participant answered by giving a score of 1-5, representing their attitude towards the given statement, where 1 equals to *Strongly disagree* and 5 equals to *Strongly agree*.

3.6 Data Collection Methods

This section describes the method used in this thesis for data collection. Data generated from facilitating DPF in focus groups as well as holding interviews were the primary sources of information in this thesis which we use to answer the research questions.

We facilitated in total 5 DPF sessions with 5 different teams, with each session lasting 1,5-2 hours. The details of each session can be found in Table 5.2.

In preparation for all of the DPF sessions except one, we held a semi-structured scoping interview to select a data source to focus on and to ensure that we had enough context as externals in order to facilitate the session well. Appendix B shows the interview guide for this meeting. It also helped the researchers from interrupting too much during the actual session with clarifying questions that only increased understanding for the researchers and not the team members. The interview lasted around 0,5-1 hour and included 1-2 people from the team.

During the DPF sessions, data was generated in several different ways, including journaling discussions in a data extraction document, data generated by Mentimeter for estimation and evaluation, and a Miro board. Appendix C shows the final version of the data extraction form. The researchers switched during the sessions between being the moderator and being the note-taker.

All interviews and focus groups were journaled in Microsoft Word, followed by a post-meeting discussion between both researchers to record more details and insights. We did not record the preliminary meetings and early interviews to encourage free discussion. In order to increase the trust in the findings, we instead do data triangulation by checking the findings with the source teams. The researchers did not collect any personally identifiable data.

Table 3.1: TAM inspired questions used to evaluate Data Protection Fortification. We look at perceived ease of use, perceived usefulness, and acceptance.

Parameter	ID	Question
Ease of Use	PEU1	The activity had an appropriate length
	PEU2	I would be able to perform this activity myself
	PEU3	The discussion questions were clear and easy to understand
	PEU4	It was easy to use Mentimeter for estimation of data fields
	PEU5	It was easy to collaborate on security measures in Miro
Usefulness	PU1	The activity felt useful to me
	PU2	The activity increase my effectiveness in identifying areas to focus security efforts on
	PU3	The discussion questions were helpful in order to evaluate the risk of the data source
	PU4	The suggested security mechanisms helped identify relevant mitigations
	PU5	The activity fits into our team's way of working
	PU6	I know more about security after this activity
Acceptance	AC1	I will use this activity, or parts of it, on future data sources
	AC1	I would like to investigate some security concerns that were identified
	AC1	It was fun to do this activity

A loosely structured interview was also done with two security experts in separate 30-45 minute sessions, to receive feedback on the potential for Data Protection Fortification from an expert's view.

3.7 Data Analysis

We analyzed the qualitative DPF session notes by mapping the notes from the session to a data extraction sheet template in Microsoft Excel. The template was based on the questions asked in DPF (see Section 4.5), but as questions evolved, we created categories that covered questions with similarities. After extracting the data, we did thematic coding analysis [60] by first creating initial codes from the answers from each team and then transferring these to a mind map for further clustering and development. As new codes emerged, they were regularly checked

against the data to ensure consistency. The codes were first clustered in categories and then further clustered in an exploratory manner, where similar characteristics or ways of handling the data (practices) were grouped.

The practices and characteristics found in each category were marked based on how they might affect the risk of data tampering. We do this evaluation based on best practices (and examples of the opposite) we find in OWASP [61], OWASP Proactive Controls Guide³ and our own experience as developers. Characteristics or practices that contribute positively toward these security properties are marked with a green check mark, while characteristics that indicate risk are marked with a red flag. An orange flag was used for characteristics that should be investigated further. If there was no clear indication of how it affects the risk, it was marked with a grey icon. For evaluating the practices, a combination of guides from OWASP⁴ was used, in addition to our own expertise.

The researchers then discussed the findings and significant results extracted and discussed.

We analyzed the evaluations from the DPF sessions to assess the participants' attitudes towards using the method. The qualitative data from the oral evaluations were translated from Norwegian, summarized, and subsequently analyzed with inductive coding. One researcher did the initial coding, and the codes were transferred to a mindmap, where they were further clustered and discussed by both researchers. The codes were grouped by the different parts of the method (discussion, prioritization, security measures) and the relevant TAM categories. A similar analysis was done for the comments from security experts.

The quantitative data from the evaluations were visualized in appropriate graphs.

3.8 Research Paradigm and Bias

The research paradigm chosen for this research is interpretivism [62]. We recognize that there may be multiple ways to interpret our data, and thus there may be multiple conclusions to draw to answer our research questions. We are not expecting to find proof that our way of assessing risk in a data source is the one to follow in all cases. Instead, we aim to understand which factors influence the risk of data tampering and when monitoring data for threats can help reduce the risk of using the data source. With this, we hope to create knowledge that can serve as a basis for others to advance on.

As researchers, we recognize that we are not neutral and have assumptions and values that inevitably shape this research. We acknowledge that we may influence the participants' understanding of concepts or practices. For instance, when we are giving examples to explain what is meant by a discussion topic, this may narrow their answers. The participants may also change their practices due to the

³<https://owasp.org/www-project-proactive-controls/>

⁴<https://owasp.org/>

researcher or the results of the research. All of these factors may make it difficult to reproduce the study. We also heavily depend on qualitative data, which is associated with the interpretive paradigm [55].

3.9 Ethics

When conducting research, considering the ethical issues is important [62]. This thesis collected only years of experience and current position as background information from participants during data collection.

We followed NTNU's guidelines⁵, and since we initially expected to collect personal data in this research project, including voice recordings of participants, we sent a notification form to the NSD⁶ for approval (see Appendix D). The application was first approved on the 4th of February 2022, but after updating our data collection methods, we sent an updated NSD form which was approved 18th of April 2022.

3.10 Data Storing

Both storage and transfer of personal data were processed by NTNU Office 365, including Microsoft Sharepoint, Microsoft Teams, and Microsoft Word. All personal data was stored in Microsoft Sharepoint and accessed through Microsoft Teams. Microsoft Word through Microsoft Sharepoint was used for journaling the Data Protection Fortification sessions held during the project period, only accessible to the master students and their supervisor. The results Mentimeter generated during Data Protection Fortification were deleted after extracting the data to Microsoft Word stored in Microsoft Sharepoint. No sensitive information was processed in Miro, and all results were removed after data analysis.

A Data Management Plan was created in NSD, and the security classification level of this research project was determined to be "Internal". NTNU's Data Storage Guide recommends using NTNU Office 365 as data storage for research data at this security classification level.

3.10.1 Security Measures

Multi-factor authentication is enabled by NTNU Office 365 in order to access any of its services. When information was journaled throughout the Data Protection Fortification sessions, any confidential information not related to the thesis was omitted. Microsoft Sharepoint provides an audit log of changes done to the document. We shared the notes with access-control through Microsoft Sharepoint when sharing findings with respective teams.

⁵<https://innsida.ntnu.no/wiki/-/wiki/English/Collection+of+personal+data+for+research+projects>

⁶<https://www.nsd.no/en/data-protection-services>

Chapter 4

Design of Data Protection Fortification

This chapter describes the design of Data Protection Fortification (DPF), a data-centric threat modeling method for development teams to (1) share knowledge about a data source and discuss possible security implications, (2) collaborate on identifying the risk of data fields, and (3) evaluate security measures for implementation to reduce the risk of receiving data that has been tampered with. Figure 4.1 illustrates the three main processes included in the method and the artifacts generated by each.

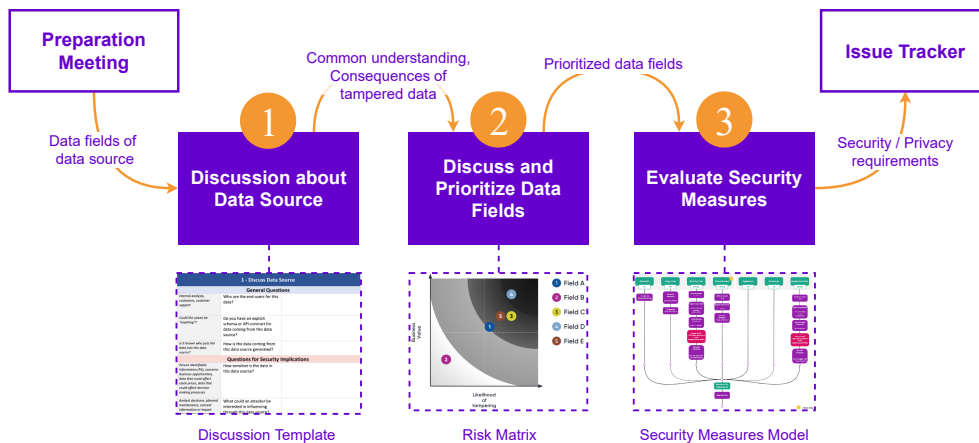


Figure 4.1: Overview of the 3 steps in DPF, highlighting their input/output relationship, and artefacts generated by each step. The last artefact generated is a model of the wanted security measures when handling the data source. The measures should be documented as tasks in the development teams' issue tracker.

Since many security testing methods require a threat model as input [51], the model generated from DPF may also provide the foundation necessary for development teams to start security testing.

The method was designed to be used on external data sources which is used

in a system. In the context of this thesis, we define an external data source as a source of data which the team integrates into their systems, where the data is not within the control of the team. This could be an integration of weather data (generated externally) to augment the user interface of a mobile application. Other examples include the use of sensor data sent from IoT devices or data sets which includes historical readings of a temperature sensor. The method has also shown to be viable for other data sources, such as form input data posted to a backend application.

Investments in cybersecurity involve decision-making under uncertainty [63], and one of the techniques for this decision-making is to do risk analysis [64]. DPF is one way of doing such risk analysis, and the outcome of the method can help decision-makers identify which data fields can and should be monitored for threats. Here, the uncertainty comes from difficulties in knowing what and where attacks are likely to happen and what the exact consequences could be. To mitigate this uncertainty, DPF can help practitioners decide where to focus their security efforts to protect systems from possible threats coming from data sources they do not control. When designing the method, there were several considerations and previous learnings to take into account, and Table 4.1 shows an overview of these.

Table 4.1: The desired properties of Data Protection Fortification.

Desired property	Source
It should guide the team in identifying risks related to data tampering.	Research problem
It should be able to identify data fields that are suitable for monitoring.	Research problem
It should be possible to run with teams who work remotely.	Preliminary meetings
Developers should be able to perform the approach and use the results.	SLR [10], Related work [23]
It should be possible to perform in the early stages of development.	SLR [10]
It should have a clear and actionable output. The output artifacts should be possible to use as documentation.	SLR [10]
It should fit into an agile context: Enable inspection of security efforts and encourage close collaboration on embedding security into the product.	SLR [10], Related work [23]
The method should create value for developers.	Related work [29]
It should have a clear process to follow.	Related work [29]
It should have good descriptions of how to perform the method for practitioners, including providing templates and examples	Related work [23]
It should be clear when threat analysis can stop	Related work [23]

In the following sections, we first outline what is needed to start doing DPF, and then describe the design of the activities involved in the method. In Section 4.3

we describe an implementation of DPF which is suitable for both remote and co-located teams. We also provide an alternative variant in Section 4.4 for co-located teams that prefer not to use digital tools.

4.1 Preparations for Data Protection Fortification

When preparing the DPF activity, there are a few relevant questions to consider. These questions include:

- *What data source should be in focus?*
- *Who should participate?*
- *What tools should we use during the activity?*

In the remainder of this section, we go into more detail about how to do the preparations to help answer these questions.

4.1.1 When to Perform Data Protection Fortification

We propose to make DPF a routine when assessing the security impact of adding a new data source to a system. In addition, performing this activity on existing data sources is relevant for assessing if the current security efforts for the data source are sufficient. It should be repeated for a data source when changes in the architecture affect how the data triggers transactions or decisions in the application or when the data fields change.

4.1.2 Select Data Source

Several selection criteria can give input to the team on which data source to choose. If the team has no prior experience with DPF, we suggest selecting a data source which the team is well acquainted with. Other good candidates are new data sources that the team considers for integration into the system under development, or previously integrated critical data sources. The selected data source will be the sole focus during the DPF session.

4.1.3 Prepare Data Fields

The participants will discuss the data fields within the data source during the activity. Therefore, prepare a list of the known data fields in advance, including their data type. If only some data fields are being used, it is possible to scope down the session by focusing on those.

4.1.4 Participants

DPF is designed for development teams, but the extended development team (including the project manager, product owner, security experts, testers, and designers) can also see relevance in participating. We also recommend including some-

one with in-depth knowledge about the data source in focus and how it is used, for instance, a representative from the supplier of the data source. This person can help address questions during the discussions and help identify relevant data fields within the data source.

4.1.5 Tool Recommendations

In this section, we give recommendations on what tools to use for performing DPF. We recommend some specific tools beneath based on our own experience of using them.

General Recommendations

- **Documenting:** We have created a Microsoft Word template¹ that can be used by facilitators to take notes in during DPF. We have also created a printable version² of this document.
- **Security Measures Glossary:** The security measure glossary is a reference tool for the participants to help evaluate relevant security measures for the data source. This glossary can be printed, or presented digitally.

Tools Recommended for Remote Teams

The digital tools listed below are also suitable for co-located participants.

- **Digital white-board:** We recommend using a digital white-board to draw the security measures onto each data field. For this, we have used Miro, which we prepared before each DPF session. A similar alternative to Miro is Mural³, but we did not test this tool in practice.
- **Digital tools for collecting votes:** For this, we have tried one tool, Mentimeter, for collecting digital "votes" during the prioritization of data fields. Other tool options include, Kahoot!⁴, or AhaSlides⁵, but these have not been tested for this purpose.

Tools Recommended for Co-located Teams

- **White-board:** Participants can use a white-board to draw the risk matrix and the data fields to be discussed, as well as the security measures on each data field.
- **Collecting votes and security measures:** The participants can use post-it notes to give estimations for data fields and to document security measures.

¹<https://github.com/Gullskatten/dpf-templates/blob/c93c60f962ca63654e2510643dd8788c81c924ce/DPF-Data-Extraction.docx>

²<https://github.com/Gullskatten/dpf-templates/blob/f884734c526b744698f281271d7ddf1659fbd973/DPF-Data-Extraction-printable.pdf>

³<https://www.mural.co>

⁴<https://www.kahoot.com>

⁵<https://www.ahaslides.com>

4.2 How to Perform Data Protection Fortification

In this section, we describe how each step in a DPF session is carried out on a conceptual level. Figure 4.2 gives an overview of all the steps in DPF.

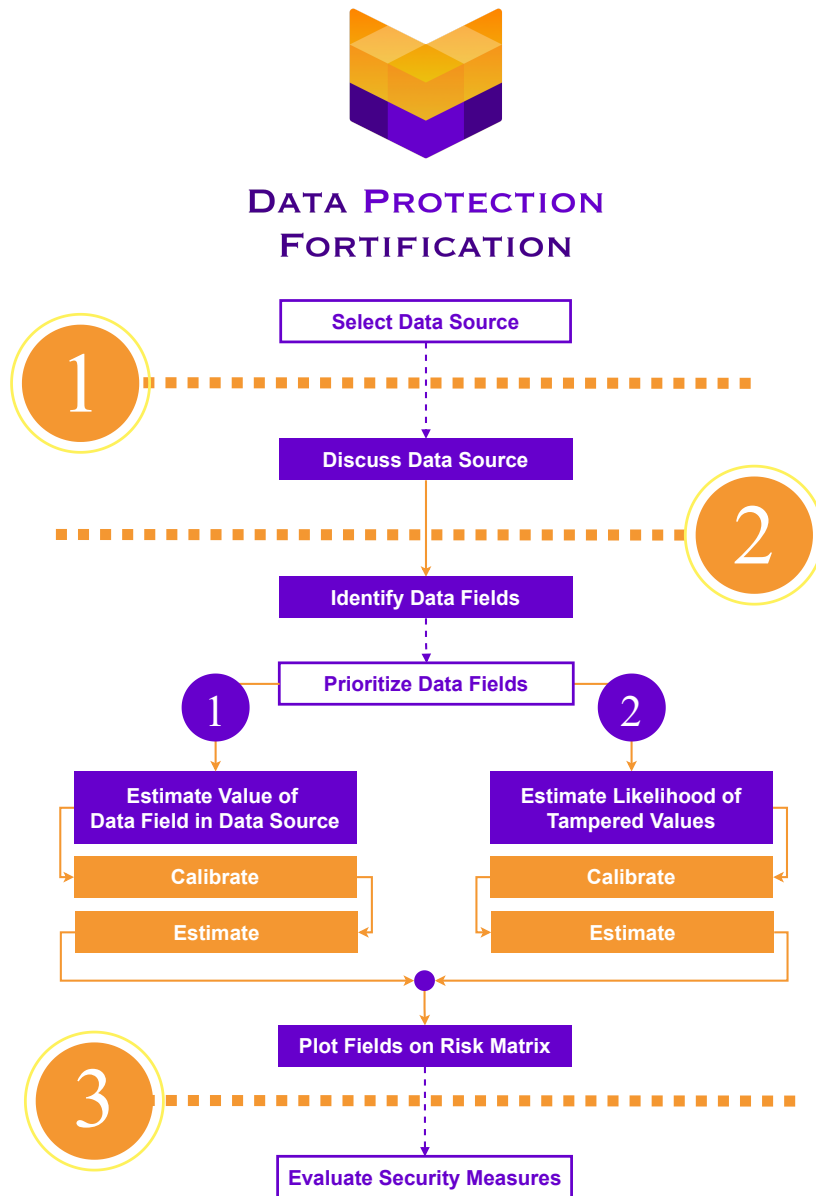


Figure 4.2: High-level diagram of the steps in Data Protection Fortification. The directional arrows show the order of each step in the method.

4.2.1 Discuss Data Source

The first step of DPF is a two-part discussion of the data source in focus. First, the participants discuss general information about the data source. Second, the participants discuss possible security implications for the data source.

The questions are discussed in Section 4.5, and a cheat sheet for practitioners can be found in Appendix A. Using the question guide, we encourage teams also to document their answers, especially if action points come up during discussions. These action points can then be added as tasks to the backlog after the activity concludes.

4.2.2 Identify Data Fields

If the facilitator has not prepared a list of the data fields that come with the data source during the preparations, the team does it in this step. Note down the field names and expected data types of the values (e.g., character values, numeric values, boolean). Typically one team member starts by describing each data field to ensure a common understanding.

This step also includes discussing how each data field influences the system, which can be as simple as mapping the field to its usage. For example, *temperature_celsius* influences production of energy and *assigned_user* helps determine user assigned to a task. Considering this helps later estimation efforts when determining the value of a field and strengthens the participants' understanding of the usage of the data fields.

4.2.3 Prioritize Data Fields

The process described below is the same for both estimation activities.

The estimation activities have their origin in Protection Poker (see Section 2.4). The estimation of *business value* is based on *value of the asset*. The estimation of *likelihood of tampering* is based on *attack exposure* where participants should estimate the exposure (ease of attack) of a given functionality. For instance, functionality that includes user input fields would increase exposure values.

The estimation part of DPF also has several similarities to the Wideband Delphi-method [65]. The Wideband Delphi method is an estimation method based on reaching consensus, where the estimation is done in several rounds until participants reach consensus. Contrary to the original Delphi method, the wideband-variant has group discussions between each round where the participants describe their rationale for their answer.

Facilitators hide results until everyone has made their vote to prevent participants from being distracted while voting and from adjusting their estimates based on the estimates of other participants. The latter is known as the anchoring bias and can affect the initial estimates of participants [65].

Calibrate Data Fields

The participants calibrate before estimating values for all data fields to give them a sense of which fields have the most and the least values for business value and the likelihood of tampering. It should also aid in getting a better spread in the values, which can make prioritizing the data fields easier.

As shown in Figure 4.2, the participants perform the *Calibrate*-step before each estimation activity, and the questions to ask the team are given below:

- **Value Calibration:** Which field is the *most* valuable?
- **Value Calibration:** Which field is the *least* valuable?
- **Tampering Calibration:** Which field is *most* likely to be tampered with?
- **Tampering Calibration:** Which field is *least* likely to be tampered with?

General Procedure to Estimate Data Fields

The general procedure for giving estimates is the same for both estimating value and estimating *likelihood of tampering*, and is as follows:

1. Pick a data field.
2. Let each participant estimate a value without showing their answer.
3. When everyone has picked a value, let the participants present their estimation values.
4. Discuss: Let the participant(s) having higher or lower estimation values than the rest give a reason for their estimate.
5. If participants bring up new insights during the discussion, decide if the participants should take another round of providing estimation values for this field.
6. The group finds the appropriate value to assign to the data field. One may also calculate the mean value based on the participants' responses and assign this as the value.
7. If more data fields remain, start over from (1).

Estimate Value of Data Field

The first estimation activity in DPF is to estimate the *business value* of each of the data fields. When attempting to find a value for this, consider the following:

- How are functions, humans, or internal services dependent on this data field?
 - Are decisions being made based on the value in this data field?
- What is the consequence if this data field is missing data?
- As an attacker, what data field would you tamper with to do the most damage?

Estimate Likelihood of Tampered Data

The next estimation activity in DPF is to estimate the likelihood of tampered data within the data fields.

When attempting to find a value for this in DPF, consider the following:

- What is the possibility of providing malicious inputs to this field?
 - Is the source of this field coming from human input, IoT devices, or is it system-generated?
 - Is this field expected to include character values or numeric values?
- Is it possible to detect that the value in this field has been tampered with?

4.2.4 Plot Fields on Risk Matrix

In this step, we create a risk matrix using the values from the previous estimation activities. Figure 4.3 shows an illustration of a risk matrix. Each field gets plotted on this matrix using the *business value* of data field as the Y-coordinate and *likelihood of tampering* as the X-coordinate. This plot is equivalent to calculating the risk using the formula shown in Equation (4.1).

$$\text{DPF Risk} = \text{Business Value} \times \text{Likelihood of Tampering} \quad (4.1)$$

The risk matrix is an example of a boundary object, which Shostack [21] recommends adding when designing threat modeling approaches. Boundary objects are helpful when including a variety of participants because all participants can actively use them in discussions [21]. The participants can refer to the risk matrix to prioritize which data fields they should start with when evaluating security measures in the next step of DPF.

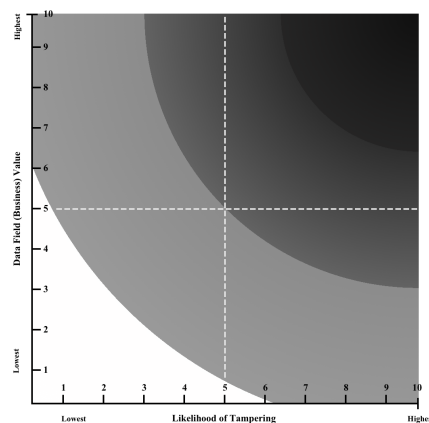


Figure 4.3: Illustration of a risk matrix used for identifying higher-risk data fields. The darker areas indicate a higher risk.

4.2.5 Evaluate Security Measures

In this step, we identify security measures that developers should apply to the fields in the data source, using the method shown in Figure 4.4. The method involves two primary artifacts, (A) a toolbox that contains security measures for commonly used data types, (B) a visualization of data fields in the data source with "lanes" for adding security measures. We suggest writing the data fields onto a physical or digital whiteboard, or somewhere they will be visible to all participants. Participants then discuss which security measures are relevant for each data field.

Participants actively use the toolbox to find relevant security measures (see Section 4.2.5) for a data field, and add these to the corresponding lane for that data field. The toolbox can be used either in a structured fashion as a checklist or more loosely structured in a collaborative brainstorming session where participants use the toolbox as inspiration. The team might discover that several of the security measures found for one field can be duplicated and added to other similar fields with a few adjustments.

We suggest that participants start by identifying security measures for the data field with the highest risk in the risk matrix. Teams may evaluate all fields or choose a threshold based on the gradients in the risk matrix.

Security Measures Toolbox

We have curated a list of security measures suitable to apply to a selection of commonly used data types. These types include, *String* (textual values), *Number* (numerical values), and *Date* (date, with or without timestamps and timezone). We decided to limit the scope to these data types, as they are typically present in most data sources. In addition to relevant measures for each data field, we have also included security measures for the data source as a whole. Table 4.2, Table 4.3, Table 4.4 lists security measures for data fields of the type *String*, *Number* and *Date*. Table 4.5 lists security measures that affect the handling of the data source as a whole.

The tables include the type of security measure, the relevant STRIDE category, and the source of this measure. In addition, we give examples for each measure and justifications where relevant. The validation measures are inspired by OWASP's guide on best practices for validation⁶ and injection prevention⁷. Such measures are recommended to prevent unauthorized input from being processed by the application. Anomaly monitoring was inspired by DataDog Anomaly monitoring⁸. We created the other measures and had these reviewed by two security

⁶https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

⁷https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html

⁸<https://docs.datadoghq.com/monitors/create/types/anomaly/>

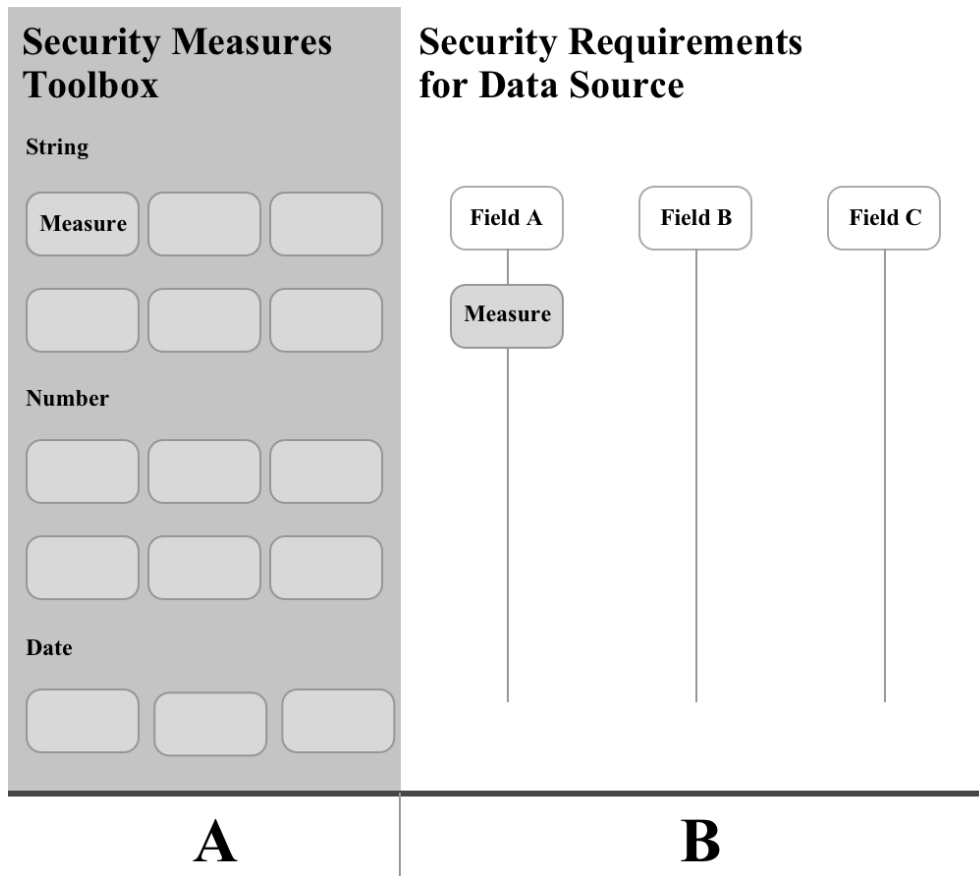


Figure 4.4: Method of evaluating security measures

experts. For measures mentioned by teams during the DPF sessions or measures that we added, we filled in a hyphen (-) as the source.

When mapping the security measures to STRIDE, some of the measures may be relevant for preventing more than one type of threat. One example of this is using prepared SQL statements which is related to several of the STRIDE categories, since an attacker may use an SQL-injections to gain higher privileges (Elevation of Privilege), cause a data leak (Information Disclosure), or delete data or drop tables (Denial of Service, Tampering). This thesis focuses on data tampering, so the tables reflect this when displaying the STRIDE category.

Table 4.2: Security measures for the *String* data type. An asterix (*) is used where security measures target more than one type of STRIDE threat.

Security Measure	STRIDE	Source
Validate length Ex: Length of characters in the field is less than 200 but greater than 0.	Tampering	OWASP
White-list values Ex: Only allow the values "Apple" or "Pear", and reject all other values.	Denial of Service	OWASP
Limit allowed characters Ex: Only allow characters "a-z" and " " (space).	Tampering	OWASP
Syntactic validation Ex: The field must match the format of a Norwegian telephone number. This type of validation often uses regular expressions.	Tampering	OWASP
Semantic validation Ex: The field "e-mail address" must match the field "confirm e-mail address".	Tampering	OWASP
Foreign key (exists) Ex: The field "UserID" must be a reference to an existing user <i>Identified during testing.</i>	Tampering	-
Foreign key (has access) Ex: The current user must have access to the ID referenced in this field <i>Identified during testing.</i>	Elevation of Privilege	-
Data type validation Ex: Verify that the value in this field is of type String (and not "null" or an integer).	Tampering	OWASP
Escaping Ex: Escape special characters in this field, such as "<" or "/".	Tampering	OWASP
Prepared SQL statements Ex: First defining all of the SQL code, then passing in each parameter to the query later (on the server-side).	Tampering*	OWASP
Value correlation monitoring Ex: The contract ID in the external data set should be the same as in the internal system. <i>Identified during testing.</i>	Tampering	-
Anomaly monitoring Ex: A market ID that usually comes in regularly with the data has been missing for the past period but should still exist. <i>To detect irregular variations in the value of a data field. Identified during testing.</i>	Tampering	-
Attack monitoring Ex: Warn if this data field receives strings in the deny-list. Warn if receiving attempted injection attacks. <i>To detect plain attempts at attacks through the data that might not be easy to catch in validation or might be interesting in metrics.</i>	Tampering*	-
Error monitoring Ex: Monitor when validation errors or warnings happen for a data field. <i>Identified during testing.</i>	-	-

Table 4.3: Security measures for the *Number* data type. An asterix (*) is used where security measures target more than one type of STRIDE category.

Security Measure	STRIDE	Source
Validate minimum / maximum value Ex: Validate that the value in the field is greater than 0 and less than or equal to 100	Tampering	OWASP
Data type validation Ex: Verify that the value in this field is a number.	Tampering	OWASP
White-list values / value ranges Ex: Only allow 1, 5, 7 in this field (white-listed values). Only allow the values 1-10 or 20-30 in this field (white-listed value ranges)	Denial of Service	OWASP
Semantic validation Ex: The field "speed" must not be greater than the field "max_speed".	Tampering	OWASP
Type conversion with strict exception handling Ex: When parsing integers, ensure that a strict exception handling is done on failure and don't allow the data to be further processed.	Tampering	OWASP
Foreign key (exists) Ex: The field "UserID" must be a reference to an existing user <i>Identified during testing.</i>	Tampering	-
Foreign key (has access) Ex: The current user must have access to the ID referenced in this field" <i>Identified during testing.</i>	Elevation of Privilege	OWASP
Prepared SQL statements Ex: First defining all of the SQL code, then passing in each parameter to the query later.	Tampering*	OWASP
Value correlation monitoring Ex: The temperature correlates with the temperature of another dataset. <i>This measure was uncovered during interviews, where data from multiple sources could explain the same phenomenon. Hence, triangulating this data to detect outliers could be of value.</i>	Tampering	-
Anomaly monitoring Ex: The temperature can not drop by more than 5 degrees since the previously received temperature within a given timeframe. <i>To detect obvious faults or more subtle attacks over time.</i>	Tampering	-
Error monitoring Ex: Monitor when validation errors or warnings happen for a data field. <i>Identified during testing.</i>	-	-

Table 4.4: Security measures for the *Date* data type

Security Measure	STRIDE	Source
Validate min/max value Ex: Validate that the date is after 11-11-2011 and before today.	Tampering	OWASP
Syntactic validation Ex: The date must match the format "yyyy-MM-ddT"HH:mm:ss.fff"Z"	Tampering	OWASP
Semantic validation Ex: The date must be in the future.	Tampering	OWASP
Prepared SQL statements Ex: First defining all of the SQL code, then passing in each parameter to the query later (on the server-side).	Tampering*	OWASP
Anomaly monitoring Ex: Detect irregular variations in the value of a date field. <i>Identified during testing.</i>	Tampering	-
Error monitoring Ex: Monitor when validation errors or warnings happen for a data field. <i>Identified during testing.</i>	-	-

Table 4.5: Security measures for handling the data source

Security Measure	STRIDE	Source
Data retention Ex: Data coming from this data source should have an "age-off" date set. After this date, the data should be deleted.	Information Disclosure	GDPR
White-list bindable values Ex: Create Data Transfer Object (DTO) for this data source, decide which values can be changed by including/excluding them in the DTO.	Information Disclosure	OWASP

4.2.6 Outcome

The tangible output of DPF includes the security measures identified for this data source, the risk matrix, as well as notes written down during the initial discussion about the data source. This documentation can be added to existing documentation about this data source or could even be the first step in documenting how the team handles it and its security implications.

For developers, we propose adding each new security measure or development initiative as a new task in the team's backlog at the end of the activity. They may be filed as bugs if the data source has already been integrated or on the feature requirement task if it is still under development. This is in-line with Shostack [21],

who also adds that identified issues should be prioritized and, if possible, marked as security bugs.

The documentation may also be used for GDPR compliance, for instance, as a part of a company's Data Protection Impact Assessment (DPIA), given that the company needs this.

4.2.7 Evaluate and Review

At the end of each DPF, we encourage teams to set aside some time to evaluate how the activity went and to review the steps and documentation generated during the activity. Inspecting the activity while it is still fresh in everyone's mind can help adapt the DPF activity to optimize its usefulness for the team. Reviewing generated documents, such as the risk matrix, may also help identify new initiatives overlooked during discussions.

4.3 Data Protection Fortification in Practice

In this section we describe how DPF, as described in Section 4.2, was facilitated in practice. This implementation of DPF is suitable both for running the activity digitally using video-conferencing tools (e.g., Teams⁹, Zoom¹⁰), and for co-located teams.

When creating this digital variant, we had to consider both practical constraints and security issues when facilitating this activity for externals. We found no backing in the literature on how to facilitate the estimation part of the activity digitally in a good way, so we have proposed our solution here. In addition, we had to consider that the content of the sessions could potentially include sensitive information about the participating team's systems and had to handle this securely.

The following subsections describe how we facilitated each step in the activity described in Section 4.2.

Discuss Data Source

We asked the team the discussion questions listed in Table 4.6, and took notes from the discussion that followed. As facilitators, we do not actively engage in the discussions. However, we provide examples of answers to help spark discussions if the participants had a hard time answering or if participants gave vague answers.

Identify Data Fields

During preparation meetings with the teams, we chose a data source and identified the relevant data fields of the source so that we could prepare Mentimeter

⁹<https://www.microsoft.com/en-us/microsoft-teams/free-video-conferencing>

¹⁰<https://zoom.us/>

slides in advance. A limitation in Mentimeter was that it was only possible to include eight options in the "scales" slide-type used for estimation. Hence it was only possible to estimate a maximum of eight data fields.

Prioritize Data Fields

For the estimation, we used Mentimeter¹¹, which NTNU has a data processor agreement with. The platform specializes in creating engaging presentations where participants join the presentation anonymously using a web browser and provide answers using various controls based on the question presented to them.

One benefit of using Mentimeter for voting is that participants can stay anonymous while voting, which can help reduce but not eliminate the bandwagon effect in the group, where participants follow the opinion of the majority instead of advocating for their view. Anonymity also helps prevent participants from being influenced by people in the group with dominant personalities or who have a higher social status. However, we encouraged the participants to share their reasoning after voting, particularly those with differing views, since it is often the views that challenge conventional thinking that may be the most important to discuss in order to reach an accurate result [65].

Before each DPF session, we had prepared a Mentimeter presentation with the selected data fields. We prepared a separate Mentimeter presentation to show the result of the estimation in a risk matrix and to perform an evaluation of the activity with the team. Discussions brought up during the estimation activity in Mentimeter were securely journaled as described in Section 3.10. We describe how the teams voted in Mentimeter in the rest of the section.

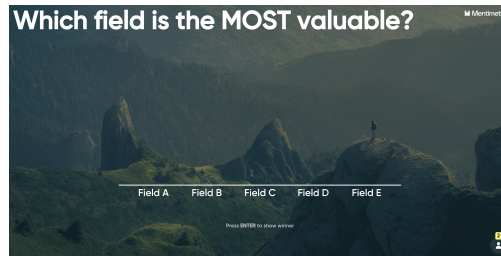
Calibrate Data Fields

In order to determine which data fields had the most and least *business value* and *likelihood of tampering*, we created a voting system in Mentimeter, as shown in Figure 4.5. The facilitator showed the result from each round of votes after every participant had placed their vote.

General Procedure to Estimate Data Fields

To give estimates on both the *business value* and the *likelihood of tampering* for each field, participants received a form in Mentimeter, as illustrated in Figure 4.6a. This form included sliders to assign values from 1-10 to each field. A description text was provided to help participants determine the estimation values.

¹¹<https://www.mentimeter.com/>



(a) The results are hidden until everyone has voted.



(b) The winner is the field that gets the most votes.



(c) Multiple winners occur if two or more fields receive equally the most votes.

 A screenshot of a Mentimeter poll interface. The question is "Which field is MOST likely to be tampered with?". Below the question, there is a sub-question: "Which field is most likely to contain malicious inputs or abnormal data?". Another sub-question follows: "Which field is 'most exposed' to human input?". Below these are five radio button options: Field A, Field B, Field C, Field D, and Field E. At the bottom, there is a yellow "Submit" button.

(d) The respondents vote on their own device.

Figure 4.5: Representative screenshots from Mentimeter when doing calibration of data fields. After the results are in, the participants discuss the results before moving on to estimating all fields.

Let's estimate: (Business) Value!

To determine a fields value, ask the following:

How are functions, humans, or internal services dependent on this field?

Are there decisions being made based on the value in this field?

What is the consequence if this field is missing data?

Field A (<DataType>) Skip

1 Least 10 Most

Field B (<DataType>) Skip

1 Least 10 Most

Field C (<DataType>) Skip

1 Least 10 Most

Field D (<DataType>) Skip

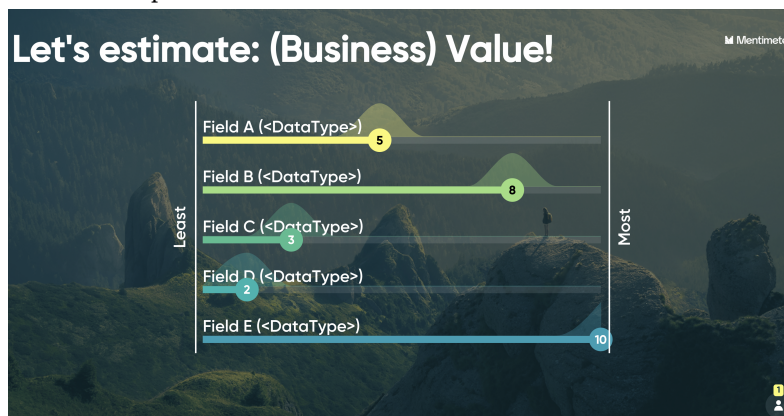
1 Least 10 Most

Field E (<DataType>) Skip

1 Least 10 Most

Submit

(a) Estimation form used by participants to provide answers in Mentimeter.



(b) Estimation results presented in Mentimeter (after all participants had given votes).

Figure 4.6: Representative screenshots from Mentimeter when doing estimation of the *business value* for the data fields. Estimating *likelihood of tampering* followed the same procedure.

Plot Fields on Risk Matrix

In a separate Mentimeter presentation, we plotted the answers from both estimation activities onto a risk matrix. A separate presentation was needed to fill in the results in the background to make the session more efficient.

Figure 4.7 illustrates the resulting risk matrix, for which we briefly explained how to read the risk matrix and highlighted the data fields with the highest and lowest scores.

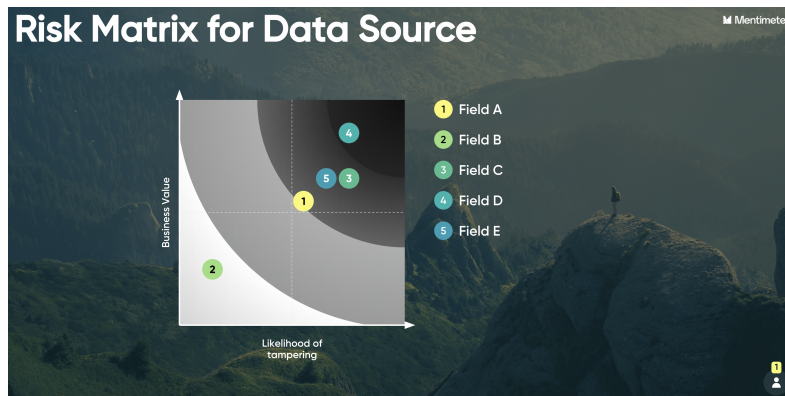


Figure 4.7: A risk matrix as shown in Mentimeter.

Evaluate Security Measures

Using the collaborative sketching tool Miro¹², we created a Miro template that would enable participants to drag security measures onto fields in their data source based on their data type. Figure 4.8 shows an illustration of this Miro template.

The Miro template consists of three panes, (A) a toolbox that contains the security measures described in Section 4.2.5, (B) a visualization of the data fields with "lanes" for adding security measures, and (C) helpful references, such as the risk matrix and related code snippets to help in prioritizing which fields to start identifying security requirements for.

Participants can use the risk matrix to prioritize where to start when evaluating security measures for the data source.

¹²<https://miro.com/>

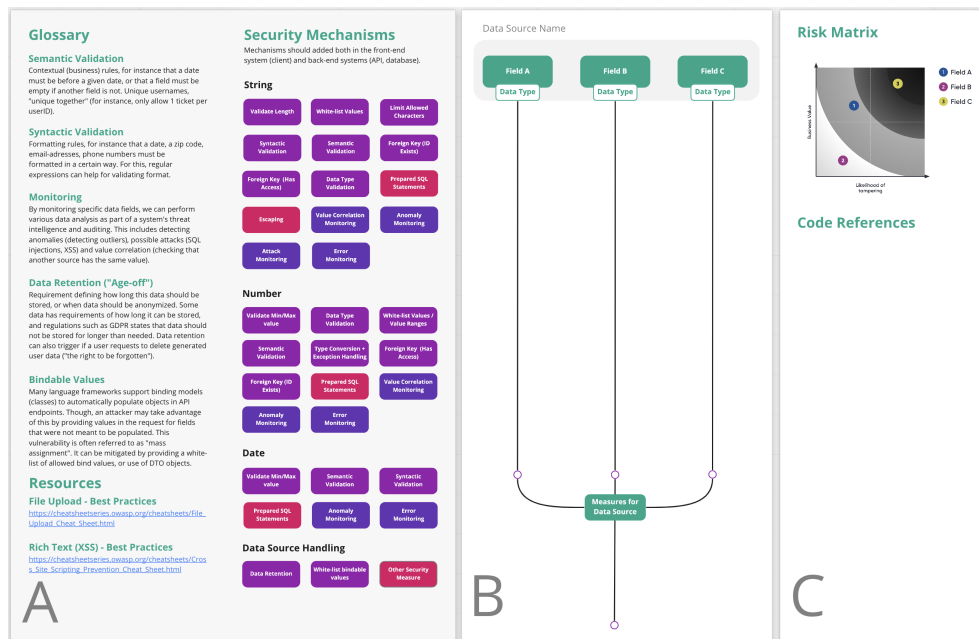


Figure 4.8: The Miro template has three panes (A,B,C). Participants drag measures found in (A) onto a data field’s lane in (B). (C) includes the risk matrix and code references to make it easier to prioritize which fields to start with.

4.4 Data Protection Fortification in Practice - Alternative

In this section, we describe a method to facilitate DPF, as described in Section 4.2, for co-located teams without using any digital tools to guide the session. Figure 4.11 shows an example of filled-in templates.

Prepare for Session

On a whiteboard, the facilitator of the session draws a table with the following columns: id, field name, data type, *business value*, *likelihood of tampering*. On a different place, draw up a risk matrix with *business value* as the y-axis, and *likelihood of tampering* as the x-axis, with values 1-10 on each. The layouts are illustrated in Figure 4.9.

Discuss Data Source

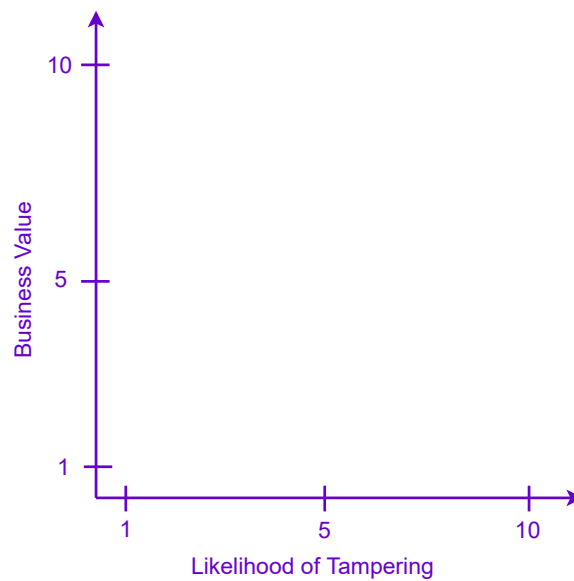
The discussion part is the same as in the digital variant.

Identify Data Fields

Someone from the team first describes each data field in the data source. If there are too many data fields, the team may collectively decide to narrow the focus

ID	Field Name	Data Type	Business Value	Likelihood of Tampering

(a) Template table for estimation of data fields on a white-board



(b) Template risk matrix for a white-board

Figure 4.9: Templates for how to draw up the table for estimation on the white-board and the risk matrix without using digital tools.

by only selecting a manageable number of fields. Based on lessons learned from performing DPF, we suggest that a boundary for this is about 20 data fields.

When the participants have decided on a set of data fields, the id column on the whiteboard is populated with rows from 1 to n (where n is the number of selected data fields). The data fields' names are then written down on separate post-it notes and pasted on the whiteboard (in the "name" column of any row). The post-its should then have the id written on them to track which estimates on the whiteboard belong to which data field when moving around the post-its. Lastly, fill in the respective data types in the table.

Prioritize Data Fields

Each participant receives a piece of paper (A4-sheet) which will act as their answering sheet. They draw a table with the following columns: id, *business value*, *likelihood of tampering*. The layout is illustrated in Figure 4.10.

ID	Business Value	Likelihood of Tampering

Figure 4.10: The template layout for each participants answering sheet during estimation rounds.

First, the participants calibrate and estimate the *business value* and, after this, continue with the *likelihood of tampering*.

Calibrate Data Fields

The facilitator presents calibration questions (see Section 4.2.3) to the participants, who then collaboratively decide on which data fields fit the best. After reaching a consensus, the participants move on to estimate the data fields.

General Procedure to Estimate Data Fields

After the calibration, the participants vote on the data fields individually on their paper. Each participant writes their answer down on their paper, and everyone shows their answer simultaneously. After discussing, the team agrees on a data

field to receive the highest and lowest scores. For each data field, the facilitator collects the estimates from each participant. After discussing and reaching a consensus on the estimate, the facilitator writes it down on the whiteboard. Participants may revote if deemed necessary, which is done by crossing over the old votes and writing new ones on the side.

Plot Fields on Risk Matrix

Once the participants fill the estimation table, it is time to use the results to create a risk matrix. The facilitator moves each post-it note representing the data fields to the appropriate place on the risk matrix. The team then discusses the result and adjusts the post-it notes if needed.

Evaluate Security Measures

As the final step in DPF, the team evaluates the security measures for the data source. Starting with the data field with the highest risk, the team individually writes down some possible measures to mitigate the risk of using this data on post-it notes. The facilitator asks each participant to present their notes, and the post-it notes are collected and put up near the respective data field on the risk matrix (or in the table).

Outcome

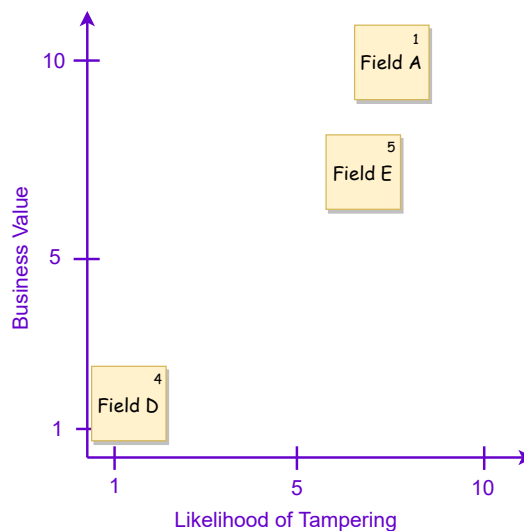
After participants identify all the relevant security measures, the team collectively documents the findings in their issue tracker/documentation.

ID	Business Value	Likelihood of Tampering
1	8	8
2	4	10
3	1	1
4	1	3
5	8	6

(a) Filled in participant answering sheet.

ID	Field Name	Data Type	Business Value	Likelihood of Tampering
1		<Data type>	10	7
2	Field B	<Data type>	5	10
3	Field C	<Data type>	1	3
4		<Data type>	2	1
5		<Data type>	7	7

(b) Filled-in white-board



(c) Half-done risk matrix

Figure 4.11: An example of how it could look when the templates are filled-in. Participants discuss the estimates of each member and decide the final value written on the whiteboard together.

4.5 Discussion Questions for Data Protection Fortification

In this section, we describe the questions asked in the data source discussion of DPF, and show an overview of these in Table 4.6. The questions are connected to the security properties described in Table 2.3 where relevant.

The initial questions were developed through discussions between the authors and supervisors and by considering the STRIDE categories to find potential characteristics that could increase the risk in data. These characteristics include: *Kind of data, data origin, data provider, data velocity, data volume, data variation, end-users, assets impacted, tampering consequences, threat actors, data validation efforts, data schema, encryption, data processing, sensitivity, fault mitigations, and connection to data source.*

4.5.1 General Questions About the Data Source

The general discussion questions ensure that the participants have a common understanding of the data source. They may also function as ice-breakers for the participants.

Q01 - What describes the data we get from this data source?

Related questions What does this data source contain?

Examples Weather data, gas prices, scooter info, customer reviews

Focus Common understanding of data source

Describing the data helps the participants gain a common understanding of which data source is being discussed, and gives a basic understanding to those in the team who might not be as familiar with it.

Q02 - Who/What provides this data source?

Related questions What is the name of the provider?

Examples Data supplier, yr.no, internal team, student, company-owned sensors

Focus Common understanding of data source, trust level

Whether the data comes from an external contractor, humans who use an application, or IoT sensors, it is good to be clear on where the data comes from. The team might find they have varying levels of trust or assumptions about the data depending on who supplies it.

Q03 - How do you receive the data?

Related questions Where in the target system is data imported?

Examples Fetching data from API, importing from a database, receiving directly from device, form input posted to backend

Focus Entry point, data flow

How the data flows into the target system is good to know in order to be aware of the possible attack surfaces, as well as being aware of where security measures like validation can be done. There might be more than one way that the team imports the source.

Q04 - How is the data from this data source used in your services or products now, or will be in the future?

Related questions How can users use the data directly or indirectly?

Examples Used in decision-making, machine learning model, aggregated and shown in dashboard, presented directly to end user

Focus Attack surface, exit point, data flow

Knowing the intended usage of the data is important to be able to reflect over some of the specific threats to the product.

Q05 - Who are the end-users for this data?

Examples Internal analysts, maintenance, customers, customer support, users of an application, management

Focus Common understanding of data source

Knowing who the end-users of the data are is essential to understand who and how many users would be adversely affected if this particular data source is tampered with.

Q06 - How often is data received from this data source?

Related questions Is the data cached, and if so, for how long?

Examples The data is downloaded by a CRON-job every night, pulled from the device every 5 min, fetched when a user triggers a service (e.g., clicks to view details)

Focus Common understanding of data source

If data is received every second, it may be a limiting factor for monitoring the whole source.

Q07 - What is the volume of the data received?

Examples Batches of 100 million rows, 15 data fields, 10 kilobytes

Focus Common understanding of data source

The volume of the data received may be a limiting factor for what part of the data source the monitoring system is capable of monitoring.

Q08 - Do you have an explicit schema or API-contract for data coming from this data source?

Related questions Could the values be “anything”?

Examples Swagger-API, System documentation

Focus Integrity, Common understanding of data source

Data sources coming from external sources should have a defined contract so that developers can look up the expected format and value types of the data source. If no such contracts are available, developers should request this from the data provider if possible.

Q09 - How is the data coming from this data source generated?

Related questions Is it known who puts the data into this data source?

Examples Human input through a webpage form, IoT device, system generated

Focus Threat actors, attack surface, entry point, trust in data

By knowing how the data is generated, one can assume who would be most likely to also tamper with it.

4.5.2 Security Implications for the Data Source

The following questions can spark discussion around aspects of the data source that can have security implications.

Q10 - How is data secured during transit?

Examples HTTPS, hash/MAC/digital signature verification

Focus Confidentiality, integrity, non-repudiation

This question target the potential of data tampering when the data is in transit due to the use of unsecured/weak protocols.

Q11 - Does any protections exist on the device to prevent physical tampering?

Related questions Can you detect physical tampering?

Examples Behind locked doors, requires a special key to open the device, hardened physical design, the device has an alarm, monitoring

Focus Integrity

For companies that get data from devices such as IoT, it may lower the risk of receiving tampered data if there are physical protections to make it harder for malicious actors to gain access to the device that sends data. Identifying devices that have been tampered with is important for operators to recover from attacks.

Q12 - Who has access to change the connection URL used to connect with the data source?

Related questions Where is this access URL stored? Is a change to this access URL logged?

Examples environment variable in Azure, configuration in database

Focus Authenticity, availability

Make the participants reflect on access control and the possibility for logging when the URL changes (to limit possible insider threats). An attacker could potentially spoof the data source by replacing the URL or affect the availability of the service by disrupting the connection.

Q13 - How sensitive is the data in this data source?

Examples Person Identifiable Information (PII), information about business opportunities/tenders, data that could affect stock prices, data that could affect decision-making processes

Focus Data sensitivity, confidentiality

Knowing the sensitivity of the data helps determine what an attacker may be most interested in.

Q14 - What could an attacker be interested in influencing through this data source?

Examples Analyst decisions, planned maintenance, conceal information or impact repudiation, company damage

Focus Misuse cases, threat actor goals

This question can help bring up specific cases, or end-goals of an attacker. It could also help participants find which of the data fields could influence certain parts of the system.

Q15 - What could the consequences be if the data source was no longer available or parts of the data were missing?

Related questions Are there certain times where the consequence would be greater?

Examples The service provided is rendered useless, users are denied access to a digital voting platform (but access to this platform is only critical during the days where voting is open)

Focus Availability, integrity

Understanding the consequences of a denial of service, or loss of integrity, is tightly coupled with the need for securing this data source.

Q16 - Who are the possible threat actors for this data source?

Examples Other nations, market competitors, disloyal employees or ex-employees, terrorists, script kids

Focus Threat actors

Knowing which threat actors are most likely to target this data source can help to understand which type of attacks are likely to come with the data, as threat actors may have very different end-goals for their attacks (from service disruption to advanced persistent threats).

Q17 - Would you discover if the data from this data source is incorrect, or if the data source is unavailable?

Related questions How would you report unavailability or inconsistencies to the suppliers of this data source?

Examples Service availability monitoring (health checks), users would discover unavailability or inconsistencies and report back to the team

Focus Fault mitigation

With this question, we are interested in routines for reporting anomalies or service unavailability. This question is more of a smoke test for participants to discuss how they handle such scenarios.

4.5.3 Evolution of Questions

The discussion questions and their ordering have changed along with the method. The wording of some of the questions was changed to increase clarity, and we merged some due to being perceived as duplicates. One question on the authentication method was removed due to not promoting any relevant answers and replaced with a question on securing the data during transit (Q10). One question on the volume of data (Q07) was initially in the method but forgotten after adapting the questions to the first session. One question was created based on answers from participants regarding a data source related to IoT data (Q11). The four questions that were added (Q07, Q10, Q11, Q16) have not been tested with participants but were added to cover more aspects of the data source. Some questions on how much the data values vary, and on validation and escaping data, were removed from the discussion after introducing the security measures toolbox since the questions were easier and more relevant to answer when talking about specific data fields.

Table 4.6: Questions for discussing the selected data source in Data Protection Fortification.

General Questions	
Q01	What describes the data we get from this data source?
Q02	Who/What provides this data source?
Q03	How do you receive the data?
Q04	How is the data from this data source used in your services or products now, or will be in the future?
Q05	Who are the end-users for this data?
Q06	How often is data received from this data source?
Q07	What is the volume of the data received?
Q08	Do you have an explicit schema or API-contract for data coming from this data source?
Q09	How is the data coming from this data source generated?
Questions for Security Implications	
Q10	How is data secured during transit?
Q11	Does any protections exist on the device to prevent physical tampering?
Q12	Who has access to change the connection URL used to connect with the data source?
Q13	How sensitive is the data in this data source?
Q14	What could an attacker be interested in influencing through this data source?
Q15	What could the consequences be if the data source was no longer available, or parts of the data was missing?
Q16	Who are the possible threat actors for this data source?
Q17	Would you discover if the data from this data source is incorrect, or if the data source is unavailable?

Chapter 5

Results

In this chapter, we present the results related to answering the research questions. First, we present practices related to the handling of data sources. Then, we present evaluation results for Data Protection Fortification (DPF). Finally, we present a conceptual architecture for a data monitoring system using audit hooks.

5.1 RQ1: What Are the Practices Reported by Companies for Securely Handling Data Coming From Devices or Other Sources?

In this section, we present the results from performing Data Protection Fortification (DPF) with the teams from the collaborating companies. The data sources we focused on in our DPF sessions were already integrated into their systems. Therefore the practices we found are derived from how the teams described their current handling of the data source as part of the data source discussions (the first step in DPF).









In Table 5.1 we summarize the findings with an evaluation of how the practices found can contribute to the risk of data tampering. If the practice reduces the risk of data tampering, we mark it with a green checkmark (). If the practice should be looked into because it might pose a risk in some cases, we mark it with an orange flag (). If the practice increases the risk, we mark it with a red flag (), and in cases where the practice does not affect the risk or where it can not be determined, we mark it with a gray icon ().

Table 5.1: Categories and practices reported by the development teams.  = Good practice,  = Should be followed up,  = Bad practice,  = Does not affect risk





















































Category	Practice	Evaluation
Data Access	Part of team can edit connection URL	
Data Access	Editing connection URL in cloud config is audited	
Data Access	Connection URL is placed in environment variable	
Data Access	Whole team can edit connection URL	
Data Access	Editing connection URL is audited in Git history	
Data Access	Connection URL is placed in database	
Data Access	Editing connection URL is not logged	
Data Access	Connection URL is placed directly in code	
Data Validity	Prepared Statements (SQL) on retrieval	
Data Validity	Whitelisting values	
Data Validity	Constraints defined in database views	
Data Validity	No sanitization on storage	
Data Validity	Only validation of phone number in frontend	
Data Validity	Rely on database queries to fail for data validation	
Data Validity	No validation	
Data Velocity	Receives data continuously	
Data Velocity	Receives new data seldom (1 time a year)	
Data Velocity	Receives new data every 5-15 minutes	
Data Velocity	When queried by user	
Data Origin	System-generated data	
Data Origin	Internally generated	
Data Origin	Externally generated	
Data Origin	Human input	
Data Origin	IoT-device data	
Data Processing	Processed by framework with schema validation	
Data Processing	Front-end framework with injection protection	
Data Processing	Normalize data fields to fit standard	
Data Processing	Exported to CSV	

Table 5.1: Practices reported by the development teams.

Icons used:  = Good practice,  = Should be followed up,  = Bad practice,  = Does not affect risk (continued).

Category	Practice	Evaluation
Data Schema	Explicit schema with API contract	
Data Schema	Schema defined through models in framework	
Data Schema	Notified if schema changes happen externally	
Data Schema	Text data type mostly used (even on numbers, dates)	
Data Schema	Must look in database for schema definition	
Data Usage	Aggregated and presented	
Data Usage	Presented directly (raw format)	
Data Usage	Used in machine learning	
Data Usage	Used in device decisions	
Fault Mitigations	Notified if source is unavailable	
Fault Mitigations	Internal lookup point for contact persons	
Fault Mitigations	Logging data that goes into ML model	
Fault Mitigations	Correlate data against internal system	
Fault Mitigations	Must find and contact the right person	
Fault Mitigations	External users report problems	
Fault Mitigations	Manually check data against source	
Fault Mitigations	Manual fault detection	
Fault Mitigations	Operations monitored manually	
Fault Mitigations	No routines to detect faults or handle it	
Fault Mitigations	No backup due to sensitivity concerns	

We add more details to the categories and reported practices we found in the following sections.

5.1.1 Data Access

We encountered a few different strategies in the teams for handling the connection URL to the external data source. We look at three aspects of this: where the connection string is set, who has access to change it, and if these changes are logged. Figure 5.1 shows an overview of the responses from the teams, mapped to these three aspects.

In one case, the team checked the connection URL into Git with the code, while the credentials needed to access it were stored in a managed identity solution provided by the cloud provider where the application was hosted. All developers had access to change it, but the changes are logged through the commit history in Git. Another team mentioned that the connection URL was committed to the Git-repository, which the entire extended team had access to edit, including a project leader who was no longer on the project.

In another case, the team set the connection string in a database, which the entire team had access to change, but any changes to this database were not logged.

One team that stored the URL in Git for one of their applications and in environment settings in Azure for another, reflected on how long it could take before any malicious changes to the URL were detected in the two cases. They mentioned it was likely to be discovered faster when it was checked into Git since everyone on the team had access to check it, and it was bundled together with the code. Whereas only a few people had access to the Azure settings (which were rarely checked), it might take longer to discover. The team also believed the cloud provider had some audit logging but that it might not be easily accessible by the team.

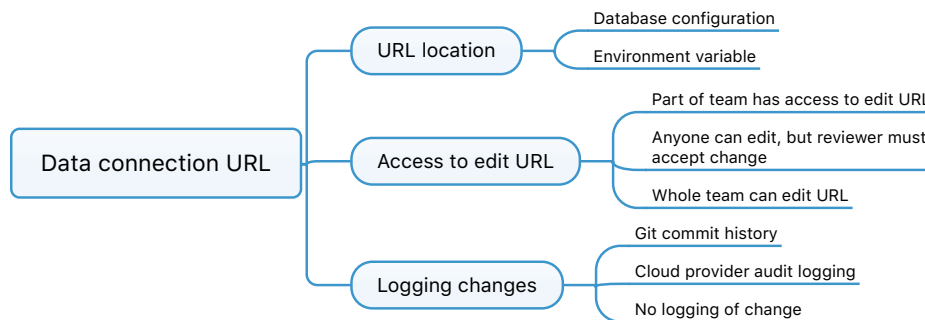


Figure 5.1: Management of connection URL to the data source as reported by the teams.

5.1.2 Data Validity and Monitoring

Three out of five teams reported having implemented some measure for ensuring the integrity of the data they receive. In Figure 5.2, we categorize the validation efforts reported by the teams by syntactic validation, semantic validation and monitoring. We decided to include only the most significant practices reported in this section in Table 5.1 to limit cluttering and to reduce the length of the reported practices.

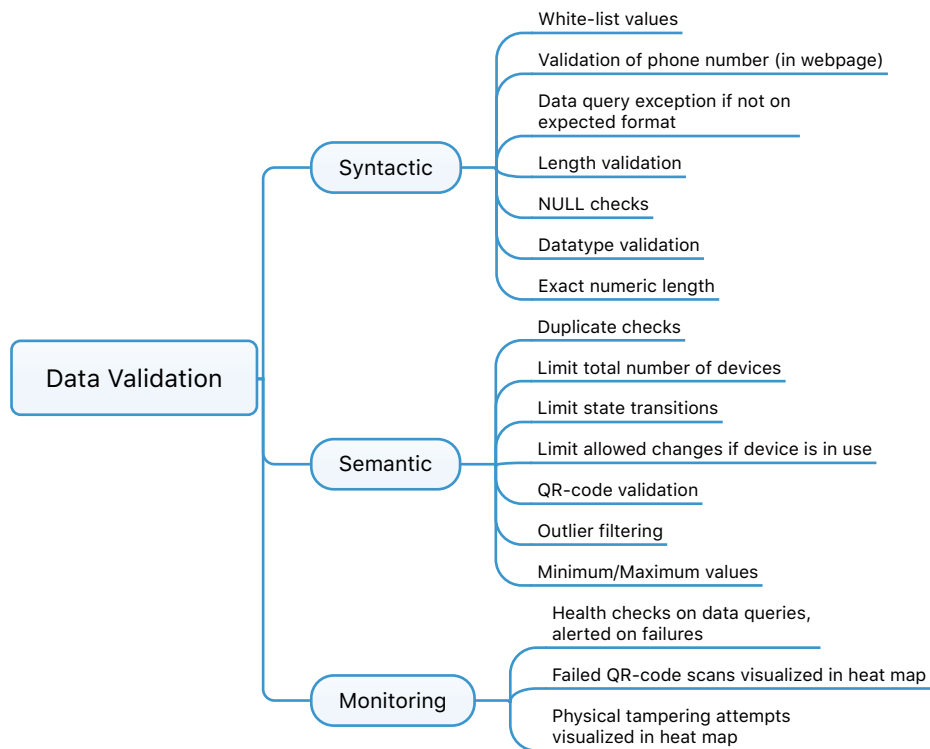


Figure 5.2: Validation and monitoring efforts reported by the teams.

Reasons for Missing Validation

One team had few expectations of what the external data source might have of values and only presented the raw values in a webpage to its users. They had previously had incidents where a field would be longer than expected and would distort the webpage layout. Despite this, no validation is done because they are uncertain of the contents of the data and would like to present it "as-is" to the users.

Another team knows the data source and its expected values but relies on the constraints defined in the database, governed by another team within the company, to ensure the integrity of their data. The queries run by the team will raise an exception if the data they receive does not have the expected data types. If such failures happen, they are notified and can change the queries accordingly.

Monitoring

One team reported having implemented various monitoring efforts to detect physical tampering with their devices. One type of monitoring is implemented to surveil QR-code scannings done by users, where failed scans are sent for auditing, along

with where and when the scan happened. The logs are visualized in a geographical heat map to allow customer service to monitor if QR-code scans fail in a particular city or district. They also monitor for physical tampering with the devices, where each device is equipped with a gyroscope to detect any movement when the device is not in use. Reported tampering attempts are visualized on a map for customer service, who may send personnel to that area to investigate further.

Another team reported on implementing "health checks" to monitor if the external data source was responding or not. The team would then be notified if the data source was unavailable so that they could take action if this occurred.

5.1.3 Data Velocity

For data velocity, we found a few practices relating to how often data is received (or sent) from the data source. We found the usage of scheduled tasks (CRON-jobs) for retrieving new data from the source in two cases. The time scheduled for when these tasks should fetch new data varied significantly, ranging from once every 5-15 minutes to once every year since the data source does not update often. One team reported receiving new data continuously from their data source, and another only queried data from the data source on user interactions.

5.1.4 Data Origin

Though this category does not involve practices, it was found that various sources generated the values of data fields within a data source. We identified three characteristics for the origin of a given data field. First, the origin of the data in a data field could be system-generated, meaning that the data was generated within the boundaries of a system. Typically, this includes timestamps used in date fields and the generation of unique identifiers. Second, the data could be generated from human input, typically from an input field. Last, the data could also come from IoT devices, such as sensor readings (e.g., percentage of battery remaining on device).

In addition, we add two more characteristics to the data origin, where we consider the (physical) location of where the data is created. If the data is generated outside the boundaries of the company, we characterize it as *externally generated*. Alternatively, if the data is generated within the boundaries of the company, we characterize it as *internally generated*.

5.1.5 Data Processing

For data processing, we are interested in how the teams process the data from the data source. Most of the teams use language-specific frameworks for handling data storage and presentation of the data. In popular JavaScript-frameworks used for creating websites, such as Vue and React, security measures for preventing injections are built-in. However, besides providing a solution for users to view the data on a web page, one team reported that their end-users prefer to process the

data in Microsoft Excel, which was identified as another attack surface. Therefore, the data is also made available for export to CSV format. Regarding data storing, one team reported that they tried to normalize the data prior to storing, for instance, by ensuring that all prices coming from the data source are given in NOK.

5.1.6 Data Schema

Figure 5.3 shows an overview of the responses from the teams related to the use of data schema to ensure the correctness of the data. The teams with an API contract for the data source have defined an explicit business contract with an external company, where the data format is formalized. Data sources from internals may not have the same defined API contracts, and as a result, they have to look up the schema in the database that is provided to them. One team had little knowledge of what the data fields contained and therefore treated all data fields as text values.

Related to the data schema is also whether the team is notified or not when the data schema changes in a data source. In one case, we found that they had an API contract that strictly required the data provider to notify them before any changes were made. In another case, the team said they would not be notified when changes occurred and would have to discover it themselves when things did not work anymore.

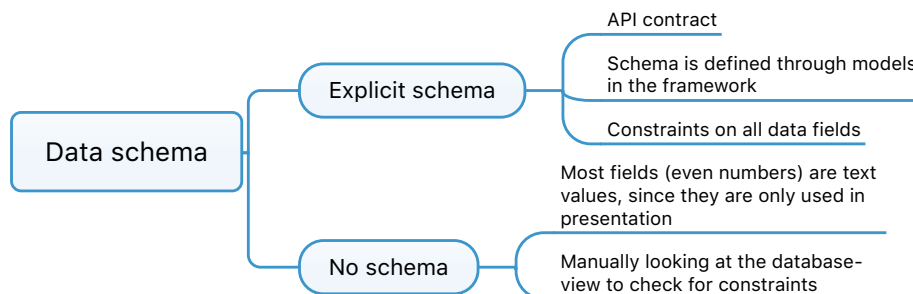


Figure 5.3: Practices related to data schema reported by the teams.

5.1.7 Data Usage

For data usage, we looked at how the teams use the data from the data source. One team did not process the data due to sensitivity concerns and presented it directly to the users. One data source had different use cases, where it was used in machine learning in one application and aggregated and presented to a user through a dashboard in another. Another data source was used by IoT devices to make decisions about how the device should function.

5.1.8 Fault Mitigations

With fault mitigation, we are interested in the team’s ability to detect and respond to unavailability, inconsistencies, or missing data in the data source. The reported practices for detecting this varied greatly, from having an automated detection process, to manually having to check for faults in the data they receive.

Related to how the teams would respond to data incidents or unavailability, we found that in one team, they can look up the correct person to contact in a *who-is* documentation in their knowledge management system. It was also reported that one team had no routines for detecting or handling faults.

5.2 RQ2: How Can Data-centric Threat Modeling Support Teams in Identifying Security Risks and Promote Secure Design in Handling Data?

This section presents the results from evaluating Data Protection Fortification (DPF) described in Chapter 4, including participants’ feedback, observations, and lessons learned from facilitating, as well as evaluations with two security experts. We facilitated in total 5 DPF sessions (in Norwegian); one for each of the participant teams presented in Section 3.3. In order to both accommodate the teams’ way of working, and to try out different variations of the activity to learn what works best, we held the activity in three ways: (1) fully remotely through Microsoft Teams, (2) co-located using digital tools, and (3) co-located (with one remote member) using physical tools. We facilitated 3, 1, and 1 session of these, respectively, and the method was improved iteratively.

Table 5.2: Overview of Data Protection Fortification sessions held.

ID	Team (# participants)	Location	Tools Used	Duration
S1	AR/VR Team (5)	Digital	Mentimeter	1,5h
S2	Eureka ML Team (3)	Digital	Mentimeter	1,5h
S3	Atlas Team (5)	Physical	Whiteboard, pen and paper	1h
S4	Webkom (9)	Physical	Mentimeter, Miro	1,5h
S5	Ryde (2)	Digital	Mentimeter, Miro	2h

In S2, the team invited an additional participant from outside their team. This person had expert knowledge of the data source being discussed and who used their platform and the data to develop a dashboard for traders.

In S3, the team invited three participants from outside their team who had expert knowledge of the data source being discussed. Two of these were from

a team that used their platform to create a data product using the data source. Due to miscommunication, we had prepared a non-digital variant of the method while one person joined the meeting remotely. This mistake negatively affected the facilitation of the meeting. Furthermore, we did not clarify the agenda of the allocated meeting time, and we discovered that we had much less time to perform DPF than necessary, which led to the session being cut short.

The security measures toolbox had not yet been invented in the first three sessions (S1, S2, S3). Evaluating security measures was done by asking the team questions to spark some ideas.

5.2.1 TAM Evaluation of DPF

We did a structured evaluation of DPF with participants from Webkom (S4) and Ryde (S5). We asked the participants to evaluate the session based on the statements given in Table 3.1. Figure 5.4 shows a radar chart of the results, where the units indicate how strongly each team, on average, agreed to each statement (where the value 1 corresponds to "strongly disagree", and 5 corresponds to "strongly agree"). *Use of Mentimeter (PEU4)* and *Useful discussions (PU3)* received the highest scores on average (4.9 and 4.8), while *Fits team (PU5)* and *Increased competence (PU6)* received the lowest average scores (3.6 and 3.9). The average score for all statements was 4.5.

5.2.2 Evaluation of the Activities in DPF

For Webkom (S4) and Ryde (S5), we asked the participants in each session to rank which of the three activities in DPF they felt was the most useful. In total, 10 participants answered from both companies, and the results are summarized in Figure 5.5. *Evaluate Security Measures* and *Prioritize Data Fields* were perceived as almost equally useful, with only one vote in favor of the former. *Data Source Discussion* received significantly lower votes, where only one participant felt it was the most useful step.

5.2.3 Verbal Evaluation of DPF

As mentioned in Chapter 3, we collected verbal evaluations from every DPF session. This section is structured by first presenting general feedback and then feedback specific to each of the three parts of DPF (discussion, prioritizing, security measures). A reference to which of the sessions (S1-S5) the statement was given is added at the end of each statement. Similar statements have, in some cases, been combined to give a better narrative of the evaluation feedback.

General feedback

Duration of session Participants mentioned that the sessions could be longer (S1, S4) and that the duration could naturally vary based on the data source in

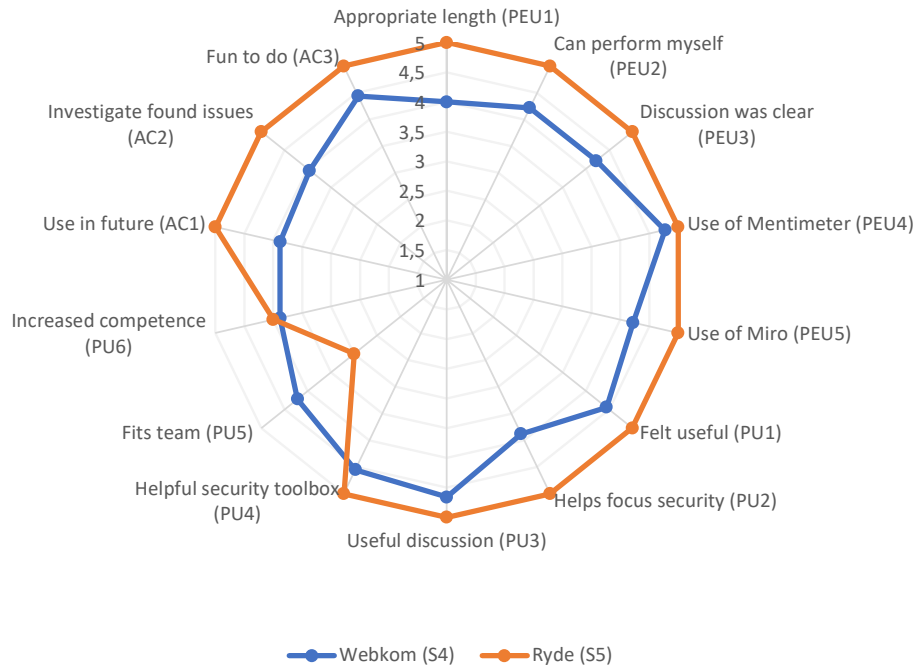


Figure 5.4: TAM evaluation results from Webkom (S4) and Ryde (S5).

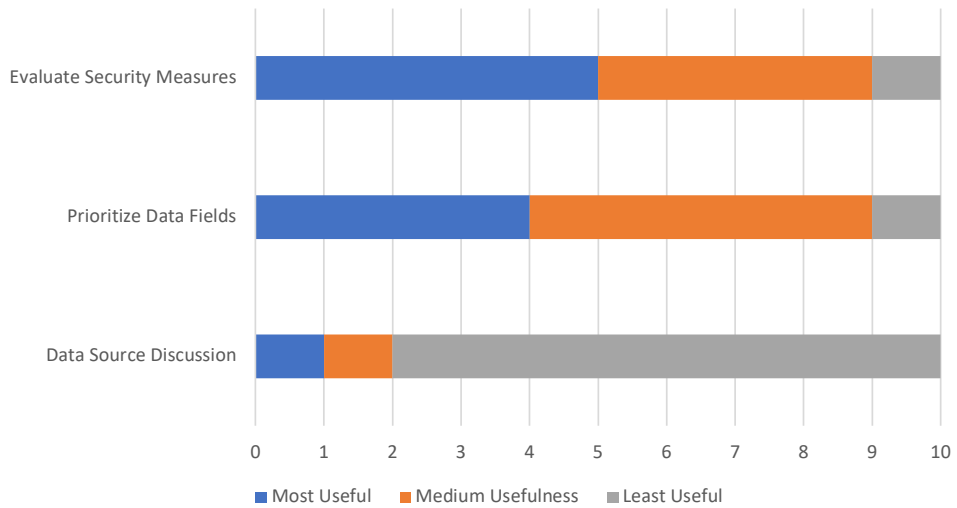


Figure 5.5: Participants evaluation of which part in Data Protection Fortification they found most useful.

question (S1), with more complex or more important data sources taking longer.

Participants Including users in the DPF session is a good idea (S2, S3). Furthermore, the session would have higher relevance by including participants from different end-points of the data flow chain to get the highest coverage (S2). By knowing more about the details of the data source, one could also find more things about the data source to consider (S5).

Ease of use The order of DPF made sense since it was a natural progression to start with open discussions and then look at more specific things to consider (S4). Some mentioned that the approach was straightforward and easy to follow (S2) and that a development team would be able to perform it on their own (S5). Still, it was practical to have someone who knows the method to facilitate the session (S4) and watch the time during discussions (S1). They liked that the session did not leave them exhausted, as it kept the team engaged and in control, in contrast to the "wall-of-text" kind of meetings they were familiar with for assessing compliance (S2).

Usefulness The method was *surprisingly* useful (S1) and enabled good reflections on both the security (S1, S5) and possible weaknesses (S2) of the data source. Focusing on data fields is helpful for developers since it is concrete and relatable (S4). It is also very useful to look at the data fields from a security perspective (S4). As for the outcome, it highlighted that the team lacks knowledge of this data source (S1); it raised issues (S4) and questions that need to be clarified with the development team (S5). It does, however, not exclude other potential security flaws (S4). One participant commented that the discussion was more concerned with what one already knows about the data source, while the prioritization and security measures part were more specific and more useful since it was more actionable (S4).

Acceptance The method is a fun (S4), straightforward approach (S2, S5). It has the potential to work for more data fields in more complex systems (S4) and in other contexts as well (S2). The session can be adapted based on the data source in focus (S1). It is also possible to perform for non-developers (S5).

Data Source Discussion

The discussion went well (S3), and the discussion questions felt relevant (S2) and sparked interesting discussions (S1). Some thought the discussion felt more like an interview between the facilitators and the group. In contrast, one participant in the same group said there could have been even more discussion questions (S4). Some questions were also perceived as ambiguous, which influenced the discussions (S4). For one group, a few questions did not fit the specific data source, but they reflected that the same questions would be relevant for other kinds of

data sources (S2). An extended discussion could be particularly good for the first session on a data source (S1).

Prioritization of Data Fields

The prioritization part using Mentimeter worked well (S1). For those familiar with traditional risk assessment, the generated risk matrix is familiar (S5) and is a good visualization of the estimation results (S2). During estimation, it was good that incoming answers were hidden while the others voted (S4), but when everyone had finished voting, it was interesting to see the distribution of votes (S1). A challenge with estimating is to prioritize when the data source only contains critical fields (S3). Another example of this is that for a platform provider, it is important that all data fields contain correct values (S2). There is more potential in the digital variant of DPF when estimating (S3). This step also requires in-depth knowledge of the data field's business value, which end-users know best (S2). This makes the validity of these estimates questionable since (most) developers are far away from the business side (S5). There would be a need for more discussion to make final estimates more accurate and an opportunity to adjust estimates put into the risk matrix (S5).

Evaluating Security Measures

For the two teams who performed security measures evaluation in Miro, it is highlighted that doing this step is the most interesting (S5), and the toolbox with common security measures is helpful to use as a reference (S4). It was also mentioned that this step is an efficient way to share knowledge about what security measures the frameworks they use have (S4). Feedback from one team (when this step was only performed verbally without Miro) was that coming up with security measures for an API that they do not control was challenging (S1).

5.2.4 Observations and Lessons Learned from DPF

Table 5.3 presents lessons we have learned from observing and reflecting on the Data Protection Fortification (DPF) sessions we have facilitated.

Table 5.3: Observations and lessons learned from facilitating Data Protection Fortification

Step	ID	Topic	Our observations from DPF sessions
<i>General</i>	LL01	Engaged participants	We experienced that participants during the session looked up assumptions and security concerns that were brought up (S4).
<i>General</i>	LL02	Showing overview of activity as agenda	It was a good idea to show a high-level overview of the method at the start of the session to teach them the flow of the method (S1).
<i>General</i>	LL03	Need for time-boxing	Time-boxing the steps in DPF is good to show participants how much time they will spend in the various steps, and where they can flex on time (S1).
<i>General</i>	LL04	Importance of the preparation meeting	When the team selected the data source without a preparation meeting(S3), we experienced challenges when facilitating. The evaluation revealed that a preparation meeting might have uncovered challenges with using this data source as the focus for DPF.
<i>General</i>	LL05	Number of participants	Performing the method using digital tools went well with 5-9 participants (S1, S4), and it showed potential to be able to handle more participants. A smaller group is better when using physical tools, as prioritizing data fields and evaluating security measures took too long (S3).
<i>General</i>	LL06	Actionable steps are more useful	While the discussion was as well-received in the sessions (S1, S2, S3), and the discussion took up brought up many interesting issues (S5), it came at the bottom of the ranking when ordering the different steps in the method (S4, S5). This indicates that participants find the step with the most specific and actionable output more useful.
<i>Data Source Discussion</i>	LL07	Challenge to engage all participants	In one case, it was difficult to get all participants engaged in discussions (S4), and evaluation revealed that participants felt the discussion was like an interview for facilitators to learn about the data source, not the participants. While we did not experience this in other sessions, there was a trend that the most knowledgeable person also talked the most (S1).
<i>Data Source Discussion</i>	LL08	Discussion examples needed	Giving examples when asking the discussion questions helped participants answer (S1). However, when none of the examples were relevant to the data source, it could confuse them (S2).
<i>Data Source Discussion</i>	LL09	Questions need to be adapted for different data sources	When testing the method on data sources like form input (S4) and augmented device data (S5), not all of the questions made sense, and we had to adapt some of them during the session to get relevant answers.
<i>Data Source Discussion</i>	LL10	Do not assume, explain	Assuming that all of the participants are familiar with the data source in focus is a risky, and it helped the session to have someone from the group describe it first (S1).

Table 5.3: Observations and lessons learned from facilitating DPF (continued).

Step	ID	Topic	Lessons learned
<i>Prioritization of Data Fields</i>	LL11	Limitations of DPF	The participants' knowledge of and the number of data fields within a data source is very likely to impact the outcome and successful adoption of DPF.
<i>Prioritization of Data Fields</i>	LL12	Manageable number of data fields	We observed that during prioritization, estimating 20 data fields at once (directly in the risk matrix) was exhausting for participants (S5). 5-8 data fields were tested to be manageable.
<i>Prioritization of Data Fields</i>	LL13	Difficulties in prioritizing	Some participants struggled to find differences in business value among the data fields (S2, S3). The teams also found it difficult to estimate the consequence of tampered data when the data was used for a broad number of applications that they did not control.
<i>Prioritization of Data Fields</i>	LL14	Digital tools scale better	In comparison with only using physical tools for prioritization, using digital tools makes this step easier to facilitate, with regards to collecting responses and showing the risk matrix to participants (S3).
<i>Prioritization of Data Fields</i>	LL15	Explain data fields first	Having one participant explain the data fields and describe their use before estimating the business value helped gain a common understanding (S2). Showing the team example of values could also aid understanding (S1).
<i>Prioritization of Data Fields</i>	LL16	Purpose of calibration	While we originally intended to set min/max values for the data fields based on the calibration, as done in Protection Poker, we found this was not easy to enforce in Mentimeter (S1, S2), and revotes are also tricky. When the participants estimated values, they did not always use the calibration result (S2, S4) or the entire range of values and instead put the fields relative to each other (S4). Calibration results in Mentimeter also showed multiple winners when participants' opinions diverged (S4). Hence, it was not suitable to reach a consensus on one data field in each min/max range. While estimations could vary significantly, the groups still got a sufficient spread of values in the risk matrix and agreed on the resulting relative prioritization. We conclude that the important purpose of the calibration is to facilitate discussion in the group to aid the subsequent estimation.
<i>Evaluating Security Measures</i>	LL17	Need a structured process	Eliciting security measures without a structured process did not produce any valuable results (S1), and there was a need for better examples (S2). After structuring the process in Miro it worked very well (S4), and the risk matrix helped prioritize which fields to start this process with (S4).

5.2.5 Evaluating DPF with Security Experts

This section presents some feedback from two security experts on the potential of DPF. We address these statements as SE1 and SE2.

Overall potential We got feedback that DPF looked interesting and had good potential (SE1). Furthermore, one security expert thinks development teams would be able to do it on their own since it is structured and visual enough for them to be easy to do if instructed (SE2).

Suitable contexts for use In addition to using DPF for data integrations, it also shows potential for data sent from users through a web interface, such as form input data (SE2). As IoT and Industrial Control Systems (ICS) often lack integrity checks, including the widespread use of unencrypted protocols and weak security mechanisms, it could be valuable for developers of such systems (SE1). The structured discussion of DPF also showed the potential for data that is subject to GDPR regulations, to evaluate the consequences for people using a data system if the data is leaked or misused (SE1).

Number of data fields It can be difficult to do DPF on the whole data source, as many data sources can have hundreds of data fields. We give value by looking at security measures at a low level, so most of our toolbox is not applicable to use at a higher level (SE2).

More factors in risk In addition to estimating the risk of data fields by using business value and the likelihood of tampering, other factors could be relevant to consider (SE2). For instance, risk could be decreased for certain types of fields if a strongly typed programming language is used, or if the maturity of the data supplier organization is high and they do many checks on the data on their side. The risk also depends on what kind of attacks the data fields are vulnerable to (e.g., different impact if data is corrupted versus can contain an injection command which could lead to system takeover or information disclosure). Furthermore, the risk of using a data source can depend on how many links the data has to flow through to get to the system from the origin. If one is close to the source, one has more control over what an attacker is allowed to put into the data source. However, it may also be more visible to an attacker if an attack works. In addition, if one uses an API, one has no control over what can be put into the data and have to trust it.

Doubtful of the value of prioritization Prioritization of data fields might be difficult for developers to give good answers to (SE1). Estimating business value requires an understanding of how the data is used, which developers might not have if they are in a company where developers are far away from the business side. Estimating the likelihood of tampering was also mentioned as potentially difficult to estimate, particularly for teams that work on micro-services where they do not see the bigger attack surface.

Security measures The security measures made sense from a security perspective, and most were simple to understand (SE1, SE2). The discussions should be focused mainly on the domain-specific security measures, as these can not easily be identified by static code analysis tools (SE1). These tools often target good programming practices (e.g., validation of length or correct format) but can not discover missing business rules in code.

Using the outcome One expert highlighted the importance of telling teams how to use the outcome (SE1) and that putting the security measures into existing issues in a task board was preferred (given that the data source is not already implemented). This could help make product owners aware that the task is not done until the security measures are implemented, as well as prevent the issues from becoming forgotten at the bottom of the backlog.

Communication tool for security Working closely with developers is the most effective way to secure code. However, it can be difficult to get time from development teams to do extra security work due to feeling rushed (SE2). Often stakeholders assume that developers are already doing the necessary security work and are therefore not telling them to slow down on the features to improve security. Developers often want to add better security but often experience pressure from stakeholders to spend their time developing features (SE1). DPF was mentioned to have the potential to become a communication tool between developers and stakeholders about security, particularly the data source diagram, as it can help visualize whether they have the necessary protections in place (SE2). Suppose one could give developers some metric, for instance, how often the data coming from the data source fails validation. In that case, it could give them a strong argument to spend more effort validating the data on their side as data consumers (SE2).

5.3 RQ3: How Can Data Monitoring Using Audit Hooks Identify Attacks in Data With Increased Risk?

Due to time constraints, we were not able to not answer this research question. Instead, we provide a conceptual architecture of the data monitoring system that future research can consider.

In the previous section, we presented the results from performing DPF. One context where DPF can be helpful is determining the need to monitor specific data fields within a data source for potential attacks. Data fields identified as higher risk in DPF may be good candidates for monitoring. Figure 5.6 shows an example of how the security measure requirements from DPF can be connected to the data monitoring system. We discovered different monitoring possibilities from performing DPF with the teams and from discussing security experts, most of which we have conceptualized as *analysis capabilities* in this section.

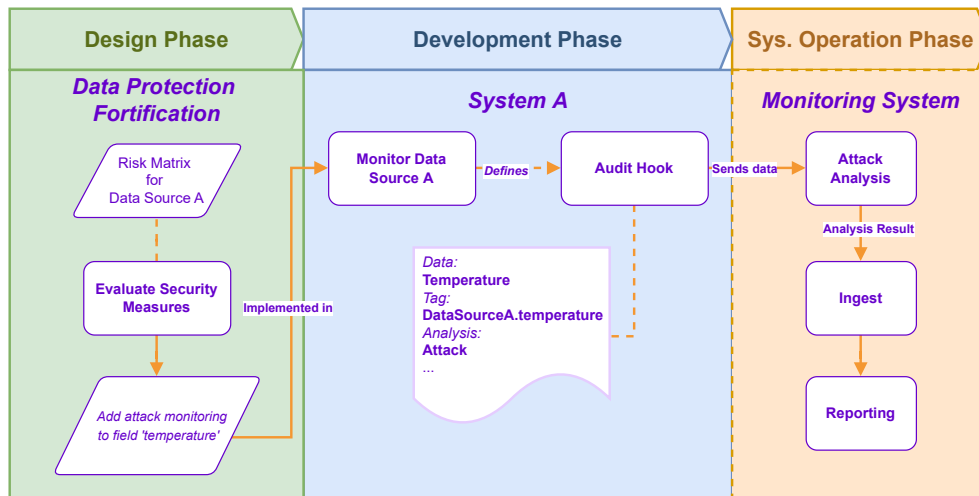


Figure 5.6: Example of how DPF is used to add a requirement for attack monitoring based on security measures evaluation during the design phase. In the development phase, the requirement is implemented as an audit hook placed in the code of System A, and the data it should send is defined. During system operation, the audit hook will send data to analysis in the monitoring system.

Message brokers are widely used in data-intensive systems due to their characteristics of high scalability, throughput, and availability [66]. Our proposed architecture for a monitoring system using audit hooks revolves around the Broker architectural pattern [66] for distributing messages between the components using a lightweight message broker. The considerations done for selecting this architecture are listed below:

- We expect high throughput of data coming into the system.
- The message flow is relatively simple, with no need for central orchestration or a transactional state. Different types of analysis can be done in isolation and do not depend on each other.
- We want to add new monitoring capabilities in the future.

As illustrated in Figure 5.7, a message sent from the *Audit Hook* component includes the data we want to monitor for attacks. Messages are sent through a *Message Broker* with a given *Routing Key (RK)*. The RK tells the message broker where to send this message. The destination of messages are either *Message Queues* or *Topics*, which have a key (name) defined for receiving messages. The broker will route messages to a topic or a message queue if the RK matches the queue's name. These names can be explicit or have an asterisk (*) for pattern-matching. Topics can be used when multiple components should receive the same message. This is the case for the analysis capabilities, where we want all capabilities to receive a copy of the message. Message queues are used when we only want one component to receive the message, as is the case with the *Ingest* component.

In the following sections, we give an overview of each component in the archi-

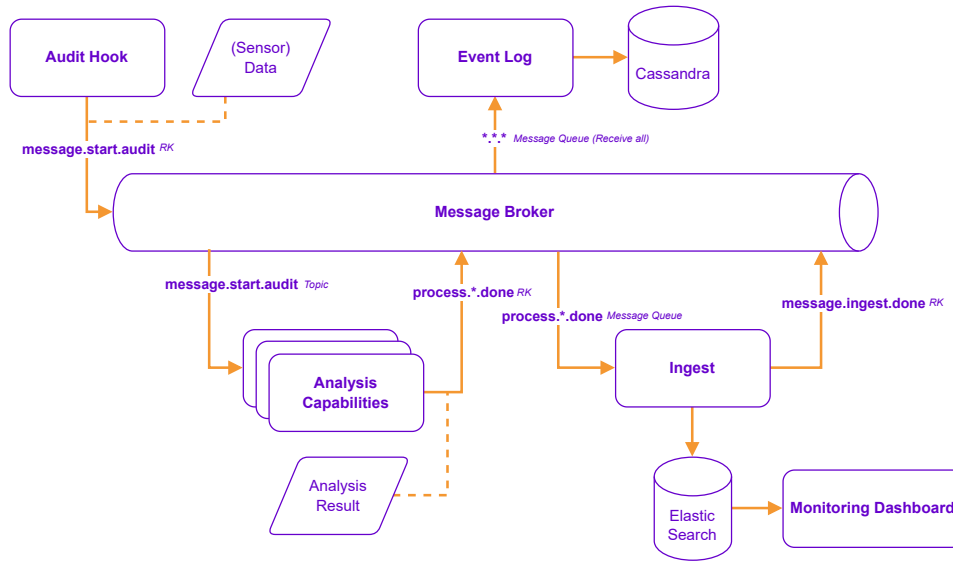


Figure 5.7: Proposed architecture for a monitoring system using Audit Hooks as the primary entry-point for data. The directional arrows between the message broker and the components are labelled with the *RK* (when sending) or the *topic/message queue* name (for receiving).

ture. The architecture is presented only as a minimum viable product (MVP) for analyzing data sent from audit hooks and has not been tested in practice.

5.3.1 Audit Hook

Audit hooks are interfaces placed in code for communicating with a tool-agnostic message broker (e.g., RabbitMQ¹) using the Advanced Message Queuing Protocol (AMQP). The format of the messages transmitted from an audit hook is specified using a schema. The data values sent into an audit hook are not bound to a specific type and could be semantically or syntactically invalid from a business perspective. In addition to sending the data value, it should be possible to add the following metadata:

- *type* - The type of data this data value is representing (e.g., currency, temperature, or humidity)
- *analysis* - An optional list used to specify which analysis should be performed
- *datetime* - When this data value was generated
- *tag* - Used to identify where this data value originated from (for instance, a combination of the data source name and the data field name)

The *datetime* and *tag* metadata is valuable for enabling value correlation, where data values matching a specific tag and datetime can be correlated.

¹<https://www.rabbitmq.com/>

5.3.2 Analysis Capabilities

Capabilities refer to the components in the monitoring system performing the analysis of messages. From the DPF sessions we performed with the companies, we identified four possible monitoring capabilities. These capabilities are described on a conceptual level in this chapter. Each of these capabilities should receive messages sent from an audit hook and then determine if it can do an analysis based on the message received. We propose including the *type(s)* of analysis wanted as part of the message body or header sent from the audit hook component. The messages sent from a capability to ingest should include the type of analysis performed (e.g., as part of the routing key), the original data sent, and the outcome of the analysis.

Value correlation For value correlation, we are interested in matching the data value sent with a previously monitored value. The analysis outcome should indicate this if the values do not match up. An open question with the current architecture is how this analysis could correlate the values to those of a different data set.

Anomaly With anomaly detection, we are interested in discovering unexpected shifts in the values received over a specific time period. An open question in the current architecture is how anomaly detection could be configured differently for different types of values. Some values change with clear patterns, whereas some values seldom or never should change. As we covered in Section 2.6.2, existing commercial products such as DataDog have included in their documentation which type of algorithms they use for monitoring, which researchers can use to gain insight into promising anomaly detection algorithms. One could also monitor metrics related to the data, as uncovered in one DPF session. This team could suspect an issue, which might or might not be due to an attack, if they received no data for an extended time within a specific period.

Attack The data values in a message are matched against a list of known attack patterns and attack signatures, for instance, using information from CAPEC ². One security expert suggested looking for particular patterns in value changes, tailored to a specific attack over a certain period. Some data fields may be tied to specific use cases, which can help create scenarios for possible attacks that should be monitored.

Error Error monitoring was inspired by the case of failed QR code scans described in Section 5.1.2. Error events can be created when data fails validation, which can then be sent for monitoring as part of handling the error in code. Failures of integrity checks could be mapped to specific attacks. For logging errors

²<https://capec.mitre.org/>

as part of error handling, one security expert mentioned that developers need to create understandable logs. Log analysis is often outsourced to separate security teams, who do not know the data or its use context within the application. By monitoring the data which generates validation errors, practitioners can detect tampering attempts that would otherwise only be visible in logs as runtime exceptions.

5.3.3 Ingest

This component is responsible for storing analysis messages into the system used as primary storage, Elasticsearch. It should also send a message to the event log after storing the message to indicate that this message has been successfully ingested.

5.3.4 Monitoring Dashboard

For presenting the monitoring results, we recommend using Kibana to connect with Elasticsearch. Kibana is a web interface for generating dashboards and analytics, primarily using Elasticsearch as a source for data. Indexes in Elasticsearch can be created using Kibana to group messages from the various analysis capabilities. These indexes can then be used to generate dashboards in Kibana to show the results from each analysis capability. Threshold alerts can be configured in Kibana to notify system operators if an analysis capability has found a positive result.

Event Log

The event log component is responsible for storing all messages as events. This is particularly useful for developers and maintainers of the monitoring system in cases when faults have occurred. It can also be used to trace messages being sent through the monitoring system. As we expect the event log to receive a considerable amount of data generated from the message queues, we recommend storing event messages in a database with highly efficient writes, such as Cassandra³.

³https://cassandra.apache.org/_/index.html

Chapter 6

Discussion

In this chapter, we discuss the results from Chapter 5 with regards to implications for research and practice. Finally, we discuss limitations to the method and study and conclude with lessons learned from conducting this research project.

6.1 Implications to Research

In this section, we present how the findings in this thesis relate to researchers interested in using the results and give directions for extending our work.

Table 6.1 show how DPF contributes to, or inhibits, participants experiencing flow (see Section 2.3.8) using the elements that contributes to flow given by Shostack [21]. From this, we take that many of the steps in DPF contributes positively to participants experiencing flow, while further work is needed to address some elements, especially related to *Clear goals* and *Direct and immediate feedback*.

Table 6.1: Experiencing flow in Data Protection Fortification. The aspects marked with an asterisk (*) are mentioned by Shostack [21] as key aspects for testing threat modeling approaches.

Aspect	Achievement level in DPF
Absorbed in activity	From evaluations, we learned that participants did not feel like the session was too long. In DPF, the participants are always the active party, and there are few periods of inactivity. However, the behavior of others during the session may negatively impact absorption, for instance, if one participant has to leave before the session ends.
Loss of self-consciousness	Including participants with different roles and security knowledge might make participants more self-conscious about their own performance, contributing negatively to this aspect. Using digital tools that support anonymity may contribute positively, as participants can vote anonymously when estimating values for business value and tampering. Anonymity could also be supported in digital tools for collaborating on evaluating security measures.

Table 6.1: Experiencing flow in Data Protection Fortification (continued).

Aspect	Achieved in DPF
Personal control over activity	Participants have control over their votes and may simultaneously add security measures in Miro. One team said they liked that they felt the team was in control (S2). Participants may choose to use the toolbox for security measures only as inspiration and can skip discussion questions that do not fit.
Clear goals*	DPF has clear internal steps. A diagram of DPF can be shown at the start of the session to give an overview to participants new to the method. As discussed in future work, clear end-criteria is missing from DPF, making it difficult to know when to end the evaluation of security measures and conclude the session itself.
Concentration and focusing	DPF requires participants to only allow a select range of information into awareness to concentrate on the different steps in the method, as they are not presented with too much information at once. The method grabs the participants' attention from the beginning and maintains their focus throughout.
Direct and immediate feedback*	When estimating in Mentimeter, participants receive feedback by seeing the voting results and the resulting risk matrix. For progression, the discussion is split into two parts with equally many questions, making the participants aware that they are halfway through after answering the first part. However, not all steps of DPF give direct and immediate feedback. Future work can look into how to give participants feedback while evaluating security measures. For instance, by giving recommendations on how many security measures should be found to be "good enough" or recommending security measures that go well together.
Balance between ability and challenge*	In DPF, the activity gradually becomes more challenging, but we provide examples in every step to aid participants if needed. In addition, the mechanisms of prioritizing and eliciting security measures are simple. In future work, different variants of the method can be considered based on the participants' level of knowledge about the data source.

In Table 6.2, we show how Data Protection Fortification (DPF) addresses some of the challenges observed by Cruzes *et al.* [67] in agile development projects using the Microsoft Threat Modeling Technique with STRIDE. Even though this is a software-centric threat modeling approach, as opposed to DPF, which is a data-centric approach focused on the data fields, many of the challenges reported are relevant to discuss for DPF.

Table 6.2: How Data Protection Fortification address challenges found by Cruzes *et al.* [67] in doing Microsoft threat modeling with STRIDE in agile development projects. The ID references the original paper. Challenges not applicable to discuss for DPF have been omitted. Lessons learned from Table 5.3 are referenced where relevant.

ID	Challenges reported in [67]	Relevance to DPF
C02	Many discussions on threats and mitigations strategies get lost	If teams do not take notes as recommended from the discussion and prioritization steps, this is also likely to happen in DPF.
C03	It is challenging to motivate the teams to draw the diagrams	Evaluations say it was fun to do DPF, and the last step where participants draw the data source diagram was found to be most useful, which suggests it might not be hard to motivate the team. In addition, cognitive load when drawing the diagram is low because there are few elements to consider and few opportunities to make mistakes.
C04	It was hard to decide the right level of abstraction to the DFDs	DPF does not draw DFDs, but instead a data source diagram with security measures, where the abstraction level is pre-defined to be on the data fields, which naturally limits the scope of the session.
C05	It takes a long time to draw the diagrams	How long it takes to draw the data source diagram depends on the number of data fields the participants consider. We provide a generic template, but it needs to be prepared in advance not to steal too much time from the session. Compared to drawing DFDs, where multiple types of components exist, DPF only actively uses security mechanism components.
C07	The approach does not make a link with the actual code	In future work, DPF could potentially generate a data source diagram based on annotations on validations in the code. One could also create a domain-specific language (DSL) for drawing the security measures that can be used to generate validation code for selected programming languages.
C08	It is challenging to maintain the DFDs	DPF does not use DFDs, but faces the same challenge for maintaining the data source diagram. Improving this is discussed as future work.
C09	The meeting needs to be structured, but it is not always clear on how to run the meeting	DPF provides a clear structure for all three steps of the method. Improving the structure of the last step is related to LL17.
C10	It is hard to decide which other people should be included in the meetings besides the "core" development team	DPF provides recommendations on whom to include for making the session the most relevant.
C12	There are challenges with running meetings in distributed settings	When using digital tools to run DPF sessions, we did not experience challenges in either distributed or co-located settings.

Table 6.2: How DPF relates to challenges in doing Microsoft threat modeling with STRIDE (continued).

ID	Challenges reported in [67]	Relevance to DPF
C13	It is hard to know when enough analysis has been done	This is related to "clear goals" in flow. We have not designed a way for practitioners to know what is considered "good enough" when evaluating security measures. This is recommended as future work.
C14	The meetings are not effective	The perceived effectiveness may come from the participants' ability to experience flow. We argue that for this, the aspects of "absorbed in the activity," "clear goals," and "personal control over activity" have an impact on how effective the meeting is perceived. From lessons learned, we observed the need for time-boxing (LL03).
C15	There is a need for a Security Expert to run the meeting	Evaluations of DPF suggest that the method provides enough examples and structure not to require a security expert to facilitate the meeting. However, this needs to be confirmed in future studies.
C16	It is not easy to have everyone participating	When participants had insufficient knowledge about the data source, there was a tendency toward the more knowledgeable talking the most in the data source discussion (LL07). Overall, we found participant contribution to be rather even.
C18	The output of the sessions is a list of concerns/threats that are not concrete	The main output from DPF are concrete security measures for each data field that should be added to existing or new issues in the development backlog. Eliciting security measures were also perceived as more useful than the first steps in the method, which have less concrete output (LL06).

Low-level data-centric threat modeling Developers had a positive attitude towards focusing the session on fortifying the security of data fields in a data source, as they were more recognizable to them. Therefore, we recommend that future research continues developing threat-modeling methods with a low granularity and a focus on data.

Improvements to produce flow Further research should find ways to provide better immediate and direct feedback to participants, especially for evaluating if the diagram created for security measures is sufficient. Furthermore, DPF needs criteria for when to conclude the session to provide clear, attainable goals to the participants. Shostack [21] recommends two testable states that must be fulfilled to conclude a threat modeling session; this includes having filed bugs and that you have "a diagram that everyone agrees represents the system." Lastly, studies on how to customize the method to match participants with various security skills or domain knowledge could make the method relevant to a broader audience.

Improve risk calculation The method of estimating the risk of data fields is based on the business value and the likelihood of tampering. However, as mentioned by one of the security experts, other factors could also affect the risk. Further research could investigate the potential of including other factors that affect the risk, which can help improve the current risk calculation.

We recommend further research on improving the scale used to prioritize data fields. During the estimation of risk, DPF uses a number scale (1-10) inspired by SINTEF's version of Protection Poker[33]. However, other scales could also be considered, and NIST advocates that differentiation of risk should be limited to High, Medium, and Low [18]. In addition, some participants reported that they used the scale differently during estimation.

Investigate need for prioritization Further research can investigate in which contexts prioritization is needed. When discussing the value of prioritization with a security expert, the value of performing estimation itself was debatable, as it can be difficult for developers to answer the questions used to prioritize the data fields. In some session evaluations, some participating teams also reported estimating business values as a concern. However, from the evaluation of the activities, the estimation step was perceived as almost as useful as evaluating security measures. Our observations also show that the teams appeared to understand the concept of likelihood of tampering and were able to find consensus when estimating this, even for the cases where they were not as familiar with the data. However, this does not mean that their accuracy is good.

In some cases, it might be viable to skip the prioritization step. For instance, when the team has enough resources to implement security measures for all the fields and has no computational restraints (which is not often the case for IoT). In this case, it would be more important to discuss the data fields in depth during the discussion, as this would typically happen during the prioritization. For other cases, the team would need to prioritize their security efforts, covering more of the lower-priority data fields over time. Future work could find ways to structure the discussion that happens naturally during prioritization to use this for prioritizing where to start when evaluating security measures instead of obtaining a risk matrix.

Extending the security measures diagram Further research is needed to find an appropriate way to make the security measure diagram more maintainable and closer to the implementation code. For this, a structured format such as JSON or Markdown could be used, which could be checked into their version control software. This lets teams represent it in their own space in their code repository instead of in a different tool. JSON could be useful for generating code such as test cases from some security measures, while Markdown provides typing options for enhanced readability. Another approach is to create a domain-specific language for defining the security measures, which can be used to generate validation code for selected programming languages.

Generating the security measures diagram Another idea is to find ways to help the team understand what input validations are already present by generating the diagram based on annotations in code. Abi-Antoun and Barnes [68] has done similar work for generating DFD diagrams for threat modeling based on code annotations that is read by a static code analysis tool.

Evaluate how the outcome is used Another possible direction for research is to assess how development teams act on the issues found after performing DPF. Following up on the security measures identified through DPF could face the same challenges as described by Bernsmed *et al.* [29], where issues often end up being ignored or disappear from the taskboard without being satisfactorily resolved. In turn, this makes the issue a part of the accumulated accepted risk of the product.

Developing a data monitoring system Due to time constraints, we did not evaluate the data monitoring system in practice. Further research is needed to develop this concept further to test the viability of using audit hooks to detect data tampering attacks coming through the data.

Regarding the goal of successfully identifying threats with such monitoring capabilities, Shostack [21] points out that anomaly detection is difficult to get right, as changes in business, or changes in values based on natural occurring changes, can lead to "normal abnormalities". One security expert echoes this by commenting that anomaly detection using machine learning generates too many false positives. Abnormal sensor readings do not necessarily stem from data tampering attacks but could also be due to natural failures of sensors. Compromised sensor nodes might also not always send malicious or false data (due to the attackers' attempts at concealing such attacks), but will be more likely to do so than non-compromised nodes [5].

Discussions with one security expert about monitoring highlighted the importance of knowing which fields are affected when an adverse event occurs. Knowing what to monitor in the data can be identified using the threat model to correlate events to specific data flows. This would then form the requirements for monitoring the data fields to look for values that correlate with an adversary event. Espenschied and Gunn [69] propose a method for detecting threats by generating models of state transitions over time, or "threat sequences". This model aims to enable decision-makers to detect adversarial sequences and gain insight after incidents by looking at the sequence of actions taken. As Shostack [21] points out, the generated threat sequence model can also be helpful to determine which data sources would detect specific state transitions (and thus, could be sent using audit hooks for event-value correlation monitoring).

Regarding attack monitoring, detecting attacks using attack signatures has been difficult since attack tools have adopted techniques to conceal attacks, for instance, by using polymorphism [21]. A security expert proposed to try to map failed integrity checks to specific kinds of attacks. Also, monitoring the amount of

failed integrity checks over time can give development teams a solid argument to spend more effort on validating data on their side as data consumers.

6.2 Implications to Practice

We observed in this study that there are several limitations and improvement points for performing Data Protection Fortification, which companies should be aware of when applying the method. In this section, we provide recommendations for practitioners wanting to adopt DPF as part of their development process to evaluate if their data source handling is secure.

Use digital tools We found that facilitating DPF sessions using digital tools hold more promise than using physical tools for multiple reasons. As the workplace has become more flexible in terms of locations where one can work, some participants may be co-located in the office, while others join in from other locations. In this case, using digital tools is the only viable option. One of our lessons learned is that digital tools also scale better, allowing more participants to participate without slowing down the session considerably. It is also easier to collect and present the votes from participants when prioritizing data fields.

Compared to post-it notes and handwritten text on a whiteboard, the visual nature of the outcomes created in the digital tools makes it more likely to be easier to maintain the outcome in later sessions and reference after the meeting. However, compared to physical sessions, which require minimal preparations, hosting a digital session requires a non-trivial amount of time invested in preparing the templates with the specific data fields. Even though this may reduce the flexibility to include more data fields during the session, the previously mentioned benefits of digital tools outweighs this drawback.

In this study, only the facilitator could add security measures to the data field lanes in the digital whiteboard, but practitioners could experiment with having all participants edit at once. This could make the session more efficient in coming up with security measures, but with the possible consequence of the meeting becoming disorderly. One could set a timer where everyone brainstorms security measures independently and collectively determines the final result. The validation security measures selected for one field may also be duplicated for other similar fields, perhaps with minor variations, allowing the participants to achieve higher efficiency in the session.

Benefits of adopting DPF We recommend performing DPF on new data sources or when a data source specification changes. Given the (anticipated) low frequency of these events, DPF will not take up considerable development time. DPF enables the team to make more informed decisions on their handling of data sources and improve the confidence of end-users that the data they use is

trustworthy. In communication with stakeholders, DPF can be used as a tool for communication about security.

Data source limitations We experienced that DPF can be used on data sources with various origins, such as sensor data, data sent from web clients, and system-generated data. For the latter, we observed during one session that there were considerable challenges in prioritizing the data fields because they all felt equally important to the participants. From this, we derive a limitation on the potential for performing data field prioritization on data sources where all the data fields are needed to describe a phenomenon or where the fields do not have apparent differences in use. In such cases, the other steps in DPF can still provide valuable outcomes without the prioritization step. Another limitation is the relatively low number of data fields that are viable to cover in one DPF session. However, some parts of the activity are possible to do at a broader level, in particular, the data source discussion.

Participants The TAM evaluation performed with two of the participating teams shows positive trends among the participants towards using this in the future and that it was fun to do. A reason for the low score on *The activity fits into our team's way of working (PU5)* for one company could be because the development team resides in a non-English speaking country, where language, cultural or hierarchical differences may impact the adoption of DPF. We found that developers are likely to be able to perform and make use of the outcome from DPF on their own, contrary to many other threat modeling techniques where there is a need for a security expert to facilitate [23]. However, there is a need to include someone with knowledge about the data source for DPF to generate the most relevant results.

Focus of security measure discussion The security measures toolbox includes many simple validations that can be useful to consider. However, the focus should be on more domain-specific measures, as static analysis tools can not easily discover such validations missing in code. However, static code analysis tools can suggest missing syntactic validations (such as missing length constraints in a database schema). Even though many security measures in the toolbox may be regarded as good practices that should always be included, many practitioners might not be aware of them or sometimes forget them. Considering them explicitly in a DPF session can help both new and experienced developers learn good practices for security.

6.2.1 Recommended Practices for Securely Handling Data

In this section, we review the practices for handling data reported by the teams in each category and give recommendations to practitioners.

Data access Access to an external data source is often defined in a connection URL. If this URL can be tampered with by someone (usually by an insider), attacks such as spoofing the data source to send false data or causing a denial of service by removing or invalidating the URL are possible. For practitioners, we recommend audit logging any changes to this URL and making these audit logs readily available. Integration tests can also be used to verify the connection to the data source.

Access controls should be placed on the audit logs so that it is harder for an attacker to hide their tracks by deleting the logs. The commit history in Git, mentioned by some teams, can be easily modified or deleted and is thus not without flaws for this purpose. One recommendation is to require a reviewer to approve changes to the URL. Restricting who has access to change the data connection URL is a good practice. Removing access permissions promptly when someone leaves the team is also recommended. However, if few people have access to read the currently used URL, then few people are also able to detect tampering with it.

Data validity Implementing security measures in the application is a cybersecurity investment, as the developers have to spend their time doing this instead of developing new functionality. However, implementing measures such as validation can prevent more costly issues that can happen later. For practitioners, we encourage system designers and architects to define semantic validations related to how the system will be or is used. The generation of misuse-case diagrams can help identify requirements for semantic validation.

Syntactic validations can help reduce the risk of "low hanging fruit" tampering efforts, such as sending large amounts of text in an input field to cause a denial of service. Semantic validations help ensure the correct behavior and use of a system and may reduce the risk of more subtle tampering efforts that syntactic validations can not easily identify.

Data origin Data originating from IoT devices may be less trustworthy due to the challenges in protecting the data from these devices. Abnormal sensor readings may also naturally occur. Data originating from a human source also pose a risk for (un)intentional tampering.

One security expert mentioned that when one has no control over what someone can put into a data source, one must trust that the data is intact. However, as consumers of such data, it is still important to validate and monitor this data to prevent possible data poisoning attacks.

Data processing In one session, a new attack surface was uncovered by looking at how the data was processed. The end-users of this data often download the data from their website and import it into Excel for processing. While the website uses a framework with built-in injection protection, Excel provides better data processing tools. Since the data had to be processed, the protected view in Excel (a sandboxed environment) had to be turned off. Since data was not sanitized when

generating the CSV file (related to CWE-77¹), malicious cell data could potentially be included from the data fields coming in with the downloaded data. As a take-away message from this, we recommend practitioners to consider all places where the data might be processed, when coming up with security measures for it.

Data schema We recommend seeking to obtain an explicit schema and ensuring that you are notified before the schema changes. Looking at the database for the schema definition is not sufficient to understand what is considered valid data since validation efforts can be located in database triggers or stored procedures outside of the schema definition.

Data usage Except for presenting the data directly without using any frameworks for safely viewing the data or limiting the length, we found few practices that directly affected the risk of data tampering.

Fault mitigations A good practice we found was having an internal *whois* documentation for whom to contact if unavailability, inconsistencies, or missing data was detected in the data source. Relying on end-users to detect and report faults of the data source may lead to both frustrated users and loss of income. Health checks are one practice that can be used to detect downtime of a data source automatically. For detecting possible tampered data, we found that one team correlates the data they receive with another source. We argue that this must be done as an integrated, automatic process for it not to become a burden to the system operators.

6.3 Threats to Validity and Limitations

The scope of Data Protection Fortification is limited to focus on threats related to data tampering, from the point of view of a data consumer. The method can be adapted in future work to consider other threats than data tampering. Furthermore, DPF does not ensure correct implementation of the security measures elicited, which could be done, for instance, by generating test cases. DPF has been iteratively developed and tested on different kinds of data, where just one of these included data from IoT devices. While this has shown the method's viability for different kinds of data sources, it should be further tested on IoT data.

Due to time constraints, the data monitoring system was not fully developed nor evaluated in practice, and claims about achieving the desired properties (high performance and throughput) of the proposed architecture can not be made.

Since we did not make audio recordings of interviews or focus group sessions, we are likely to suffer from inaccuracy or incompleteness of data in the notes from journaling. To mitigate this, both researchers participated in every meeting

¹<https://cwe.mitre.org/data/definitions/77.html>

and held debriefs afterward to add more context and improve the notes. Furthermore, we do data triangulation by checking the findings with the source teams to avoid false interpretations and inconsistencies. We discussed the results with two supervisors to prevent researcher bias when analyzing and interpreting data. We increased the reliability of the study by leaving an audit trail in terms of raw data, field notes, researcher notes, and data analysis details, which we referenced actively during the analysis.

Due to the gradually evolving nature of DPF, the evaluations done in the early sessions may not reflect the current version of the method, and further studies are needed to evaluate it properly. As described in Section 4.5.3, some discussion questions have not been tested with participants. Furthermore, while the evaluations of DPF suggest that developers would be able to perform this method on their own (without a security expert), we have only evaluated it in sessions where the researchers were the facilitators. The participation of the researchers may have influenced how the participants understood the questions asked during the session.

When verbally evaluating DPF with participants, bias could be introduced as the participants might have been inclined to give less negative responses as a courtesy to the researchers. Furthermore, this kind of group evaluation can lead to some participants dominating in giving the feedback and some participants showing agreement implicitly through body language, which can lead to inconsistencies in the notes. These evaluations are also likely to be subject to response bias, where differences in how the researcher asked the questions could affect the answers received. Introducing the TAM-inspired evaluation in the two final DPF sessions made it easier for us to collect opinions from every participant regarding their attitude towards using DPF as a method. Though the results show that the participants are positive towards using DPF, we still would require more evaluations to give more credibility, as the results are not generalizable. Since these evaluations were carried out anonymously, this could reduce courtesy bias. The evaluations of DPF with security experts are limited by their short durations and unstructured format.

We believe that by describing the context of the participating teams, the details of the research process, and including different kinds of case teams, the results from this study can be easier to generalize.

6.4 Lessons Learned

During this research project, we have learned how the scope of a research project may change along the way as one gains more insight into the case one is researching. As none of the authors have any previous experience conducting research of this size, we also learned the methods that could support our research as the project progressed. For example, TAM evaluations proved very helpful in structuring the evaluation of the method we proposed, as opposed to only getting verbal feedback which we did in the first sessions. Knowing such methods will also be

valuable for us in our professional careers as developers.

From collaborating with companies, we felt that there was indeed a need for the research we were doing, and the general attitude towards performing the method was positive. As we saw from conducting DPF with the development teams, lots of tacit knowledge on security exists that could, with the correct tools, be shared among the team members.

Chapter 7

Conclusion

Due to the many challenges of securing IoT devices, it is difficult to ensure that data from these devices has not been tampered with by an adversary wanting to influence decision-making based on this data. Systems consuming this data must be designed with security measures to detect and prevent damage from data tampering attacks. However, there is a lack of methods to aid practitioners in understanding the risk in data.

Findings from the SLR on security testing of IoT that we did in our pre-study show that security is less focused in the design stage and that monitoring data sources is a potential way to improve security of IoT data. Based on these, we performed an initial exploratory study on five teams in the collaborating company to identify how and where to monitor data. We learned that there is an implicit trust in the data from external sources used by these teams and that monitoring the data sources is critical for detecting tampering. However, knowing what to monitor and how to do it is essential before integrating monitoring systems into an architecture. As a result, we developed a method named Data Protection Fortification (DPF) to help practitioners evaluate these requirements.

Data Protection Fortification (DPF) is a data-centric threat modeling method that aims to support development teams in evaluating if their handling of an external data source is secure and how to improve it. The method has also proved practical in other contexts beyond data originating in IoT devices. We also give recommendations to practitioners for secure handling of data sources and a conceptual architecture for a data monitoring system for detecting attempts at tampering with data.

Addressing security from a low-granularity data-oriented perspective showed promise for acceptance among practitioners. Developers felt familiar with looking at securing data fields and discussing their implications for use across various system functions. Contrary to many other threat modeling methods, DPF shows the potential not to require a security expert to facilitate the meeting. Applying DPF in the design phase equips the development team with a structured method to iron out possible assumptions about the data. Unless addressed, these assumptions may introduce security challenges when implementing functions using the data.

Furthermore, DPF has the potential to become a future communication tool for security between developers and stakeholders by helping to highlight where more security efforts are needed. While DPF is not meant to be the only activity a team does for security, it can be a good starting point.

In the future, one could extend DPF with a structured output, possibly bridging the existing gap between the design of security measures and implementation. We also call for further research on developing data-centric threat modeling approaches for IoT and to build on the conceptual monitoring system to assess its viability in detecting data tampering attacks.

Data Protection Fortification is currently being evaluated in a industrial project in the context of a data provider.

Bibliography

- [1] S. Chakrabarty and D. W. Engels, "A secure iot architecture for smart cities," in *2016 13th IEEE annual consumer communications & networking conference (CCNC)*, IEEE, 2016, pp. 812–813.
- [2] T. Qiu, Z. Zhao, T. Zhang, C. Chen, and C. P. Chen, "Underwater internet of things in smart ocean: System architecture and open issues," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4297–4307, 2019.
- [3] J. Sakhnini, H. Karimipour, A. Dehghantanha, R. M. Parizi, and G. Srivastava, "Security aspects of internet of things aided smart grids: A bibliometric survey," *Internet of things*, vol. 14, p. 100 111, 2021.
- [4] Statista. "Number of internet of things (iot) connected devices worldwide from 2019 to 2030, by vertical." (Jun. 2022), [Online]. Available: <https://www.statista.com/statistics/1194682/iot-connected-devices-vertically/>.
- [5] D.-W. Huang, W. Liu, and J. Bi, "Data tampering attacks diagnosis in dynamic wireless sensor networks," *Computer Communications*, vol. 172, pp. 84–92, 2021.
- [6] S. Siboni, V. Sachidananda, Y. Meidan, M. Bohadana, Y. Mathov, S. Bhairav, A. Shabtai, and Y. Elovici, "Security testbed for internet-of-things devices," *IEEE transactions on reliability*, vol. 68, no. 1, pp. 23–44, 2019.
- [7] G. Sharma, S. Vidalis, N. Anand, C. Menon, and S. Kumar, "A survey on layer-wise security attacks in iot: Attacks, countermeasures, and open-issues," *Electronics*, vol. 10, no. 19, p. 2365, 2021.
- [8] H. T. Reda, A. Anwar, and A. Mahmood, "Comprehensive survey and taxonomies of false data injection attacks in smart grids: Attack models, targets, and impacts," *Renewable and Sustainable Energy Reviews*, vol. 163, p. 112 423, 2022.
- [9] M. Souppaya and K. Scarfone, "Guide to data-centric system threat modeling," National Institute of Standards and Technology, Tech. Rep., 2016.
- [10] S. M. Kvamme and E. Gudmundsen, "On security testing of iot systems: A systematic literature review," Department of Computer Science, Norwegian University of Science and Technology, Tech. Rep., 2021, Appendix E.

- [11] I. Shrestha and M. Hale, "Detecting dynamic security threats in multi-component iot systems," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.
- [12] IEEE, "Iso/iec/ieee international standard - systems and software engineering-vocabulary," *ISO/IEC/IEEE 24765:2017(E)*, pp. 1–541, 2017. DOI: 10.1109/IEEESTD.2017.8016712.
- [13] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi, "A taxonomy of computer program security flaws," *ACM Comput. Surv.*, vol. 26, no. 3, pp. 211–254, Sep. 1994, ISSN: 0360-0300. DOI: 10.1145/185403.185412. [Online]. Available: <https://doi.org/10.1145/185403.185412>.
- [14] NIST. "Vulnerabilities." (Jun. 2022), [Online]. Available: <https://nvd.nist.gov/vuln>.
- [15] G. McGraw, "Software security," *IEEE Security Privacy*, vol. 2, no. 2, pp. 80–83, 2004. DOI: 10.1109/MSECP.2004.1281254.
- [16] W. A. Jansen, T. Winograd, K. Scarfone, *et al.*, "Guidelines on active content and mobile code," *NIST Special Publication*, vol. 800, p. 28, 2001.
- [17] H. Ni, A. Chen, and N. Chen, "Some extensions on risk matrix approach," *Safety Science*, vol. 48, no. 10, pp. 1269–1278, 2010, ISSN: 0925-7535. DOI: <https://doi.org/10.1016/j.ssci.2010.04.005>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925753510001049>.
- [18] NIST, "Minimum security requirements for federal information and information systems," 2006. DOI: 10.6028/NIST.FIPS.200.
- [19] R. Bachmann and A. D. Brucker, "Developing secure software: A holistic approach to security testing," *Datenschutz und Datensicherheit (DuD)*, vol. 38, pp. 257–261, 2014.
- [20] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements," in *Symposium on requirements engineering for information security (SREIS)*, Citeseer, vol. 2005, 2005, pp. 1–8.
- [21] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.
- [22] K. Tuma, G. Calikli, and R. Scandariato, "Threat analysis of software systems: A systematic literature review," *Journal of Systems and Software*, vol. 144, pp. 275–294, 2018.
- [23] K. H. Håkonsen and V. Ahmadi, "Threat analysis in agile," Department of Computer Science, Norwegian University of Science and Technology, Tech. Rep., 2021.
- [24] M. Souppaya and K. Scarfone. "Guide to data-centric system threat modeling (no. nist special publication (sp) 800-154 (draft)," National Institute of Standards and Technology (NIST). (Mar. 2016), [Online]. Available: <https://csrc.nist.gov/publications/detail/sp/800-154/draft>.

- [25] OWASP. "Threat modelling process." (Dec. 2021), [Online]. Available: https://owasp.org/www-community/Threat_Modeling_Process.
- [26] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and A. Pretschner, "Security testing: A survey," in *Advances in Computers*, vol. 101, Elsevier, 2016, pp. 1–51.
- [27] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack. "Uncover security design flaws using the stride approach." (Jul. 2019), [Online]. Available: <https://docs.microsoft.com/en-us/archive/msdn-magazine/2006/november/uncover-security-design-flaws-using-the-stride-approach>.
- [28] N. Messe, V. Chiprianov, N. Belloir, J. El-Hachem, R. Fleurquin, and S. Sadou, "Asset-oriented threat modeling," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 491–501. DOI: 10.1109/TrustCom50675.2020.00073.
- [29] K. Bernsmed, D. S. Cruzes, M. G. Jaatun, and M. Iovan, "Adopting threat modelling in agile software development projects," *Journal of Systems and Software*, vol. 183, p. 111 090, 2022.
- [30] M. Csikszentmihalyi and M. Csikzentmihaly, *Flow: The psychology of optimal experience*. Harper & Row New York, 1990, vol. 1990.
- [31] P. Sweetser and P. Wyeth, "Gameflow: A model for evaluating player enjoyment in games," *Computers in Entertainment (CIE)*, vol. 3, no. 3, pp. 3–3, 2005.
- [32] L. Williams, A. Meneely, and G. Shipley, "Protection poker: The new software security "game"," *IEEE Security & Privacy*, vol. 8, no. 3, pp. 14–20, 2010.
- [33] SINTEF. "Protection poker." (), [Online]. Available: <https://www.sintef.no/en/digital/sos-agile-blog/protection-poker/>.
- [34] L. Williams, M. Gegick, and A. Meneely, "Protection poker: Structuring software security risk assessment and knowledge transfer," in *Engineering Secure Software and Systems*, F. Massacci, S. T. Redwine, and N. Zannone, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 122–134, ISBN: 978-3-642-00199-4.
- [35] C. S. R. C. (NIST. "Glossary." (Mar. 2022), [Online]. Available: <https://csrc.nist.gov/glossary>.
- [36] Y. Wang, E. Kjerstad, and B. Belisario, "A dynamic analysis security testing infrastructure for internet of things," in *2020 Sixth International Conference on Mobile And Secure Services (MobiSecServ)*, IEEE, 2020, pp. 1–6.
- [37] Y. Mahmoodi, S. Reiter, A. Viehl, O. Bringmann, and W. Rosenstiel, "Attack surface modeling and assessment for penetration testing of iot system designs," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, IEEE, 2018, pp. 177–181.

- [38] S. Karagiannis, M. Manso, E. Magkos, L. L. Ribeiro, and L. Campos, “Automated and on-demand cybersecurity certification,” in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, IEEE, 2021, pp. 174–179.
- [39] M. Pucher, C. Kudera, and G. Merzdovnik, “Avrs: Emulating avr microcontrollers for reverse engineering and security testing,” in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020, pp. 1–10.
- [40] J. Vijiuk, L. Perkov, and A. Krog, “Bug detection in embedded environments by fuzzing and symbolic execution,” in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, IEEE, 2020, pp. 1218–1223.
- [41] T. Yue, P. Wang, Y. Tang, E. Wang, B. Yu, K. Lu, and X. Zhou, “EcoFuzz: Adaptive Energy-Saving greybox fuzzing as a variant of the adversarial Multi-Armed bandit,” in *29th USENIX Security Symposium (USENIX Security 20)*, USENIX Association, Aug. 2020, pp. 2307–2324, ISBN: 978-1-939133-17-5. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity20/presentation/yue>.
- [42] J. Men, G. Xu, Z. Han, Z. Sun, X. Zhou, W. Lian, and X. Cheng, “Finding sands in the eyes: Vulnerabilities discovery in iot with eufuzzer on human machine interface,” *IEEE Access*, vol. 7, pp. 103 751–103 759, 2019.
- [43] Y. Zheng, A. Davanian, H. Yin, C. Song, H. Zhu, and L. Sun, “Firm-afl: High-throughput greybox fuzzing of iot firmware via augmented process emulation,” in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1099–1114.
- [44] P. Sun, L. Garcia, G. Salles-Loustau, and S. Zonouz, “Hybrid firmware analysis for known mobile and iot security vulnerabilities,” in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2020, pp. 373–384.
- [45] S. Marksteiner, R. Ramler, and H. Sochor, “Integrating threat modeling and automated test case generation into industrialized software security testing,” in *Proceedings of the Third Central European Cybersecurity Conference*, 2019, pp. 1–3.
- [46] M. Bettayeb, O. A. Waraga, M. A. Talib, Q. Nasir, and O. Einea, “Iot testbed security: Smart socket and smart thermostat,” in *2019 IEEE Conference on Application, Information and Network Security (AINS)*, IEEE, 2019, pp. 18–23.
- [47] A. Liu, A. Alqazzaz, H. Ming, and B. Dharmalingam, “Iotverif: Automatic verification of ssl/tls certificate for iot applications,” *IEEE Access*, 2019.

- [48] R. Cayre, V. Nicomette, G. Auriol, E. Alata, M. Kaaniche, and G. Marconato, "Mirage: Towards a metasploit-like framework for iot," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019, pp. 261–270. DOI: 10.1109/ISSRE.2019.00034.
- [49] B. Feng, A. Mera, and L. Lu, "P2im: Scalable and hardware-independent firmware testing via automatic peripheral interface modeling," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1237–1254.
- [50] T. Zitta, M. Neruda, L. Vojtech, M. Matejkova, M. Jehlicka, L. Hach, and J. Moravec, "Penetration testing of intrusion detection and prevention system in low-performance embedded iot device," in *2018 18th International Conference on Mechatronics-Mechatronika (ME)*, IEEE, 2018, pp. 1–5.
- [51] R. Ankele, S. Marksteiner, K. Nahrgang, and H. Vallant, "Requirements and recommendations for iot/iilot models to automate security assurance through threat modelling, security analysis and penetration testing," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ser. ARES '19, Canterbury, CA, United Kingdom: Association for Computing Machinery, 2019, ISBN: 9781450371643. [Online]. Available: <https://doi.org/10.1145/3339252.3341482>.
- [52] C. M. Coman, G. D'amico, A. V. Coman, and A. Florescu, "Techniques to improve reliability in an iot architecture framework for intelligent products," *IEEE Access*, vol. 9, pp. 56 940–56 954, 2021.
- [53] T. W. Tseng, C. T. Wu, and F. Lai, "Threat analysis for wearable health devices and environment monitoring internet of things integration system," *IEEE Access*, vol. 7, pp. 144 983–144 994, 2019.
- [54] M. Ren, X. Ren, H. Feng, J. Ming, and Y. Lei, "Z-fuzzer: Device-agnostic fuzzing of zigbee protocol implementation," in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021, pp. 347–358.
- [55] P. Johannesson and E. Perjons, *An introduction to design science*. Springer, 2014, vol. 10.
- [56] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.
- [57] R. Davison, M. G. Martinsons, and N. Kock, "Principles of canonical action research," *Information systems journal*, vol. 14, no. 1, pp. 65–86, 2004.
- [58] F. D. Davis, "A technology acceptance model for empirically testing new end-user information systems: Theory and results," Ph.D. dissertation, Massachusetts Institute of Technology, 1985.
- [59] M. Chuttur, "Overview of the technology acceptance model: Origins, developments and future directions," 2009.

- [60] C. Robson and K. McCartan, *Real world research: a resource for users of social research methods in applied settings*. Wiley, 2016.
- [61] OWASP. “Cheat sheet series.” (Jan. 2021), [Online]. Available: <https://cheatsheetseries.owasp.org/>.
- [62] B. J. Oates, *Researching Information Systems and Computing*. Sage, 2006.
- [63] M. Chronopoulos, E. Panaousis, and J. Grossklags, “An options approach to cybersecurity investment,” *IEEE Access*, vol. 6, pp. 12 175–12 186, 2017.
- [64] P. Ow. “Making decisions under uncertainty and risk.” (Apr. 2022), [Online]. Available: <https://practicalrisktraining.com/making-decisions-under-uncertainty-and-risk>.
- [65] J. Winkler and R. Moser, “Biases in future-oriented delphi studies: A cognitive perspective,” *Technological forecasting and social change*, vol. 105, pp. 63–76, 2016.
- [66] M. Richards, *Software architecture patterns*. O’Reilly Media, Incorporated 1005 Gravenstein Highway North, Sebastopol, CA . . . , 2015, vol. 4.
- [67] D. S. Cruzes, M. G. Jaatun, K. Bernsmed, and I. A. Tøndel, “Challenges and experiences with applying microsoft threat modeling in agile development projects,” in *2018 25th Australasian Software Engineering Conference (ASWEC)*, IEEE, 2018, pp. 111–120.
- [68] M. Abi-Antoun and J. M. Barnes, “Analyzing security architectures,” in *Proceedings of the IEEE/ACM international conference on Automated software engineering*, 2010, pp. 3–12.
- [69] J. Espenschied and A. Gunn, “Threat genomics,” *Metricon 7.0. Bellevue, WA*, 2012.

Appendix A

DPF Cheat Sheet

Data Protection Fortification

Cheat Sheet

1 – Data Source: General Discussion

Q01 - What describes the data we get from this data source?

Related questions: What does this data source contain?

Examples: Weather data, gas prices, scooter info, customer reviews

Focus: Common understanding of data source

Q02 - Who/What provides this data source?

Related questions: What is the name of the provider?

Examples: Data supplier, yr.no, internal team, student, company-owned sensors

Focus: Common understanding of data source, trust level

Q03 - How do you receive the data?

Related questions: Where in the target system is data imported?

Examples: Fetching data from API, importing from a database, receiving directly from device, form input posted to backend

Focus: Entry point, data flow

Q04 - How is the data from this data source used in your services or products now, or will be in the future?

Related questions: How can users use the data directly or indirectly?

Examples: Used in decision-making, machine learning model, aggregated and shown in dashboard, presented directly to end user

Focus: Attack surface, exit point, data flow

Q05 - Who are the end-users for this data?

Examples: Internal analysts, maintenance, customers, customer support, users of an application, management

Focus: Common understanding of data source

Q06 - How often is data received from this data source?

Related questions: Is the data cached, and if so, for how long?

Examples: The data is downloaded by a CRON-job every night, pulled from the device every 5 min, fetched when a user triggers a service (e.g., clicks to view details)

Focus: Common understanding of data source

Q07 - What is the volume of the data received?

Examples: Batches of 100 million rows, 15 data fields, 10 kilobytes

Focus: Common understanding of data source

Q08 - Do you have an explicit schema or API-contract for data coming from this data source?

Related questions: Could the values be "anything"?

Examples: Swagger-API, System documentation Focus: Integrity, Common understanding of data source

Q09 - How is the data coming from this data source generated?

Related questions: Is it known who puts the data into this data source?

Examples: Human input through a webpage form, IoT device, system generated

Focus: Threat actors, attack surface, entry point, trust in data

#2 Data Source: Security Implications

Q10 - How is data secured during transit?

Examples: HTTPS, hash/MAC/digital signature verification

Focus: Confidentiality, integrity, non-repudiation

Q11 - Does any protections exist on the device to prevent physical tampering?

Related questions: Can you detect physical tampering?

Examples: Behind locked doors, requires a special key to open the device, hardened physical design, the device has an alarm, monitoring

Focus: Integrity

Q12 - Who has access to change the connection URL used to connect with the data source?

Related questions: Where is this access URL stored? Is a change to this access URL logged?

Examples: environment variable in Azure, configuration in database

Focus: Authenticity, availability

Q13 - How sensitive is the data in this data source?

Examples: Person Identifiable Information (PII), information about business opportunities/tenders, data that could affect stock prices, data that could affect decision-making processes

Focus: Data sensitivity, confidentiality

Q14 - What could an attacker be interested in influencing through this data source?

Examples: Analyst decisions, planned maintenance, conceal information or impact repudiation, company damage

Focus: Misuse cases, threat actor goals

Q15 - What could the consequences be if the data source was no longer available or parts of the data were missing?

Related questions: Are there certain times where the consequence would be greater?

Examples: The service provided is rendered useless, users are denied access to a digital voting platform (but access to this platform is only critical during the days where voting is open)

Focus: Availability, integrity

Q16 - Who are the possible threat actors for this data source?

Examples: Other nations, market competitors, disloyal employees or ex-employees, terrorists, script kids

Focus: Threat actors

Q17 - Would you discover if the data from this data source is incorrect, or if the data source is unavailable?

Related questions: How would you report unavailability or inconsistencies to the suppliers of this data source?

Examples: Service availability monitoring (health checks), users would discover unavailability or inconsistencies and report back to the team

Focus: Fault mitigation

Appendix B

DPF Preparation Meeting Guide

Forberedelsesmøte til Data Protection Fortification

Om Teamet	
Antall teammedlemmer?	
Hvilke roller finnes i teamet?	
Praktiseres det smidig metodikk? Sprint planning? (Planning Poker?)	
Hvor lenge har teamet eksistert?	
Hvilket behov dekker teamet? Rollen i organisasjonen?	

Om Datakildene	
Hvilke systemer har dere?	
Hvilke forskjellige datakilder bruker dette systemet?	
Hva beskriver dataen vi får fra denne datakilden?	
Hvor blir dataen brukt i produktene deres?	
Hvem tilbyr datakilden?	
Hvem benytter seg av dataproduktet dere leverer?	
Hvordan deler dere dataen med de som skal bruke den?	
Hvilke attributter har dataobjektene fra denne kilden? Hvilke funksjoner påvirker attributtet?	
Hva kan konsekvensene være dersom dataen som kommer inn er feil?	
Hvor stor kunnskap har teamet om detaljene i datakildene? Hvor mye kan de om kilden?	

Appendix C

DPF Data Extraction Form

Data Extraction

Data Protection Fortification

Meta data	
Date	
Number of participants	
Distribution of roles participating	

1 - Discuss Data Source		
General Questions		
Examples	Data source in focus	
<i>Weather, prices, customer reviews</i>	What describes the data we get from this data source?	
<i>What is the name of the provider? External to company?</i>	Who/What provides this data source?	
<i>Fetching from API? Importing from database?</i>	How do you receive the data?	
<i>Used in processing, aggregated, presented directly to end user, used in machine learning</i>	How is the data from this data source used in your services or products now, or will be in the future?	
<i>Internal analysts, customers, customer support</i>	Who are the end-users for this data?	
<i>Fetches when user triggers a service (clicks to view details about an order), data is downloaded by a CRON-job every night</i>	How often is data received from this data source?	
	What is the volume of the data received?	
<i>Could the values be "anything"?</i>	Do you have an explicit schema or API-contract for	

	data coming from this data source?	
<i>Is it known who puts the data into this data source?</i>	How is the data coming from this data source generated?	
Questions for Security Implications		
<i>HTTPS, hash/MAC/digital signature verification</i>	How is data secured during transit?	
<i>Behind locked doors, requires a special key to open the device, hardened physical design, the device has an alarm, monitoring</i>	Does any protections exist on the device to prevent physical tampering?	
<i>environment variable in Azure, configuration in database</i>	Who has access to change the access URL used to connect with the data source? <i>Where is this access URL stored? Is a change to this access URL logged?</i>	
<i>Person Identifiable Information (PII), concerns business opportunities, data that could affect stock prices, data that could affect decision making processes</i>	How sensitive is the data in this data source?	
<i>Analyst decisions, planned maintenance, conceal information or impact repudiation, company damage</i>	What could an attacker be interested in influencing through this data source?	
<i>The service provided is rendered useless, users are denied access to a digital voting platform (but access to this platform is only critical during the days where voting is open)</i>	What could the consequences be if the data source was no longer available, or parts of the data was missing? <i>Are there certain times where the consequence would be greater?</i>	
<i>Other nations, market competitors, disloyal employees or ex-employees, terrorists, script kids</i>	Who are the possible threat actors for this data source?	

<p><i>Service availability monitoring (health checks). Users would discover unavailability or inconsistencies and report back to us</i></p>	<p>Would you discover if the data from this data source is incorrect, or if the data source is unavailable?</p> <p><i>How would you report unavailability or inconsistencies to the suppliers of this data source?</i></p>	
	<p>Other</p>	

2 - Prioritization of Data Fields	
Estimate Value	
<p>Max value Min value Remaining values</p>	
<p>General discussion</p>	
Estimate Likelihood of Tampered Values	
<p>Max value Min value Remaining values</p>	
<p>General discussion</p>	
Prioritization Matrix	

<p>Does the result shown in the matrix make sense?</p> <p>Should it be rearranged?</p>	
---	--

3 – Identify Security Measures

<p>Evaluate security measures discussion</p> <p>Is the current handling of the data source sufficient?</p> <p>In what ways can we fortify the security of handling the data source?</p> <p><i>Examples:</i></p> <p><i>Validation (type-checking? min/max values? min/max length?)</i></p> <p><i>Monitoring (anomaly detection, correlation, auditing)</i></p> <p><i>Other measures that can decrease the risk?</i></p>	
---	--

Evaluation of Session

<p>Feedback from practitioners</p> <p>How did you feel like the session went?</p>	
--	--

Appendix D

NSD Notification Form

NSD NORSK SENTER FOR FORSKNINGSDATA

Meldeskjema

Referansenummer

447600

Hvilke personopplysninger skal du behandle?

- Bakgrunnsopplysninger som vil kunne identifisere en person

Beskriv hvilke bakgrunnsopplysninger du skal behandle

Position in workplace, years of experience

Prosjektinformasjon

Prosjektittel

Improving security practices in agile development teams

Prosjektbeskrivelse

Investigate the current state of security practices at a Norwegian-based energy company. What security testing activities are they doing? Find areas can security testing practices be improved or introduced. Given the large amount of data generated by their systems, how can one detect security issues at scale? How do they verify the integrity of the data today?

By interviewing different project roles we can gain insight into the different perceptions different roles may have about the security activities they are doing. The data collected from the interviews help us gain an understanding on the current state of practices, which can be used for finding areas of improvement.

Begrunn behovet for å behandle personopplysningene

To correlate work experience with perception of security we need to collect the minimum required personal data needed for this. We also need to gather consent form participants.


Ekstern finansiering

Type prosjekt

Chapter D: NSD Notification Form

123

Studentprosjekt, masterstudium

Kontaktinformasjon, studentSigrid Marita Kvamme, sigrimk@stud.ntnu.no, tlf: **Behandlingsansvar**

Behandlingsansvarlig institusjon

Norges teknisk-naturvitenskapelige universitet / Fakultet for informasjonsteknologi og elektroteknikk (IE) / Institutt for datateknologi og informatikk

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)

Daniela Soares Cruzes, daniela.s.cruzes@ntnu.no, tlf: 94249891

Skal behandlingsansvaret deles med andre institusjoner (felles behandlingsansvarlige)?

Nei

Utvalg 1

Beskriv utvalget

Developers in a development team

Rekruttering eller trekking av utvalget

The sample will be recruited through conversation with the employees working in relevant teams in the company.

Alder

20 - 65

Inngår det voksne (18 år +) i utvalget som ikke kan samtykke selv?

Nei

Personopplysninger for utvalg 1

- Bakgrunnsopplysninger som vil kunne identifisere en person

Hvordan samler du inn data fra utvalg 1?**Gruppeintervju**

124

*E. Gudmundsen & S. Kvamme: Data Protection Fortification***Grunnlag for å behandle alminnelige kategorier av personopplysninger**

Samtykke (art. 6 nr. 1 bokstav a)

Gruppeintervju**Grunnlag for å behandle alminnelige kategorier av personopplysninger**

Samtykke (art. 6 nr. 1 bokstav a)

Informasjon for utvalg 1**Informerer du utvalget om behandlingen av opplysningene?**

Ja

Hvordan?

Skriftlig informasjon (papir eller elektronisk)

Tredjepersoner

Skal du behandle personopplysninger om tredjepersoner?

Nei

Dokumentasjon

Hvordan dokumenteres samtykkene?

- Elektronisk (e-post, e-skjema, digital signatur)

Hvordan kan samtykket trekkes tilbake?

By contacting the researchers using the contact information given on the consent form.

Hvordan kan de registrerte få innsyn, rettet eller slettet opplysninger om seg selv?

The subjects can get access to all content they have provided through interviews, by contacting the researchers. More technically, we will share the related folder with read-only access to the subject.

Requests to correct or delete personal data will be followed up by processing all relevant documents, and correcting the study's results. This can be requested at any time during the project duration.

Chapter D: NSD Notification Form

125

Totalt antall registrerte i prosjektet

1-99

Tillatelser

Skal du innhente følgende godkjenninger eller tillatelser for prosjektet?**Behandling**

Hvor behandles opplysningene?

- Ekstern tjeneste eller nettverk (databehandler)

Hvem behandler/har tilgang til opplysningene?

- Student (studentprosjekt)
- Databehandler
- Prosjektansvarlig

Hvilken databehandler har tilgang til opplysningene?

Microsoft Office 365. The project leader will not have access to the voice recordings.

Tilgjengeliggjøres opplysningene utenfor EU/EØS til en tredjestat eller internasjonal organisasjon?

Nei

Sikkerhet

Oppbevares personopplysningene atskilt fra øvrige data (koblingsnøkkel)?

Ja

Hvilke tekniske og fysiske tiltak sikrer personopplysningene?

- Opplysningene anonymiseres fortløpende
- Opplysningene krypteres under forsendelse
- Adgangsbegrensning
- Flerfaktorautentisering
- Endringslogg

126

E. Gudmundsen & S. Kvamme: Data Protection Fortification

Varighet

Prosjektperiode

14.02.2022 - 30.11.2022

Skal data med personopplysninger oppbevares utover prosjektperioden?

Nei, alle data slettes innen prosjektslutt

Vil de registrerte kunne identifiseres (direkte eller indirekte) i oppgave/avhandling/øvrige publikasjoner fra prosjektet?

Nei

Tilleggsopplysninger

Appendix E

On Security Testing of IoT Systems: A Systematic Literature Review

On Security Testing of IoT Systems

A Systematic Literature Review

Sigrid Marita Kvamme

Espen Gudmundsen

Autumn 2021

Supervisor: Daniela Soares Cruzes

Co-supervisor: Tosin Daniel Oyetoyan

Contents

Contents	i
Figures	iii
Tables	iv
Acronyms	v
Glossary	vi
1 Introduction	1
2 Background and Related Work	3
2.1 Secure Software Development Life-Cycle	3
2.2 DevOps	4
2.3 Approaches and techniques for Security Testing	5
2.3.1 Model-based Security Testing	6
2.3.2 Code-based Security Testing and Static Analysis	6
2.3.3 Penetration Testing and Dynamic Analysis	6
2.3.4 Testbeds for IoT	7
2.4 Related work	7
3 Research Methodology	8
3.1 Research Questions	8
3.2 Data Sources and Search Strategy	10
3.3 Inclusion and Exclusion Criteria	11
3.4 Quality Criteria	12
3.5 Study Selection Steps	13
3.6 Data Extraction and Data Synthesis	14
4 Results	16
4.1 RQ1 Approaches	18
4.1.1 RQ1.1 Categorizing	18
4.1.2 RQ1.2 Application	19
4.1.3 RQ1.3 Generating	19
4.1.4 RQ1.4 Executing	21
4.1.5 RQ1.5 Reporting	23
4.2 RQ2 Automation	26
4.3 RQ3 Adoption	28
4.3.1 RQ3.1 Ease of adoption to agile teams	30
4.3.2 RQ3.2 Prioritization of test cases	31
4.3.3 RQ3.3 Scoping of test areas	31

5 Discussion	32
5.1 RQ1 Approaches	32
5.1.1 RQ1.1 Categorizing	32
5.1.2 RQ1.2 Application	33
5.1.3 RQ1.3 Generating	35
5.1.4 RQ1.4 Executing	36
5.1.5 RQ1.5 Reporting	36
5.2 RQ2 Automation	38
5.3 RQ3 Adoption	38
5.3.1 RQ3.1 Ease of adoption to agile teams	39
5.3.2 RQ3.2 Prioritization of test cases	40
5.3.3 RQ3.3 Scoping of test areas	40
5.4 Implications for Research	40
5.5 Implications for Practice	41
5.6 Threats to Validity	41
6 Conclusion	43
6.0.1 Future Work	43
Bibliography	44

Figures

2.1	The DevOps Life-cycle	5
3.1	Study selection process for the study	14
4.1	Selected papers distributed by publication year	16
4.2	Overview of the required skills needed to perform the approach . .	23
4.3	Types of report file outputs from each approach	25
4.4	Degree of automation of the approaches	28
4.5	Expertise needed to make full use of the approach	29
4.6	Maturity of the tools presented in the approaches	30

Tables

2.1	Security activities in development phases	4
3.1	Final Search Strings	11
3.2	Inclusion and exclusion criteria	12
3.3	Quality criteria	13
3.4	Filtering steps in study selection process	14
3.5	Data extraction fields	15
4.1	Overview of the selected papers	17
4.2	Each paper mapped to a type of security testing approach	18
4.3	Each paper mapped to a development phase	19
4.4	Explanation of categories for preparations needed	20
4.5	Explanation of the level of preparation needed	20
4.6	Preparations needed for each approach	21
4.7	Explanation of expertise roles	22
4.8	The output from each approach	24
4.9	Output target audience for approaches	26
4.10	Explanation of levels of automation	28
4.11	Explanation of levels of tool maturity used for ease of adoption	29

Acronyms

- CD** Continuous Deployment. 34, 38, 41
- CI** Continuous Integration. 30, 34, 38, 41
- CVE** Common Vulnerabilities and Exposures. 19, 27, 30, 31, 33, 39, 40
- DAST** Dynamic Application Security Testing. 6, 22
- GDPR** General Data Protection Regulation. 30
- IIoT** Industrial IoT. 4
- IoT** Internet of Things. 1, 2, 7, 10, 14, 18, 20, 25, 30, 32–35, 38–41, 43
- MBST** Model-based Security Testing. 6, 18, 33
- ML** Machine Learning. 18, 25, 27, 31, 33, 39, 41
- SAST** Static Application Security Testing. 4, 6, 33, 34, 41
- SDLC** Software Development Life-Cycle. 3
- SLR** Systematic Literature Review. 2, 8, 9, 12, 13
- SSDLC** Secure Software Development Life-Cycle. 3, 34
- SUT** System Under Test. 5

Glossary

asset Some data or component of a system that has to be protected, and is associated with one or more security properties that has to hold [1], e.g. a database that should not be leaked. vi

bug Causes a program to crash or behave unexpectedly. May become a vulnerability if it's exploitable. 23, 37

exploit A concrete malicious input that makes use of a vulnerability in the system to violate a security property of an asset [1]. vi

risk Likelihood of an unwanted incident, and its consequence for a specific asset [1]. The risk is proportional to the likelihood of an incident occurring and the value of the asset. 23, 37

vulnerability A verified security fault in the system. Either a security mechanism is missing, or is implemented in a faulty way [1]. 23, 37

Chapter 1

Introduction

The Internet of Things (IoT) has in recent years gained a lot of traction. It is rapidly being introduced in many different fields such as agriculture, city life, health care, smart homes and manufacturing. The benefits it can give are attractive for many industries, especially for cutting costs and increasing efficiency by reducing manual labor.

Appliance in industry range from industrial power grids [2] to monitoring the use of football fields [3] to lengthen the life span of the grass mats. Another appliance of IoT includes monitoring of water levels in sewers [4] to warn against flooding, allowing precautions to be taken in time before any harm is done.

There are many medical uses as well, both in the context of personal health and professional health care. Smartwatches can monitor heart rate and report steps taken throughout the day for personal interest. Safety alarms used in health care, may trigger an emergency response when a patient has fallen down, which for many elderly could indicate life threatening circumstances. There are also sensors for diabetics that monitor glucose levels in the blood, and automatically triggers an alarm when it gets too low or too high.

The usages of IoT are many, and by consequence, so are the misuse cases. Attacks are only multiplying with time, with an 100% increase in attacks against IoT devices in only the first half of 2021 [5]. A recent ransomware attack that shut down a fuel pipeline in the U.S. in 2021 [6] exemplifies the need for an increased push for cybersecurity standards and regulations. In May 2021, the Biden government of the USA issued an executive order on improving cybersecurity [7].

With the striking increase in attacks targeting IoT recently, research into methods for ensuring securing mechanisms and integrity of devices are highly relevant. Though testing of IoT poses a wide range of challenges, such as tightly coupled hardware and software [8], we believe that synthesizing current knowledge on approaches used for security testing will guide practitioners in adopting techniques or tools fit for their systems and expertise.

To the best of our knowledge, this is the first systematic literature on the security testing of IoT systems. The main contributions of this paper are the following:

- A review of the state-of-the-art security testing approaches for IoT.
- An assessment of the degree of automation in the approaches, and their ease of adoption to guide practitioners.
- Recommendations for practitioners wanting to adopt security testing for their IoT systems.
- Possible directions for future research, based on existing literature.

The rest of the SLR is organized as follows. Chapter 2 gives an introduction to the concepts and fields relevant for this thesis and the related work. In Chapter 3 we describe the research protocol and steps taken for the SLR. The results are presented in Chapter 4, with a discussion of the findings in Chapter 5. Finally, we conclude the SLR in Chapter 6.

This SLR is part of a contribution on how to perform security testing of IoT in an agile setting. This paper will synthesize current knowledge in the field, providing guidance for our future work. The contributions of this paper will be further expanded on when building a prototype of a tool to support security testing.

Chapter 2

Background and Related Work

In this chapter we explain concepts that are relevant for this SLR.

2.1 Secure Software Development Life-Cycle

The Secure Software Development Life-Cycle (SSDLC) encompasses integrating security as part of all the development phases defined in the Software Development Life-Cycle (SDLC). The SDLC phases usually include, *Analysis, Design, Development, Deployment* and *Monitoring* [1]. The SSDLC guide development teams into a "shift-left" in security, by proposing security activities in all of these phases. This places the responsibility for security onto the developers, not as an afterthought performed when the software is ready for production. To quote the classic paper of McGraw [9] on Software Security; "There is no such thing as magic crypto fairy dust: we need to focus on software security from the ground up."

Bachmann and Brucker [10] reports security best practices recommended for practitioners during different phases of development. Though these recommendations are meant for security activities in general and not for testing, the development phases presented are a good fit also for this study. In contrast to the phases usually included in SDLC, we believe it is easier for practitioners to familiarize and distinct the phases described in [10]. Table 2.1 attempts to highlight these similarities of the two, including examples of security activities commonly found in the phases.

Table 2.1: Development phases presented by Bachmann and Brucker [10] and the phases usually found in Software Development Life-Cycle (SDLC)

Bachmann and Brucker	SDLC	Security Activity Ex.
During Planning and Design	Analysis Design	Threat Modelling
During Application Development	Development Deployment	SAST
Executable in a Test Environment		Fuzz Testing
System Operation and Maintenance	Monitoring	Regression Testing

2.2 DevOps

DevOps is an acronym combining the two practices (software) *developers* and *operations* [11]. In practice, it seeks to integrate these two practices through the use of automation and monitoring tools during both development and deployment, as well as close collaboration between them. This allows organizations to deliver value faster and in a continuous manner [12]. DevOps encompass various development phases, as shown in Figure 2.1. In this study, we hope to identify security techniques or tools that may be adopted to agile development, where applicability to the development phases of DevOps is of particular relevance, enabling what is often referred to as "DevSecOps" [13]. Also relevant to this is the work of Monsalve *et al.* [11], who show the importance and relevance of adopting DevOps to develop solutions for Industry 4.0 (Industrial IoT).

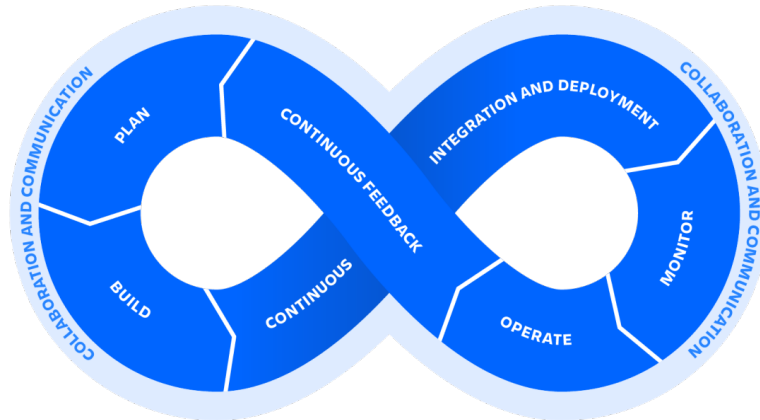


Figure 2.1: 6 phases are represented in the "DevOps Life-cycle" [14]. The development phases to the left are related to development teams, while the phases on the right side are for operation teams. The phases are encircled with the principle of close collaboration and communication.

Source: Atlassian: What is DevOps? [14]

2.3 Approaches and techniques for Security Testing

Many approaches exist for performing security testing of software, and a good overview of different types is presented by Felderer *et al.* [1], which we describe briefly in the following subsections. When testing IoT, other approaches also exist, such as setting up a testbed that is based on dynamic analysis/penetration testing or a hybrid approach. In security testing, one often differentiates approaches based on how much information the tester has about the System Under Test (SUT).

Black-box Testing

In black-box testing, the tester tests the system from the outside, in a setup comparable to the setup an actual attacker would have. This means that the tester only has basic or no information about the system, and only interacts with the system through public interfaces [1, 15].

Gray-box Testing

In gray-box testing, the tester has access to a limited information about the system, which could be knowledge of the target network or the number of hosts [15].

White-box Testing

In white-box testing, the tester has access to detailed information about the system, such as the source code, network architecture, or other development artifacts [1, 15]. This helps guide developers to locate the vulnerable code.

2.3.1 Model-based Security Testing

Model-based Security Testing (MBST) use requirements and models of the system (or its environment), that have been created during the analysis or design phase, as input when performing the testing. The input artifact is a model, and not the actual system under test. Generating security tests for MBST is based on different types on input models: *architectural and functional models* (security requirements and their implementation), *threat, fault and risk models* (causes and consequences of system failures, weaknesses or vulnerabilities), and *vulnerability models* (describe weaknesses or vulnerabilities). *Risk-based Security Testing* is a variant of MBST, where the approach is based on a risk model, such as a threat model, and it explicitly considers the risks when performing the phases of the test process. [1].

2.3.2 Code-based Security Testing and Static Analysis

Code-based security testing and static analysis bases itself on the source code and byte code generated during development. This includes manual code reviews and static code analysis, also called Static Application Security Testing (SAST). Symbolic execution is a form of static code analysis. This form of testing is important, as it can catch vulnerabilities at an early stage of development, where fixing them is still cheap [1, 16]. This is considered a white-box approach. In this category, the authors in [1] only consider approaches that don't require an executable test system, but hybrid approaches, such as concolic testing, do exist.

2.3.3 Penetration Testing and Dynamic Analysis

Penetration testing and dynamic analysis encompasses all testing done on running systems, either in a test or a production environment. It is commonly a combination of manual testing and using automated tools, such as vulnerability scanners [1]. Since it requires a running, functioning, system, it usually is performed very late in the development cycle. Dynamic Application Security Testing (DAST) fall under this category. Penetration testing is often considered as a black-box approach, but white-box and gray-box versions are also possible.

Fuzzing

Fuzzing is a popular and effective automated software testing method for detecting vulnerabilities in software [17], that is based on the idea of feeding random data to a program until it crashes. A common challenge in fuzzing is how to

make it more efficient for finding potential vulnerabilities, and so many kinds of fuzzing exist today. Approaches include random fuzzing, mutation-based fuzzing, generation-based fuzzing and advanced fuzzing techniques that combine different kinds of fuzzing [1]. Performing fuzzing generally requires a significant amount of resources, both in form of powerful (often dedicated) hardware, and time, as most approaches generate hundreds or often thousands of test cases that all need to be checked.

Concolic Testing

Concolic testing is the term for a hybrid approach where symbolic execution is combined with dynamic testing. For example combining symbolic execution with a fuzzing approach [1].

2.3.4 Testbeds for IoT

Testbeds for IoT is not a testing technique per se, but are used to monitor the behavior of a device during security testing in a controlled environment. In addition to the device(s) under test, supporting software and hardware are usually added to allow fine-grained capturing of traffic from the device, or to perform the security testing itself. The use of shielded rooms, which protects the devices from hazards or frequency disturbances, is often used to better control the environment.

2.4 Related work

Cortéz *et al.* [18] gathers testing approaches found in 79 studies on testing of IoT, in order to answer what types of testing is most commonly found in the IoT field. Relevant to this study, Cortéz *et al.* finds that "Performance" testing is the most adopted testing type (performed in 27.37% of studies), while "Security" testing was only performed in 5 of the 79 studies included. "Security testing ... [is] an essential type of test for IoT applications" [18], leaving a question for the reader as to why it is rarely reported in the IoT field. In the context of our study, we will complement the study of [18] with a focus on literature relevant to the field of security testing, and perhaps spark an increased interest in this research area.

Macedo *et al.* identified security challenges of IoT in a systematic literature review [19], synthesizing issues reported in 131 studies on security in IoT. In a systematic review by Lu *et al.* [20], the authors propose a taxonomy for cyber-attacks targeting IoT, as well as an assessment of current research trends for IoT. The latest security trends for IoT reported in [20] include "Cloud-service security", "5G" and "Quality of Service-Based Design". Though these studies do not target security testing specifically, they both echo the need of applying security holistically, also including the low-end capacity devices often found in IoT.

Chapter 3

Research Methodology

In this chapter we describe the research methodology for this Systematic Literature Review (SLR), which we conducted according to Kitchenham's guidelines [21]. According to the guidelines, an SLR consists of three phases: planning, conducting, and reporting the review. In this chapter we describe our the research questions and review protocol (which includes a search strategy, inclusion and exclusion criteria, and quality criteria), which were created as a part of the planning phase. We also describe our study selection process, development of a data extraction form, and strategy for synthesizing the results, which were done as a part of the conducting phase.

3.1 Research Questions

The purpose of this study is to gain insight into security testing for IoT by systematically studying approaches in literature, in order to guide practitioners wanting to adopt security testing when developing IoT systems. The research questions for this SLR are the following:

- **RQ1 Approaches:** What are the approaches for security testing in IoT?
 - **RQ1.1 Categorizing:** What are the characteristics of the identified security testing approaches?
 - **RQ1.2 Application:** At what level in the development process are the approaches applicable?
 - **RQ1.3 Generating:** What is needed to generate test cases and prepare the approach?
 - **RQ1.4 Executing:** What skills, methods or tools are required to perform the approach?
 - **RQ1.5 Reporting:** How are the results of the approach reported?
- **RQ2 Automation:** To what degree can the approaches be automated?
- **RQ3 Adoption:** What is the ease of adoption of the approaches found?
 - **RQ3.1** How easily adoptable are the approaches to agile teams?

- **RQ3.2** To what extent does the approaches support prioritization of test cases?
- **RQ3.3** To what degree does the approaches support scoping of test areas?

We used Tuma *et al.*'s SLR [22] as inspiration for structuring of the research questions, and for the wording of RQ1 and RQ3. In the rest of this section, we give background for each research question.

RQ1 Approaches: What are the approaches for security testing in IoT?

We have organized RQ1 into five inquiries to describe the approaches found.

Categorizing (RQ1.1). This question will help us characterize what kind of security testing approaches are described in literature, which allows us to get an overview of the field.

Application (RQ1.2). Knowing when the approaches can be used in the development process can be helpful for practitioners to understand where in their workflow the approach will fit in, and also for researchers to identify gaps in research.

Generating (RQ1.3). This question looks at what is needed to prepare the approach, and what is needed as input to generate the test cases. If this is well-defined, it is easier for practitioners to judge whether they have what is required to use an approach.

Executing (RQ1.4). This question looks at who the target audience for the approach is, and the methods and tools used to perform it. The target audience refers to the ones who has the skills necessary to perform the tests, which could be e.g. developers, security experts, or test engineers. Less skills required could indicate that the approach is easier to adopt.

Reporting (RQ1.5). This question looks at what the approach gives as output (e.g., risks, bugs), and how the results from each approach are reported (e.g., pdf, csv, plain text). Who can make use of and understand the results, are also of interest. Knowing what output the approach gives is important for practitioners in determining whether or not it will be useful in their case.

RQ2 Automation: To what degree can the approaches be automated?

Here we are interested in to what degree the approaches are already automated, from test generation and test execution, to reporting of results, and their potential to become automated. Approaches that gives its results in a structured form (e.g. csv, json) are easier to feed into other systems for further processing, and thus decrease the amount of manual work. Obstacles to automate steps reported in literature are also of interest.

RQ3 Adoption: What is the ease of adoption of the approaches found?

How easily adoptable an approach is depends on many factors, including how much preparation the approach needs, how expensive it is to set up, if you need experts to setup or use it, if the approach scales well, on tool support, and on documentation and guidelines available. Tool availability is a good indicator of technique maturity [22], and tools that are either open source or commercialized are easier to make use of than prototypes. Approaches that are easier to adopt while still providing useful results, could be low-hanging fruits for teams to increase the security of their IoT system without too much effort. Answering this question could also uncover approaches that are harder to implement, but also could increase the security of the product significantly, and thus could be a worthwhile investment.

RQ3.1. Agile teams, adopting frameworks like Scrum or DevOps, value small product increments and less up-front planning, and are less process-oriented than traditional "waterfall" teams. Thus, approaches that can be integrated into their existing workflow without adding too much complexity or takes too much time to perform, will be easier to adopt than approaches that do not fit into the agile way of working. Approaches that can be automated, that for instance promote transparency between teams, are therefore good candidates, especially for DevOps that we describe in Section 2.2.

RQ3.2. Prioritizing test cases in some way could make it easier for practitioners to achieve "good enough" security, and increase efficiency of running tests which makes it possible to run them more frequently, and thus detect issues quicker.

RQ3.3. The scope of testing refers to what part of the system the approach tests, which could be the entire system or just particular modules. This can be useful information for practitioners when deciding on what approaches they need to include to cover the areas they need to test.

3.2 Data Sources and Search Strategy

Databases

Scopus and Web of Science databases were used in this study. Initially, ACM Digital Library and IEEE were also considered, but due to limitations of both libraries in search features and export functionality, these were excluded. We believe that the papers found in Scopus and Web of Science is still representative to the current state of research in the area of security testing for IoT, as papers are often shared in multiple libraries.

Preliminary Searches

A number of preliminary searches were performed in order to find promising papers based on title and abstract and also to refine the keywords used in searches. The promising papers were used as matching criteria for future searches, to verify

that the search also would include these. Various combinations of keywords such as "testing" were attempted during this phase. As a result, the final search string includes many synonyms for "testing", such as "verification", "assurance" or "assessment". It was also found that many interesting papers do not actually mention "security" in their title or abstract, but are often more specific as to what type of security testing is performed (e.g., "fuzzing"). To limit the possibility of filtering out such papers too early, it was decided to search for testing in general, and then manually verify if they relate to security in a later step.

Search String

The search strings used for both Scopus and Web of Science are presented in Table 3.1. Scopus supports additional "LIMIT-TO" metadata filters, which help narrow down the search results. For this study these were used to directly apply some of the inclusion criteria to the search results. The filters includes "subject area", where "computer science" is the only relevant, "document type" must be "article" or "conference proceedings", and the "language" must be "English".

Table 3.1: Final Search Strings

Source	Search string
Web of Science	TS=("software" AND ("testing" OR "tests" OR "assurance" OR "verif*") AND ("iot" OR "iiot" OR "internet of things" OR "industry 4.0"))
Scopus	TITLE-ABS("software" AND ("testing" OR "tests" OR "assurance" OR "verif*") AND ("iot" OR "iiot" OR "internet of things" OR "industry 4.0")) AND (LIMIT-TO (SUBJECTAREA,"COMP")) AND (LIMIT-TO (DOCTYPE,"cp") OR LIMIT-TO (DOCTYPE,"ar")) AND (LIMIT-TO (LANGUAGE,"English"))

3.3 Inclusion and Exclusion Criteria

Table 3.2 lists our inclusion and exclusion criteria. We found that it did not make a substantial difference in number of search results if only including papers after 2015 and later, so we did not include a criteria for this. We found that a significant amount of search results focused on topics outside of our research field. For this reason, we included 2 more exclusion criteria (EC4, EC5).

Table 3.2: Inclusion and exclusion criteria

Inclusion criteria	
IC1	The main focus of the article must be on security testing of Internet of Things (IoT).
IC2	The article must be written in English.
IC3	The article is not a summary, blog post, report, or chapter in a book.
IC4	The article is published in either a journal or a conference proceedings.
IC5	The full version of the article must be available through institutional access or provided for free.
Exclusion criteria	
EC1	IoT is only mentioned as an example.
EC2	Testing is only mentioned as an example.
EC3	The article is shorter than 5 pages.
EC4	Papers on remote attestation, secure updates, networks, and similar.
EC5	Papers outside the field of computer science

3.4 Quality Criteria

Table 3.4 presents the quality criteria used in this study. According to Kitchenham [21], there is no agreed definition of study "quality". In this study, quality criteria are presented in a checklist consisting of "quality items" [21]. They are used to weigh the relevance of a particular study based on its study design [21], and in turn strengthen the validity of the study and minimize bias. Applying quality criteria is done for each paper in the last part of this study's study selection process, presented in Section 3.5.

For applying the quality criteria, we echo the method used in the SLR of Macedo *et al.* [19], where each criterion is answered using one of the options on the measurement scale, *no*, *yes*, or *partially*. Allowing for "partially" is important to support nuances, since giving a straight yes/no answer might be misleading for some papers. Upon summarizing the total score of a paper, each quality criteria answer is given points: 0.0 points for "no" answers, 0.5 points for "partially" answers, and 1.0 points for "yes" answers. This allows for a numerical assessment of quality, where a paper should only be included if the score is equal to or above 4.0 points (of 6.0 available), as shown in Equation (3.1). This corresponds to at least 66% of the available points, similar to the threshold used in [19]. We emphasize that studies that do not meet our quality criteria are not poor papers, but failed to meet the quality requirements which are based on the research questions

of this study.

$$QC_1 + QC_2 + \dots + QC_6 \geq 4.0 \quad (3.1)$$

Table 3.3: Quality criteria

Quality criteria	
QC1	Is the aim of the research and context well described?
QC2	Is the research design prepared sufficiently?
QC3	Is the approach for testing described sufficiently?
QC4	Does the approach seem feasible to implement?
QC5	Does the results support the ideas of the paper?
QC6	Does the paper discuss limitations or validity?

3.5 Study Selection Steps

An illustration of our study selection steps is shown in Figure 3.1. This illustration was inspired by Tuma *et al.*'s SLR [22]. The first step in this process was to perform searches in the selected digital libraries. The search for papers was performed in September 2021 using the search strings listed in Table 3.1. This resulted in 1126 results in Scopus, and 661 result in Web of Science which were exported to comma-separated files (CSV). These files were then imported into Zotero [23], a reference manager software. Zotero automatically identifies duplicates (based on title, DOI, and ISBN), but manual effort was needed in order to merge the duplicates. During merging, only the versions found in Web of Science were selected, after verifying that the duplicates were in fact duplicates. Selecting Web of Science versions was a matter of convenience of fewer click interactions per merge. Both Scopus and Web of Science had included the most relevant metadata from the references (title, abstract, authors and keywords). 452 duplicates were removed in the first filtering step.

In the second filtering step, inclusion/exclusion criteria were applied to 1335 papers by reading through the abstract and title of each paper. In this process, we labeled the papers in Zotero with *OK* for inclusion, and *NOT_RELEVANT* for exclusion. If the paper was excluded, a reason was given using either (or both) of the labels "*_NO_TEST_FOCUS*" or "*_NO_IOT_FOCUS*". This work was split equally between the two researchers, and a third label "*_NOT_SURE*" was used for papers that required further discussion. As the total number of pages were missing from the metadata of most papers, the exclusion criteria (EC3) was applied in a later step where the full text was available. 1222 papers were excluded in this step.

In the third filtering step, both researchers worked collaboratively on labeling the papers with *SECURITY* based on the type of testing reported in the paper. In cases where it was unclear, the full paper text was used to determine if the paper

was relevant for security testing or not. 2 supervisors assessed the labeling, and both included, and excluded a few papers previously marked as security related. This led to 81 papers being excluded in this step, most of which did not have security testing in focus.

In the final filtering step, the full text of each of the 31 papers remaining was evaluated using quality criteria. This was done simultaneously as data extraction in order to better justify the points given to each criterion. This work was split equally between the researchers, and cross-checked by the other researcher simultaneously as the data extraction form was checked. In this step, the exclusion criteria (EC3) was also applied. The final list of papers used in this study is 21, where 10 of the studies either failed the quality criteria or were too short to be included.

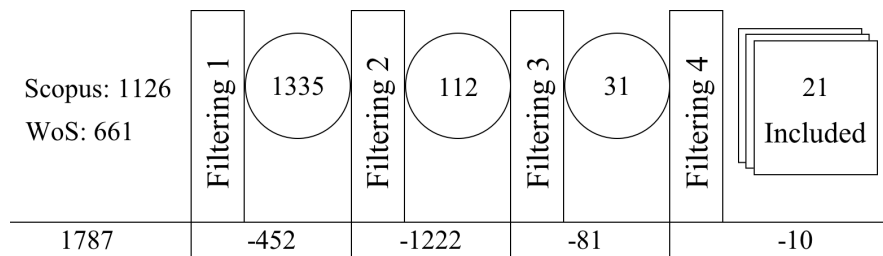


Figure 3.1: Study selection process used in this study.

Table 3.4: An explanation of each filtering step in the study selection process.

Filtering steps

<i>Filtering 1</i>	Removing duplicates
<i>Filtering 2</i>	Apply inclusion / exclusion criteria to title and abstract of papers
<i>Filtering 3</i>	Based on keywords, title and abstract, include only papers with focus on security testing of IoT
<i>Filtering 4</i>	Apply quality criteria to full paper

3.6 Data Extraction and Data Synthesis

Table 3.5 shows the relevant information extracted from each paper in order to answer the research questions. In total we did the data extraction on 31 papers, after which we excluded 10 due to applying more exclusion criteria and quality criteria. P11 failed EC3 due to being too short, but a decision was made to still include it due to its relevance to the research questions. The other papers that were too short, also failed the quality criteria, so no exception were made for these.

Table 3.5: Relevant information extracted from the papers in order to answer the research questions

Data Extraction Fields	
RQ1	Approach for security testing, abstraction level of approach, preparations needed, method for generation of test cases, methods and tools used for performing approach, target audience, output, output report format, target audience for output.
RQ2	Degree of automation, comments of possibilities for automation.
RQ3	Approach tool maturity, expertise needed for entirety of approach, configuration needed, comments on scalability, comments on relevance for agile, scope of testing, prioritization of tests.

After data extraction, we did the data synthesis on 21 papers, which are included in the results.

Chapter 4

Results

This chapter contains the results of our findings for answering the research questions proposed in Section 3.1. In total we included 21 papers from 2018 to 2021. Table 4.1 shows an overview of the papers. Figure 4.1 shows the distribution of publication years, and we note that since it can take time to get a paper published, the distribution does not necessarily indicate a decrease in research in 2021.

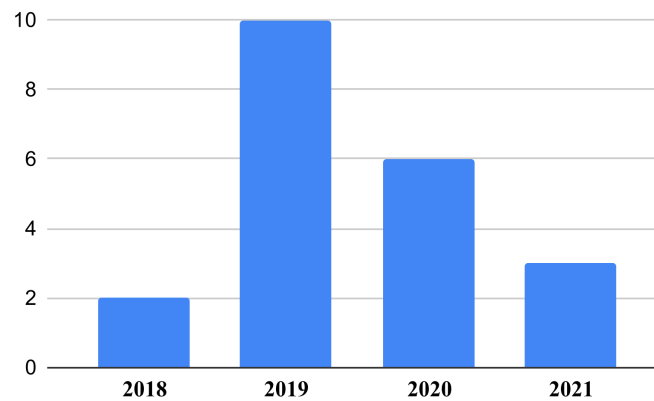


Figure 4.1: Publication years of selected papers.

Table 4.1: Overview of the selected papers

Code	Title	Authors	Approach for security testing
P01	A Dynamic Analysis Security Testing Infrastructure for Internet of Things	Wang <i>et al.</i> [24]	DAST using network capturing methods
P02	Attack Surface Modeling and Assessment for Penetration Testing of IoT System Designs	Mahmoodi <i>et al.</i> [25]	Penetration testing and attack-surface modelling
P03	Automated and on-demand cybersecurity certification	Karagiannis <i>et al.</i> [26]	Automated security audits to certify components
P04	AVRS: Emulating AVR microcontrollers for reverse engineering and security testing	Pucher <i>et al.</i> [27]	Fuzzing of AVR firmware
P05	Bug detection in embedded environments by fuzzing and symbolic execution	Vijtiuk <i>et al.</i> [28]	Symbolic execution and fuzzing of software
P06	Detecting Dynamic Security Threats in Multi-Component IoT Systems	Shrestha <i>et al.</i> [29]	Multi-component threat analysis using audit hooks
P07	EcoFuzz: Adaptive energy-saving greybox fuzzing as a variant of the adversarial multi-armed bandit	Yue <i>et al.</i> [17]	Coverage-based gray-box fuzzing
P08	Finding Sands in the Eyes: Vulnerabilities Discovery in IoT with EUFuzzer on Human Machine Interface	Jiaping <i>et al.</i> [30]	Black-box mutation-based fuzzing
P09	FIRM-AFL: High-Throughput Greybox Fuzzing of IoT Firmware via Augmented Process Emulation	Zheng <i>et al.</i> [31]	Coverage-based gray-box fuzzing
P10	Hybrid Firmware Analysis for Known Mobile and IoT Security Vulnerabilities	Sun <i>et al.</i> [32]	Binary code similarity analysis
P11	Integrating Threat Modeling and Automated Test Case Generation into Industrialized Software Security Testing	Marksteiner <i>et al.</i> [33]	Automated security testing through risk-analysis-enhanced threat modelling
P12	IoT Testbed Security: Smart Socket and Smart Thermostat	Bettayeb <i>et al.</i> [34]	Easy-to-setup testbed for security assessment
P13	Iotverif: Automatic Verification of SSL/TLS Certificate for IoT Applications	Liu <i>et al.</i> [35]	Constructing model from runtime communication and checking it for vulnerabilities
P14	Mirage: towards a Metasploit-like framework for IoT	Cayre <i>et al.</i> [36]	Framework for security audits and penetration testing
P15	P2IM: Scalable and hardware-independent firmware testing via automatic peripheral interface modeling	Feng <i>et al.</i> [37]	Fuzzing with randomized input of micro-controller firmware
P16	Penetration Testing of Intrusion Detection and Prevention System in Low-Performance Embedded IoT Device	Zitta <i>et al.</i> [38]	Penetration testing
P17	Requirements and Recommendations for IoT/IIoT Models to automate Security Assurance through Threat Modelling, Security Analysis and Penetration Testing	Ankele <i>et al.</i> [15]	Using metadata from common diagrams and models to automate security assurance
P18	Security Testbed for Internet-of-Things Devices	Siboni <i>et al.</i> [39]	Generic testbed performing standard and advanced security tests
P19	Techniques to Improve Reliability in an IoT Architecture Framework for Intelligent Products	Coman <i>et al.</i> [40]	Securing code and device at many layers
P20	Threat Analysis for Wearable Health Devices and Environment Monitoring Internet of Things Integration System	Tseng <i>et al.</i> [41]	Threat modelling and penetration testing
P21	Z-Fuzzer: Device-agnostic fuzzing of Zigbee protocol implementation	Ren <i>et al.</i> [42]	Improved grammar-based fuzzing through coverage heuristics

4.1 RQ1 Approaches

4.1.1 RQ1.1 Categorizing

Table 4.2 shows the approaches found, categorized by the testing approaches described in Section 2.1. A large portion of our selected papers were on fuzzing, so to emphasize that, we separated those from the rest of the papers in the penetration testing and dynamic analysis category. We based the categorization (unless explicitly mentioned in the paper) on the tools used to perform the approach. For instance, the use of tools for network analysis, indicates that this approach uses dynamic analysis. We found no papers that focused on code-based testing and static analysis.

P10 can be categorized as concolic testing, as it is a hybrid approach for doing binary code similarity analysis, combining both static and dynamic approaches, where the dynamic part employs fuzzing to generate different input sets. However uncertainty exists, since the authors do not mention it themselves.

P12 and P18 employ a variety of penetration testing and dynamic analysis techniques in their testbed setups.

P17 could not be mapped, as it is a high level approach discussing the requirements for metadata in diagrams, with the aim of enabling automation of security assurance.

P19 employs many approaches for securing IoT, including automatic penetration tests. The other approaches mentioned in P19, could not be mapped to our categories, as they cover measures like embedded code protection, physical hardening, cryptographic acceleration, monitoring for vulnerabilities and regularly updating dependencies.

Table 4.2: Each paper mapped to a type of approach.

	Black-box	Grey-box	White-box
Model-based Security Testing (3)			P02, P11, P20
Code-based Testing and Static Analysis (0)			
Penetration Testing and Dynamic Analysis (7)	P01, P13, P14, P16, P19		P03, P06
Fuzzing (8)	P04, P08, P10	P05, P07, P09, P15	P21
Testbed (2)	P12, P18		

We found 5 papers that use Machine Learning (ML) to further enhance capabilities in their security testing approaches. Some papers use ML for the purpose of analyzing suspicious behavior during monitoring, using various collected data,

such as method input-parameters (P06) or network traffic (P18, P19). In P10, deep neural networks is used to automatically assess vulnerabilities in firmware binaries based on previously reported CVE's. P07 uses a variant of the adversarial multi-armed bandit algorithm for optimizing the coverage-based grey-box fuzzing strategy used in AFL [43], reportedly improving its path-coverage by 214% with 32% less test case generations required.

4.1.2 RQ1.2 Application

Table 4.3 distributes the approaches found during data extraction to each of the development phases described in Section 2.3. The table shows that the majority of approaches should be applied when the developed system is in a test environment. We found no papers that could be applied during application development.

Table 4.3: Each paper mapped to a development phase.

During Planning and Design (3)	P02, P11, P17
During Application Development (0)	None
Executable in a Test Environment (17)	P01, P03, P04, P05, P07, P08, P09, P10, P12, P13, P14, P15, P16, P18, P19, P20, P21
System Operation and Maintenance (1)	P06

4.1.3 RQ1.3 Generating

In this section we present our findings for answering what is required in order to start using the approaches. Three categories were identified to generalize the actual preparations needed. In Table 4.4, we detail the categories identified with examples. To highlight that some approaches has high demands in certain preparation categories, we found it necessary to include a "level" of preparations to assign each category. The levels used are described in Table 4.5. The examples in each level are used as a reference for determining the preparation level of each approach. In Table 4.6, each approach is given a score based on the sum of preparation level scores from each category. Lower scores indicate that less preparation is needed for the approach. From the results in Table 4.6, we found that approaches that were applicable during the planning and design phase need a high amount of upfront information (P02, P11, P17).

Table 4.4: Explanation of categories for preparations needed.

Category	Description	Example
Hardware	Physical computer devices required to <i>support</i> the testing of an IoT device	Computer with 500GB SSD storage, NVIDIA GTX1080 graphics card, WiFi-router
Software	Installation of packages, operating systems or programs required for execution	Ubuntu 18.0, KaliLinux, Python 3.0
Information	Input to the testing approach is required	A valid HTTP-request, model of the system in a specific XML-format, configuration of input parameters

Table 4.5: Levels of required preparation with examples of preparations needed in each level. Cases are marked with - if it was difficult to identify preparations needed, or it was not applicable.

Level	Hardware	Software	Information	Score
None	No preparations required			0
Low	1 device	1 software installation	1-4 data fields in 1 document	1
Medium	2 devices	2-5 software installations	5-10 data fields in 1 document	2
High	3+ devices	6+ software installations	10+ data fields in 2+ documents	3

Table 4.6: Preparations score needed for each approach. A lower score means less preparations are needed. 0-3 is Low, 4-6 is Medium, 7-9 is High. The score for each paper has been calculated according to Table 4.5. Approaches where preparations needed is not applicable or could not be placed, are marked with -. We found that approaches that were applicable during the planning and design phase need a high amount of upfront information.

Code	Hardware	Software	Information	Result (Score)
P04	Low	Low	None	Low (2)
P07	Low	Low	Low	Low (3)
P08	Low	Medium	None	Low (3)
P09	Low	Low	Low	Low (3)
P10	Low	Low	Low	Low (3)
P15	Low	Low	Low	Low (3)
P17	None	None	High	Low (3)
P02	Low	None	High	Medium (4)
P05	Low	Medium	Low	Medium (4)
P06	Low	None	High	Medium (4)
P14	Medium	Low	Low	Medium (4)
P21	Low	Low	Medium	Medium (4)
P11	Low	Low	High	Medium (5)
P16	Medium	Medium	Low	Medium (5)
P01	High	Medium	Low	Medium (6)
P12	High	Medium	Low	Medium (6)
P03	Medium	Medium	High	High (7)
P13	Medium	High	Medium	High (7)
P18	High	High	Low	High (7)
P20	Medium	Medium	High	High (7)
P19	-	-	-	-

4.1.4 RQ1.4 Executing

Figure 4.2 shows the skills needed to set up and perform the approaches. The skills needed to understand the output from the approaches is considered in Section 4.1.5. Due to a lack of time, we do not detail the methods and tools found in the approaches, even though this was interesting to answer the research question.

How we separate the role of a *Security Expert* and *Developer* is shown in Table 4.7. Some papers require both security expertise and developer skills to fully make use of the approach or output, and have been categorized as *Security trained developer*. This categorization is based on the work of Tuma *et al.* [22], where a *Security*

trained engineer (re-phrased to "Security trained developer" in this study), is considered to "possess an active knowledge of security related concepts" [22], but not as extensive knowledge as a security expert.

Table 4.7: Explanation of expertise roles.

Role	Description	Example
Security Expert	Expert knowledge in security threats and vulnerability analysis.	Assess vulnerabilities found, understand how insecure code can be exploited
Developer	Expertise in code development and basic system operations skills (server setup, networking).	Set up a computer with Kali Linux and enable ARP spoofing on local network. Create test cases using Python. Provide code-fixes for vulnerabilities found.
Security Trained Developer	A combination of developer skills and expert knowledge on security is needed.	Integrate a DAST tool, and assess which modules are most likely to include potential security vulnerabilities to guide tool usage.

We can see from Figure 4.2 that 50% of the approaches are possible to be performed by a normal developer who does not have any particular security knowledge. 20% of the approaches require a security expert, an 30% require developer skills in combination with some security know-how.

The recommendations found in P17 may be used by System Architects to include data into system models that enables automation of threat modeling or future security activities.

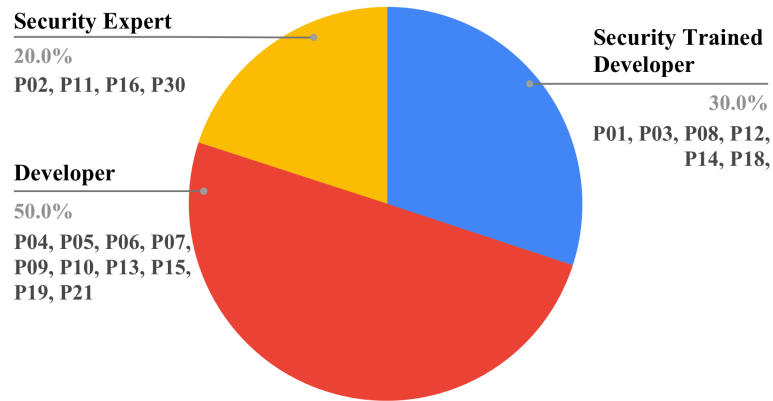


Figure 4.2: Overview of the required skills needed to perform the approach (not including analyzing the output)

4.1.5 RQ1.5 Reporting

Table 4.8 shows the outputs from each approach, which come in the form of bugs, risks and vulnerabilities. From this we can see that fuzzing approaches mainly output bugs.

Figure 4.3 shows each paper mapped to the type of report file output that it gives. We classify output that has to be parsed in order to be used further in systems (e.g. pdf, html, txt, pcap), as *unstructured output*. And we classify output that is codified and can be directly input into another system (e.g. json, csv, xml) as *structured output*. Papers where the approach output is not mentioned explicitly, are marked as such.

From Figure 4.3, we can see that 57% of the papers don't mention what kind of file output the approach uses to report the results. 14% outputs the results in a structured format, 19% in unstructured formats, and 10% use both structured and unstructured file outputs.

Table 4.8: The output from each approach

Code	Bugs	Risks	Vulnerabilities
P01		•	
P02			•
P03		•	•
P04	•		
P05	•		
P06		•	
P07	•		
P08	•		
P09	•		
P10			•
P11			•
P12		•	•
P13			•
P14		•	•
P15	•		
P16			•
P17			
P18		•	•
P19		•	•
P20		•	•
P21	•		•

Table 4.9 shows the target audience(s) for the output of each approach. In addition to the definitions of a *Developer* and *Security Expert* given in Table 4.7, we explain the terms *Test Engineer* and *Compliance* here. A *Test Engineer* is concerned with the execution and generation of manual test cases, verifying that the product works according to its specifications. Approaches that are useful for *Compliance* are intended for regulators performing auditing, by supporting the process of verifying that a device under test is behaving according to its specifications.

The output from P02 is intended to be used by System Architects or System Designers during planning or design phases. The System Architect role was omitted from the table to reduce cluttering, as this was only relevant for one paper in the output.

P05 mentions that the output is often crashes, not vulnerabilities. While only developer expertise is needed to perform the approach, security expertise is needed to check if a crash is exploitable.

P06 reports its result as audits sent to a centralized service. This output may be

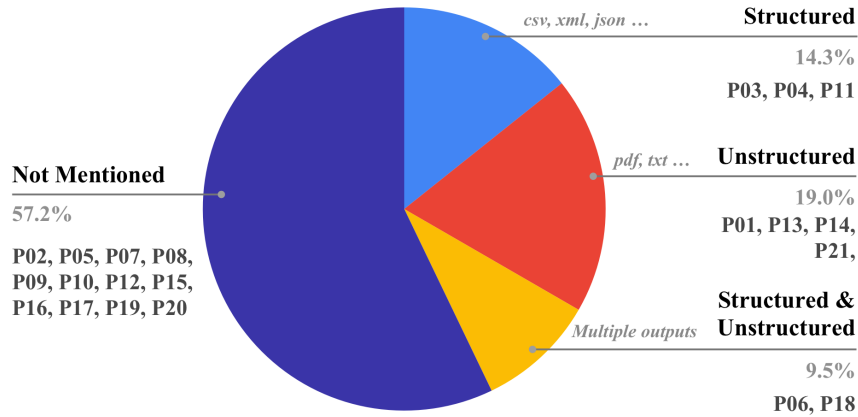


Figure 4.3: Types of report file outputs from each approach.

used internally for training ML models. Though we categorized its target audience as *Developer* and *Security Expert*, external systems may also be the target audience of this approach.

P10 outputs vulnerable functions in the IoT firmware, that has been linked to CVEs.

P16 does not give a common report, as the approach use several different penetration testing tools where all of which give different outputs.

P18 presents the most details of the output from their testbed, than any other included paper. All the tools provided by the test bed generates reports in different file formats. In addition, the Orchestrating Machine presented, combines all the reports from the tools into one single report to present to the user in the format that they request, where they mention that PDF is one of them.

P20 provides a strategy for combining security testing tools. The results from the approach can help guide test engineers in creating new test cases.

P21 returns call stack traces for each memory crash, which the authors use to manually analyze the affected functions in the source. The authors also say that the output includes the test case which triggered the crash.

Table 4.9: Output target audience for approaches.

Code	Developer	Security Expert	Test Engineer	Compliance
P01		•		•
P02				
P03	•			•
P04	•	•		
P05	•	•	•	
P06	•	•		
P07	•	•	•	
P08	•	•	•	
P09	•	•	•	
P10	•	•		
P11			•	•
P12	•	•		
P13	•			•
P14		•		•
P15	•	•		
P16		•		
P17				
P18	•	•	•	
P19	•			
P20	•	•	•	
P21	•	•		

4.2 RQ2 Automation

Figure 4.4 shows an overview over the degree of automation in the approaches for security testing that the papers present. We describe how we define each of the categories *Manual*, *Partly automated* and *Automated* in Table 4.10.

P05 presents 8 open source fuzzers, and 1 symbolic execution tool, which are all automated tools.

P06 is an example of a partly automated approach. It uses audit hooks to gather data for analysis, and while the analysis of the data is fully automated, acquiring the datasets of "normal" behavior is partly automated as you have to orchestrate the data collection yourself. Developers have to manually insert audit hooks into the code where they want to collect data from, and calibrate the system by gathering new datasets every time these change.

P11 is also partly automated, as they manually do threat modelling, and manu-

Chapter 4: Results

27

ally creates test cases based on this, but the execution of the tests is automated. This paper mentions that it has the potential to automate the test case generation based on CVEs, since the output of the threat modeling is XML.

P14 is a partly automated approach, that is a Metasploit-like security audit framework specifically for IoT. Creating the test scenarios is a manual process, as one uses the framework to first to evaluate the attack surface. Attacks can be constructed by chaining together different modules, so for example, one could run an entire Man-in-the-Middle attack in an automated manner, once it has been set up (some existing attacks are included in the framework). However the reporting and analysis of results to determine if the device is vulnerable, is still a manual process.

P17 presents an approach to automate security assurance by enriching commonly used diagrams and models from the software development process, with metadata that security testing tools require as input. In this way, one does not have to create a detailed threat model (if it does not already exist), which many earlier approaches for automation require. This paper also mentions that not all the steps of security tests can be automated, and that these will still require experienced security auditors and cryptographers to perform.

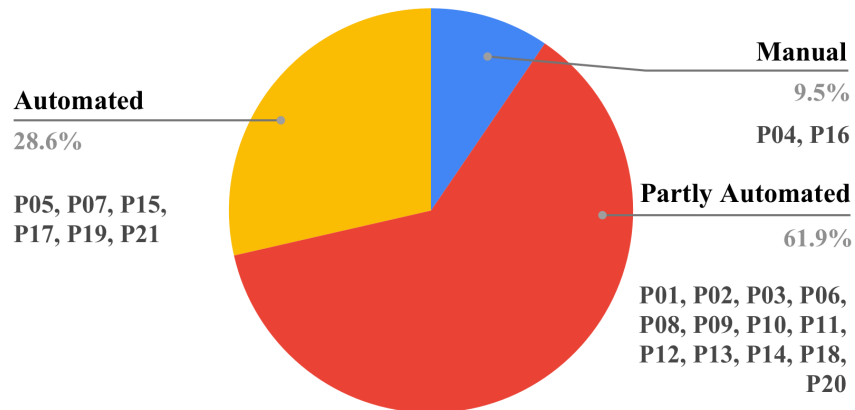
P18 is a partly automated approach, in the form of a testbed that can run both standard security tests (e.g. spoofing attacks, vulnerability scan, fuzzing) using off-the-shelf tools, and advanced security tests (e.g. using ML algorithms to identify device type and detect suspicious behavior, test resilience to DoS-attacks, test password complexity). Most of the standard tests appears to be able to be run in an automated manner, while a few of the advanced tests might require some manual steps during the testing process (e.g. physically interfere with device), and if no clear pass/fail of a test can be determined, it supports manual analysis. The authors mention as future work that they want to further automate the testing process.

P19 is an example of an automated approach, where the penetration tests (provided by a commercial service [44]) performed were fully automated. They set up Gitlab CI for the project, to automatically test and build new software versions.

P21 is another automated approach, in the shape of a grammar-based fuzzer for Zigbee protocol implementations, where each test case is executed at runtime in a simulator.

Table 4.10: Explanation of levels of automation

Level	Description	Examples
Manual	Fully manual job	Generating the test cases and executing the tests is a manual job.
Partly automated	Some manual effort required	Generating test cases is manual, but executing them is automatic. At some point during test execution, a human is required to advance the test.
Automated	Minimal manual effort required	The entire test generation and test execution is automatically done, all the way up to the final report being generated and shown to the user, or its output is automatically fed into another process.

**Figure 4.4:** Degree of automation of the approaches.

4.3 RQ3 Adoption

Figure 4.5 show the expertise needed to perform the approach as a whole, which includes the skills needed to perform the approach, as well as fully understanding the output. From Figure 4.5, a *Developer* could perform 24% of the approaches, 14% of the approaches requires a *Security Expert*, while 62% could be done by a

Security Trained Developer. It is worth noting that a *Security Trained Developer* is able to perform the approaches requiring a *Developer*, which gives a total of 86% of the approaches that could be performed by a *Security Trained Developer*.

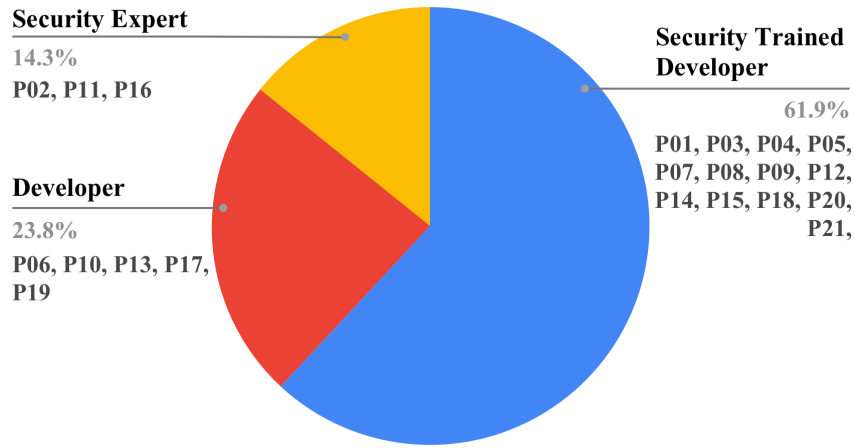


Figure 4.5: Expertise needed to make full use of the approach, including steps to perform it, and understanding the results.

Figure 4.6 shows the an overview of the maturity of the tools presented in the approaches. Table 4.11 describes how we categorized the levels of maturity of the tools presented in the approaches.

As show in Figure 4.6, we found that 60% of the tools presented in the approaches are available as open-source tools. 35% of the approaches were not available, and are only presented as prototypes, and 1 paper intended to make the tools in the approach available as a service to in the future. We did not categorize P11, as the paper contained too little information about their setup to determine correctly.

Table 4.11: Explanation of levels of tool maturity used for ease of adoption

Level	Description
Prototype	The tool(s) presented in the approach is described, but not available. It may or may not be based on open-source tools.
Open source	The tool(s) are available for open-source use and modification.
Proprietary	The tool(s) are available as a commercial software (or planned as so) that can be licensed to users.

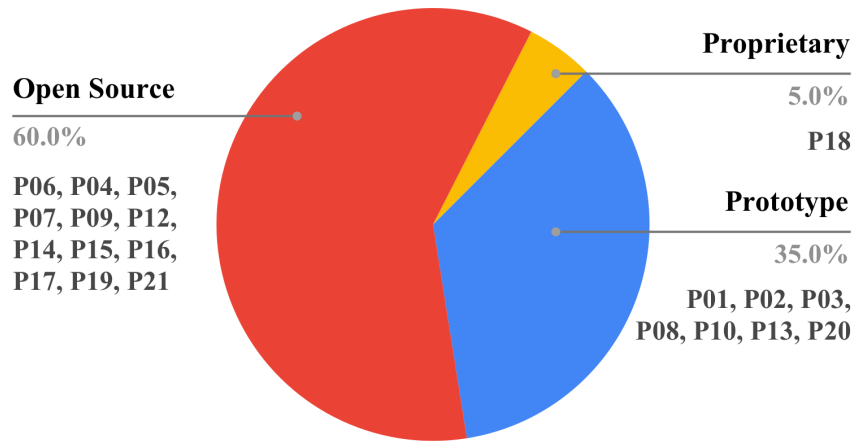


Figure 4.6: Maturity of the tools presented in the approaches.

4.3.1 RQ3.1 Ease of adoption to agile teams

To answer RQ3.1 *How easily adoptable are the approaches to agile teams?* we found only a few approaches relevant to the *Monitoring* and *Deployment* phases of DevOps, as described in Section 2.2. In fact, only one paper (P06) explicitly mention its relevance to agile development. In the following subsections, we elaborate on approaches that could provide beneficial during deployment or monitoring of IoT systems.

Deployment

The "cyber-security certification process" described in P03 may be used during service deployment to ensure the service is "compliant" by running vulnerability detection and compliance checking using rulesets derived from CVE databases and GDPR rule sets. Custom rulesets can also be defined using XML. Integrated into the Continuous Integration (CI) pipeline, P19 fully automates its penetration testing efforts to ensure reliability of software prior to releasing new versions.

Monitoring

For continuous auditing, P03 and P06 both rely on sending data to a centralized logging service. While P03 audits installed packages, program versions and currently running services on the devices, P06 audits parameters sent to methods inside the running system. Both approaches contributes to the integrity of the running services and transparency for the development teams.

4.3.2 RQ3.2 Prioritization of test cases

We found only a few papers in data extraction that prioritize test cases in some way, or use specifications, guidelines or vulnerability databases in order to guide test case generation.

P03 use the Common Vulnerability Scoring System (CVSS), Common Vulnerabilities and Exposures (CVE)s, and GDPR rules to come up with test cases.

P11 uses a threat model to generate test cases, and use attack patterns from publicly available catalogs like CVE.

P18 uses OWASP vulnerabilities to create test cases.

P20 uses STRIDE for threat identification, and DREAD for calculating the risks to prioritize test cases.

P21 is a fuzzing tool that stores "favored" test cases, which are test cases that lead to an increase in code coverage, and then prioritize these when choosing the next test case to mutate.

4.3.3 RQ3.3 Scoping of test areas

We found that the majority of approaches do not mention any scoping of test areas, and have no strategy beyond attempting to test all code statements. The fuzz testing approaches found are usually coverage guided (P05, P07, P09, P15), meaning they aim for reaching full code coverage.

P07 presents EcoFuzz, which improves the search strategy of AFL [43]. Instead of aiming for full code-coverage, EcoFuzz aim to optimize its path traversal using an ML algorithm to identify "less" frequently traversed code.

P19 runs penetration tests on the infrastructure and the cloud web application every month.

P21 presents Z-fuzzer, where they mention that it cannot reach full code coverage due to its focus to generate high-quality test cases, which excludes exception handling code.

Chapter 5

Discussion

In this chapter we discuss the results in order to find answers our research questions.

5.1 RQ1 Approaches

In order to answer *RQ1: What are the approaches for security testing in IoT?*, we discuss the findings from each sub-research question in this section.

5.1.1 RQ1.1 Categorizing

In order to answer *RQ1.1: What are the characteristics of the identified security testing approaches?*, we show an overview of the approaches found in Table 4.1, and categorize the approaches in Table 4.2. In this section we further discuss the results.

Most of the approaches fall under the category of *Penetration Testing and Dynamic Analysis*, and *Testbeds* which make use of these techniques. These approaches require an executable test environment, which raises challenges in IoT. For IoT test environments, this implies having a working emulator (or the actual device) to run the system under test, due to the software and hardware being tightly coupled [8]. In systems relying heavily on software, such as web applications, running the systems may not pose a challenge, while the heterogeneous hardware and protocols used in IoT [36] complicates this matter. Creating simulators or emulators that support all of them is challenging. This is also mentioned as a pain point by P21, who say that it is a "significant obstacle to deal with low-level hardware events" when doing fuzzing, as the existing fuzzing tools did not have the necessary support for this. P04 reportedly succeeds in emulating firmware for AVR microcontrollers, which are often used in embedded systems. Though this effort shows that emulating is not an impossible task, AVR is but one of many architectures.

We found no mentions of approaches that focus on *Code-based testing and Static Analysis*, as defined in Section 2.3. A reason for this may be a limitation

in our search query, which limits the papers to IoT. Since by nature, IoT systems encompass devices communicating, and not specifically the software used, which this category focuses on. SAST tools may not be specific to IoT, but specific to the language used in programming IoT software. Hence, this study might have missed papers that focus on developing SAST relevant for IoT development. Performing static analysis of code is mentioned by Bachmann and Brucher to be "the most efficient and effective security testing method if only one method is applied during software development" [10], but also stressing that only performing static analysis is not sufficient on its own. Nevertheless, future research could investigate how SAST may be integrated into security testing for IoT. The implications of not finding any papers in this category, are also connected to the applicability of approaches during the *Planning and Design* phase, as discussed in Section 5.1.2. P10 used static analysis to analyze the firmware of the IoT to find any similarities between the tested binary and vulnerabilities from a CVE database. The findings from the static analysis was then used as input to a dynamic analysis engine, which verified the vulnerability on-device or in emulator to ensure it was not a false positive. Although this paper used static analysis as a part of the approach, it is not purely using static analysis, and is categorized thereafter.

P06 used machine learning to identify incoming attacks, and trained their models using data collected from system operation in a test environment. P07 improved an existing fuzzer with machine learning capabilities to enhance its performance. This suggests that the appliance of ML is relevant for IoT security testing, both in order to support analysis of device behavior during test, or as an integrated part of the security testing tool. ML is becoming widely adopted in software development, with many easy-to-use tools [45, 46]. We suggest more research to be addressing how to leverage ML both in existing security tools, and in novel security approaches, for instance to tackle the large amount of data generated by IoT systems.

5.1.2 RQ1.2 Application

In order to answer *RQ1.2: At what level in the development process are the approaches applicable?*, we look at the approaches mapped to a development phase, as shown in Table 4.3. In this section we will discuss the findings in each of the development phase in detail.

During Planning and Design

In P02, Model-based Security Testing (MBST) is performed using "virtual prototypes" simulating both the devices themselves and communication between them to enable an extensive system vulnerability analysis. The output from this approach is valuable to system designers or architects for early assessment of principles such as "least privilege", and help identify possible security flaws in design. Defects found in this phase of development is much less costly to fix [16], than later during product development, or even worse, fixing them in production. Based

on this, we propose using recommendations from P17 to further enhance threat modeling efforts reported in P11, in order to ease the automation of activities such as automatic test case generation.

During Application Development

We found no mentions of approaches that could be used during application development. A wide range of commercially-available software exists for performing SAST, such as SonarQube [47], which is commonly used for identifying code-vulnerabilities during development. SonarQube [47] supports languages often associated with IoT development, such as C and C++, and also compilers for AVR and ARM microprocessors. We found only one mention (P10) of an approach that used SAST as part of their tool set, while no approach solely focused on this test method. Very scarce results in this phase might reflect a general concern of security testing, where security is often only prioritized late in the SSDLC [1]. It is however important to enable security in all phases of development, therefore we want to direct future research into approaches that can fit into this phase of development. An interesting topic for future research in this phase is to integrate security testing into activities such as Continuous Integration (CI) or Continuous Deployment (CD). Another possibility is integrating a dependency-check tool for third-party libraries specifically for IoT, into mainstream SAST tools.

Executable in a Test Environment

Most of the approaches found requires a running test environment. As previously mentioned in Section 5.1.1, emulating devices is a challenge, which makes running the systems on a physical device often a necessity. When emulating, systems may be closely monitored, while physical devices often need "supportive" tools such as network capturing tools and network analysis to assess the implementation of security measures.

While several approaches test an IoT device that is communicating with one other device, we propose more research to test the integration of IoT devices in a larger network. The behavior of one device could affect others, which might uncover unwanted behaviors that are hard to discover when only testing one device in isolation. The audit hooks presented in P06 allow for multi-component analysis in the *System Operation and Maintenance* phase, which is the closest to this idea that we have found in the papers.

System Operation and Maintenance

In order to understand how the system operates in production, P06 makes use of a centralized logging service which receives input parameters as audits from security-related methods using a method referred to as *audit hooks*. The use of audit hooks in IoT is a novel approach which is promising in order to gain insight and direct future security efforts in a system during operation. As future research,

we echo P06 in improving the efforts of mining collected data from the audit hooks to isolate vulnerable components or to provide mitigation suggestions for the reported malicious inputs received.

Another important topic of this development phase is how to perform maintenance of IoT devices. Although we found no security testing approaches targeting maintenance in our selection of papers, ensuring integrity during maintenance of devices is important. One approach that exists for this purpose is the open-sourced PURE tool [48], extending the remote attestation capabilities of the tool VRASED [49] to ensure integrity of the device after a software update. Future research could look into managing the use of such tools on a larger scale, including functionality for ease of management such as reporting.

5.1.3 RQ1.3 Generating

In order to answer *RQ1.3 What is needed to generate test cases and prepare the approach?*, we looked at what is required in order to start using the approaches identified, as shown in Table 4.6. From data extraction, three categories emerged as relevant for preparations needed to start with the approaches, *Software*, *Hardware*, and *Information*.

Software

One interesting finding is the prominent use of security tools included in the open-source Linux-distribution Kali Linux [50] (such as *nmap*, *arpspoof*, or *Wireshark*) in most of the testbeds and security testing methods (P01, P12, P16, P18, P20). In contrast, only two studies mentions use of proprietary tools for security testing, namely Burp Suite Professional [51] in P20 and Pentest-Tools [44] in P19. Reasons for its wide adoption in our included studies could be that Kali Linux is free to use, that it includes many proficient security tools, and that it does not require much hardware to run. For practitioners wanting to start with security testing for IoT, studies such as P12 and P20 provide good experience reports on how to set up and use many of the tools included in Kali Linux.

Information

For approaches applicable during planning and design, relevant information needed includes threat templates (such as STRIDE or VAST), high-level system documentation such as communication protocols, trust boundaries or data exchanged between the system components. P17 includes specific recommendations for what information should be present in IoT models to automate security activities. For instance, on the "Smart Devices and Sensors" level, highly recommended information include hardware interfaces, IP/MAC addresses, network protocols and system information (operating system, firmware versions).

Fuzzing tools usually require some upfront information such as a valid input-parameter to guide the generation of test cases. This information may be manually

provided, or automatically, based on output from other methods as reported in P05. An exception to this, is the black-box fuzzers reported in Table 4.2, which requires only an entry-point and initial setup to begin fuzzing.

In general, we found that few papers clearly specify what input is needed to create test cases, but rather mention it in a more abstract way (e.g. a valid message format) without giving concrete examples. The lack of clear guidelines, could make it more difficult for practitioners to adopt the approach.

Hardware

Fuzzing tools generally require dedicated hardware to run. Studies such as P07 and P09 use high-end computers (e.g., 40GB/125GB RAM and 40-core CPUs) to perform testing of their approach. We were unable to find any mentions if such hardware was indeed a requirement. As fuzzers could run continuously for anywhere between a few hours to a couple of months (or 490 CPU days as reported in P08), millions of I/O operations may happen. Hence, this have an impact on the weariness of the hardware used. This makes these approaches unsuitable for running on developer machines (if intended to run for longer periods) [17].

5.1.4 RQ1.4 Executing

In order to answer *RQ1.4 What skills, methods or tools are required to perform the approach?*, we assessed the skills required to perform each approach. From the results, we derive that 50% of the approaches could be performed by developers alone. Given that development teams invest in security training, we believe that an additional 30% of the approaches may be performed. This could be as simple as making developers understand which components in the system is security critical, in order to guide their choice of security testing method. A possible threat to validity for this research question is that few of the papers actually consider who should perform the approach (in all cases, it is the researchers themselves who performs any use cases or "field experimentation" of their approach). This result is therefore based on our own assumptions of what we would expect of security knowledge from a developer. Moreover, some security experts probably know how to program or install software. An explanation of roles (shown in 4.7) was developed prior to the extraction of data to limit bias and inconsistencies among our own perceptions.

In a number of approaches, a developer's skills would be sufficient to use the approach/perform the testing. However, usually the approaches require someone with more knowledge about security to fully understand the output or report, which we further discuss in Section 5.1.5.

5.1.5 RQ1.5 Reporting

In order to answer *RQ1.5 How are the results of the approach reported?*, we use the results from the type of issues reported (Table 4.8), the type of report format

(Figure 4.3), and its target audience (Table 4.9).

The type of issues reported is useful for practitioners wanting to adopt an approach for instance to identify bugs, vulnerabilities or risks in particular.

By knowing the target audience of each approach, practitioners may better understand who would be the beneficiaries of an approach. We acknowledge that there could exist a broader audience for these approaches, such as managerial roles, or regulators concerned with auditing. By including the *Compliance* column, we intend to bring better visibility to this audience.

The report format of the approaches, is intended to guide practitioners in highlighting the feasibility of integrating the approach in existing systems. Approaches that report its results in a structured format, may be used directly as input to other tools. While unstructured formats often generates finalized outputs, such as pdf reports, or websites presenting the results.

Fuzzers return as output the test cases that caused the system to crash. This output may be used by developers to ease creation of unit tests that will reproduce this behavior. Another example, in P05, the output of symbolic execution is a set of input parameters for each unique path traversed in code. Even if the symbolic execution did not result in any new vulnerabilities found, the input parameters generated are valuable to test engineers in order to help guide further test case generation. However, since P05 did not document the format of this output, we were unable to categorize it as either structured or unstructured.

Many of the examples of the final reports presented in P18, are easily understandable and written with clear words. Due to this, we believe that many developers could make sense of it, and possibly take action, without having detailed security knowledge. However, for several issues, it would still require a security expert to do an analysis of the results, and recommend actions to fix them.

Developers generally only have the ability to report an error discovered during testing as an error, while a security expert can report it as a vulnerability with a certain risk. Tools that take the role of an security expert for you, by e.g. mapping the vulnerabilities to CVEs and reporting it in a human readable output, enable developers to make more use out of it, since CVEs are easier to search for and understand than e.g. the stack traces given by many fuzzers. Stack traces given by fuzzers are usually not human readable in their raw form, and therefore require external tools to process them, such as GDB with GDB Exploitable Plugin [52], AFL Crash Triage [53], or Crashwalk [54] for iterating through crashes.

In general it might be hard to motivate developers to act on bugs, risks, or vulnerabilities, if they are not reported in a format where they can understand why it is important to fix them.

Most approaches do not describe their output in enough detail, or at all. This makes it harder for both practitioners and researchers to know what value this approach gives, or how they can make use of the output in other systems, and how to improve their product with it by feeding it back into development teams. Based on this, we suggest that future researchers specify the output clearly.

5.2 RQ2 Automation

In order to answer *RQ2 To what degree can the approaches be automated?*, we look at how automated each approach appears to be, as shown in Figure 4.4, and the results from RQ1.

P14 presents the Mirage framework for security audits, which relies on physical hardware. Due to this it might be difficult to fully automate this approach, unless efforts are made to be able to use virtual instead of physical devices.

P17 presents the idea of using metadata from commonly used diagrams and models (provided they contain enough information) to automate security assurance, which seems promising, but is not yet mature enough to be put into use. It also does not consider the process of inserting the metadata into the diagrams or models, nor how to extract the data.

P18 presents a testbed which supports many automated tests, and also provides much of the output in a structured format, which is promising for enabling further automation. They also provide an example of how to parse the output from one test, in order to use it in further analysis to discover vulnerabilities.

P21 runs in a simulator, which is promising for minimizing the amount of manual work a tester has to do when running (and rerunning) the tests.

Challenges with emulating hardware, as discussed in Section 5.1.1, are also an obstacle to enable automation. In a DevOps CI/CD pipeline, it does not seem practical to have a physical test environment available for performing security assurance. So in this case it would be necessary simulate or emulate the relevant IoT devices in order to perform testing in a virtual environment. In addition, approaches that could be performed during *Application Development* are more likely to fit into a CI pipeline, since it is easier to integrate testing that do not require complex testing environments, but no approaches found were suitable for this.

As most approaches do not mention the output clearly, and only a few approaches report their results in a structured format, it can be hard to use these approaches in an automated setting without making changes to support this.

As IoT networks can consist of hundreds or thousands of devices, dealing with all of them in a manual manner is unfeasible, so achieving a high level of automation is of importance. IoT devices generate large quantities of data, and one challenge is creating test cases cover all the relevant scenarios that can occur, and also it can be hard to know what the correct behavior should be.

5.3 RQ3 Adoption

In order to answer *RQ3 What is the ease of adoption of the approaches found?*, we consider the expertise needed for making use of the approaches, as shown in Figure 4.5, and draw upon the results from RQ1 and RQ2.

We found that while 50% of approaches were possible to be performed by *Developers*, only 24% of the approaches provided output that they are likely to fully understand. In other words, most of the included security testing approaches

would require practitioners to either train their developers in security, or hire security experts in order to make full use of the results. We believe the former to be most cost-effective.

As a good starting point for performing security testing of IoT, fuzzing using EcoFuzz (P07), and analysis of firmware vulnerabilities using PATCHECKO (P10), can be used with little preparations needed.

P17 approach seems like a promising enabler for automated security assurance if adopted early in the design process. It is still an abstract approach that cannot be easily implemented for mature IoT companies. In addition, we believe much work is required in order to derive the recommendations from this paper into an actual IoT model.

In P18, the role of security expert is practically digitized, using ML to assess possible vulnerabilities and provide information such as existing CVE reports and severity indicators to guide development.

We found that 60% of the tools in the approaches were available open-source (with varying levels of documentation), which is a good sign for practitioners interested in trying out the approaches. The Mirage security testing framework presented in P14, has the best guidelines for developers for how to use the tool, and how to extend it for their own use cases.

The testbed presented in P18 seems like a powerful approach for doing security testing. However, the software for orchestrating the security testing and the advanced security tests, is not available. Thus, it is not an approach that is easy to adopt for practitioners, as they would have to attempt to make their own from scratch (which seems like a considerable task). It is also presented as an approach that works best in a shielded room, which might make it infeasible for some companies. However, the authors mention that they plan to provide the testbed as a service where individuals or enterprises can use it on their own IoT environment, to obtain a metric score of the level of security.

5.3.1 RQ3.1 Ease of adoption to agile teams

In the search to answer *RQ3.1 How easily adoptable are the approaches to agile teams?*, we would expect to find some of the approaches mentioning their applicability or relevance to developers of IoT software. We found one paper mentioning DevOps, while none of the other papers mentions agile or any particular development frameworks. The scarce results could be related to our findings for RQ1.2, which indicate that none of the approaches were applicable during development. As we reported in Section 4.3.1, approaches relevant for agile teams were indeed found. P06 and P03 could prove relevant for bringing transparency to development teams by monitoring systems at code level (P06) and system level (P03) respectively.

Additionally, the use of containerized technology (Docker), as reported by P05, makes managing and installing security testing tools easier. The tool used for symbolic execution in P05, KLEE, is available as a docker image [55]. As 60% of the

approaches used in this study are available open-source, we strongly encourage researchers to (when possible) provide containerized versions of their tools. This, we believe, would greatly ease the adoption of tools for any development team regardless of development methodology used.

5.3.2 RQ3.2 Prioritization of test cases

In order to answer *RQ3.2 To what extent does the approaches support prioritization of test cases?*, we looked at what the approaches mention that could be relevant to this.

We found only a few approaches that seem a bit relevant for answering this research question, and so, our discussion here contains mainly vague ideas.

One idea could be to generate many test cases, and use some metrics or vulnerability catalog like CVE, to prioritize the order of executing test cases. Or for example deciding which test cases to run at certain times for regression testing (which may take longer to run and could be unfeasible to run continuously), based on some metric, e.g time since last run or time since last code-change in a relevant component. In this way, one could run tests that might have a higher likelihood of uncovering faults. Another way to prioritize tests could be based on abnormalities detected in a system, which could be uncovered using a monitoring approach like P06.

We suggest more research is done in prioritizing test cases for IoT systems, as we think it could be a promising idea for helping practitioners achieving "good enough" security for their systems, allowing for the tests to be run more frequently, which might result in issues being detected earlier.

5.3.3 RQ3.3 Scoping of test areas

In order to answer *RQ3.3 To what degree does the approaches support scoping of test areas?*, we looked at what the approaches mention that could be relevant to scoping the test areas in some way. We found no interesting results related to this research question.

5.4 Implications for Research

In this section, we summarize the implications for research found in this study.

In General

- A need for "left-shifting" security testing of IoT. Approaches that are applicable during planning and design may uncover security-related flaws or defects that will be costly to fix later in the development phases.
- The papers used in this study did not underpin their evaluation or research method using existing theory. We call for a theoretical framework for evaluating security testing methods for IoT.

- Few papers address how their security testing approach reports its results.

Possible Directions For Future Research

- Leverage ML both in existing security tools, and in novel security testing approaches.
- Perform security testing when integration more than one or two IoT devices in a larger network.
- Prioritizing test cases for IoT systems.
- Integrate SAST tools into security testing for IoT.
- Integrate security testing into activities such as CI/ CD for use during application development.
- Both improving the efforts of mining collected data from audit hooks, and applying audit hooks in a large scale to evaluate its performance.

5.5 Implications for Practice

In this section, we summarize the implications for practice found in this study.

- Output from fuzz testing is valuable as input to other test methods, such as unit testing or further test case generation efforts. In addition, fuzzing tools may be collectively stronger when combined. The input which generated a crash reported by one fuzzer, may be used as input to other fuzzers or for symbolic execution.
- Mirage [36], presented in P14, seems promising for auditing and for performing wireless security analysis.
- The use of audit hooks presented in P06 is a novel approach for monitoring IoT systems. This can be used in production environments to detect abnormal behavior, and alert potential misuse of methods.

5.6 Threats to Validity

By only searching for papers in only two databases, where several more could be applicable, relevant papers could have been omitted. In addition, since this research area is still relatively young, literature in the field is lacking, and there could be approaches used in the industry that have been documented in gray literature that this SLR excluded. This study could be extended to capture more types of literature.

We acknowledge that during selection of papers related to "security testing", the selection could be biased by research interest, or by interpreting the inclusion criteria falsely. To limit the possibility of excluding possibly relevant papers, two supervisors were included in assessment of the process.

In interpreting the results, the researchers assumptions and interpretation of the approaches described might have affected the results, which is particularly

likely in the cases where the papers do not explicitly mention what the researchers are looking to find, but it is possible to judge what is likely the case. The results were cross-checked and discussed extensively by both authors to limit the bias. We made efforts to make the results traceable by providing references to the papers where we mapped them to different categories, and when making claims.

The majority of papers reports of a "successful" approach, while also performing the evaluation of their own approach, or not evaluating their proposed method at all, which indicates publication bias. We believe that excluding papers with a lack of proper evaluation alone, would not be possible given the current state of research found in this field.

Chapter 6

Conclusion

Ranging from devices used in our everyday life, to devices used in agriculture and manufacturing, Internet of Things (IoT) devices augments our perception of the physical world with the use of sensory data. The objective of this systematic literature review was to synthesize current knowledge on testing the security of IoT systems, to determine what approaches are most prominent in literature. 1335 articles were identified on testing of IoT, of which we included 21 papers that focused on security testing. Fuzz testing was found to be the most prominent security testing method. The majority of approaches were only applicable once the system was in a production-ready state, a finding that supports the need for a "shift-left" in security. Our contributions include an overview of current state-of-the-art approaches for performing security testing of IoT, including degree of automation and their ease of adoption. Moreover, we propose areas for future research, and guidelines for practitioners who want to improve the security of their IoT systems.

6.0.1 Future Work

We will use the findings of this SLR to guide future work in creating a prototype of a tool to support security testing of IoT systems. In particular, we want to shift-left security testing, and also explore possibilities for adapting any of the approaches recommended in this study for use in an agile setting.

Bibliography

- [1] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu and A. Pretschner, 'Security testing: A survey,' in *Advances in Computers*, vol. 101, Elsevier, 2016, pp. 1–51.
- [2] Henrik Strand. (2020). 'IoT based monitoring for power grid components,' [Online]. Available: <https://blog.sintef.com/sintefenergy/iot-based-monitoring-for-power-grid-components/> (visited on 15/12/2021).
- [3] TeBe Sport AS. (). 'Utnyttelsegrad og aktivitetsmåling for idrettsanlegg og aktivitetsflater,' [Online]. Available: <https://www.tebe-sport.no/aktivitetsmaling-og-utnyttelsesgrad/> (visited on 14/12/2021).
- [4] V. Edmondson, M. Cerny, M. Lim, B. Gledson, S. Lockley and J. Woodward, 'A smart sewer asset information model to enable an 'internet of things' for operational wastewater management,' *Automation in Construction*, vol. 91, pp. 193–205, 2018.
- [5] Tara Seals. (2021). 'IoT Attacks Skyrocket, Doubling in 6 Months,' [Online]. Available: <https://threatpost.com/iot-attacks-doubling/169224/> (visited on 06/09/2021).
- [6] Shear, Michael D. and Perlroth, Nicole and Krauss, Clifford. (2021). 'Colonial Pipeline Paid Roughly \$5 Million in Ransom to Hackers,' [Online]. Available: <https://www.nytimes.com/2021/05/13/us/politics/biden-colonial-pipeline-ransomware.html> (visited on 15/12/2021).
- [7] The White House. (2021). 'Executive Order on Improving the Nation's Cybersecurity,' [Online]. Available: <https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/> (visited on 15/12/2021).
- [8] G. Murad, A. Badarneh, A. Qusef and F. Almasalha, 'Software testing techniques in iot,' in *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, IEEE, 2018, pp. 17–21.
- [9] G. McGraw, 'Software security,' *IEEE Security Privacy*, vol. 2, no. 2, pp. 80–83, 2004. DOI: 10.1109/MSECP.2004.1281254.
- [10] R. Bachmann and A. D. Brucker, 'Developing secure software: A holistic approach to security testing,' *Datenschutz und Datensicherheit (DuD)*, vol. 38, pp. 257–261, 2014.

- [11] E. Suescun Monsalve, C. Calvache, S. Muñoz and A. Uribe, 'Devops in industry 4.0: A systematic mapping,' *Revista Facultad de Ingeniería*, vol. 30, pp. 1–16, Jul. 2021. DOI: 10.19053/01211129.v30.n57.2021.13314.
- [12] C. Ebert, G. Gallardo, J. Hernantes and N. Serrano, 'Devops,' *Ieee Software*, vol. 33, no. 3, pp. 94–100, 2016.
- [13] H. Myrbakken and R. Colomo-Palacios, 'Devsecops: A multivocal literature review,' in *International Conference on Software Process Improvement and Capability Determination*, Springer, 2017, pp. 17–29.
- [14] Atlassian. (2021). 'What is DevOps?' [Online]. Available: <https://www.atlassian.com/devops> (visited on 06/12/2021).
- [15] R. Ankele, S. Marksteiner, K. Nahrgang and H. Vallant, 'Requirements and recommendations for iot/iIoT models to automate security assurance through threat modelling, security analysis and penetration testing,' in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ser. ARES '19, Canterbury, CA, United Kingdom: Association for Computing Machinery, 2019, ISBN: 9781450371643. [Online]. Available: <https://doi.org/10.1145/3339252.3341482>.
- [16] M. Dawson, D. Burrell, E. Rahim and S. Brewster, 'Integrating software assurance into the software development life cycle (sdlc),' *Journal of Information Systems Technology and Planning*, vol. 3, pp. 49–53, Jan. 2010.
- [17] T. Yue, P. Wang, Y. Tang, E. Wang, B. Yu, K. Lu and X. Zhou, 'EcoFuzz: Adaptive Energy-Saving greybox fuzzing as a variant of the adversarial Multi-Armed bandit,' in *29th USENIX Security Symposium (USENIX Security 20)*, USENIX Association, Aug. 2020, pp. 2307–2324, ISBN: 978-1-939133-17-5. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity20/presentation/yue>.
- [18] M. Cortés, R. Saraiva, M. Souza, P. Mello and P. Soares, 'Adoption of software testing in internet of things: A systematic literature mapping,' in *Proceedings of the IV Brazilian Symposium on Systematic and Automated Software Testing*, 2019, pp. 3–11.
- [19] E. L. Macedo, E. A. de Oliveira, F. H. Silva, R. R. Mello, F. M. França, F. C. Delicato, J. F. de Rezende and L. F. de Moraes, 'On the security aspects of internet of things: A systematic literature review,' *Journal of Communications and Networks*, vol. 21, no. 5, pp. 444–457, 2019.
- [20] Y. Lu and L. Da Xu, 'Internet of things (IoT) cybersecurity research: A review of current research topics,' *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2103–2115, 2018.
- [21] B. Kitchenham, 'Procedures for performing systematic reviews,' *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

- [22] K. Tuma, G. Calikli and R. Scandariato, 'Threat analysis of software systems: A systematic literature review,' *Journal of Systems and Software*, vol. 144, pp. 275–294, 2018.
- [23] Zotero. (2021). 'Zotero,' [Online]. Available: <https://www.zotero.org/> (visited on 09/12/2021).
- [24] Y. Wang, E. Kjerstad and B. Belisario, 'A dynamic analysis security testing infrastructure for internet of things,' in *2020 Sixth International Conference on Mobile And Secure Services (MobiSecServ)*, IEEE, 2020, pp. 1–6.
- [25] Y. Mahmoodi, S. Reiter, A. Viehl, O. Bringmann and W. Rosenstiel, 'Attack surface modeling and assessment for penetration testing of iot system designs,' in *2018 21st Euromicro Conference on Digital System Design (DSD)*, IEEE, 2018, pp. 177–181.
- [26] S. Karagiannis, M. Manso, E. Magkos, L. L. Ribeiro and L. Campos, 'Automated and on-demand cybersecurity certification,' in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, IEEE, 2021, pp. 174–179.
- [27] M. Pucher, C. Kudera and G. Merzdovnik, 'Avrs: Emulating avr microcontrollers for reverse engineering and security testing,' in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020, pp. 1–10.
- [28] J. Vijiuk, L. Perkov and A. Krog, 'Bug detection in embedded environments by fuzzing and symbolic execution,' in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, IEEE, 2020, pp. 1218–1223.
- [29] I. Shrestha and M. Hale, 'Detecting dynamic security threats in multi-component iot systems,' in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.
- [30] J. Men, G. Xu, Z. Han, Z. Sun, X. Zhou, W. Lian and X. Cheng, 'Finding sands in the eyes: Vulnerabilities discovery in iot with eufuzzer on human machine interface,' *IEEE Access*, vol. 7, pp. 103 751–103 759, 2019.
- [31] Y. Zheng, A. Davanian, H. Yin, C. Song, H. Zhu and L. Sun, 'Firm-afl: High-throughput greybox fuzzing of iot firmware via augmented process emulation,' in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1099–1114.
- [32] P. Sun, L. Garcia, G. Salles-Loustau and S. Zonouz, 'Hybrid firmware analysis for known mobile and iot security vulnerabilities,' in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2020, pp. 373–384.

- [33] S. Marksteiner, R. Ramler and H. Sochor, 'Integrating threat modeling and automated test case generation into industrialized software security testing,' in *Proceedings of the Third Central European Cybersecurity Conference*, 2019, pp. 1–3.
- [34] M. Bettayeb, O. A. Waraga, M. A. Talib, Q. Nasir and O. Einea, 'Iot testbed security: Smart socket and smart thermostat,' in *2019 IEEE Conference on Application, Information and Network Security (AINS)*, IEEE, 2019, pp. 18–23.
- [35] A. Liu, A. Alqazzaz, H. Ming and B. Dharmalingam, 'Iotverif: Automatic verification of ssl/tls certificate for iot applications,' *IEEE Access*, 2019.
- [36] R. Cayre, V. Nicomette, G. Auriol, E. Alata, M. Kaaniche and G. Marconato, 'Mirage: Towards a metasploit-like framework for iot,' in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019, pp. 261–270. DOI: 10.1109/ISSRE.2019.00034.
- [37] B. Feng, A. Mera and L. Lu, 'P2im: Scalable and hardware-independent firmware testing via automatic peripheral interface modeling,' in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1237–1254.
- [38] T. Zitta, M. Neruda, L. Vojtech, M. Matejkova, M. Jehlicka, L. Hach and J. Moravec, 'Penetration testing of intrusion detection and prevention system in low-performance embedded iot device,' in *2018 18th International Conference on Mechatronics-Mechatronika (ME)*, IEEE, 2018, pp. 1–5.
- [39] S. Siboni, V. Sachidananda, Y. Meidan, M. Bohadana, Y. Mathov, S. Bhairav, A. Shabtai and Y. Elovici, 'Security testbed for internet-of-things devices,' *IEEE transactions on reliability*, vol. 68, no. 1, pp. 23–44, 2019.
- [40] C. M. Coman, G. D'amico, A. V. Coman and A. Florescu, 'Techniques to improve reliability in an iot architecture framework for intelligent products,' *IEEE Access*, vol. 9, pp. 56 940–56 954, 2021.
- [41] T. W. Tseng, C. T. Wu and F. Lai, 'Threat analysis for wearable health devices and environment monitoring internet of things integration system,' *IEEE Access*, vol. 7, pp. 144 983–144 994, 2019.
- [42] M. Ren, X. Ren, H. Feng, J. Ming and Y. Lei, 'Z-fuzzer: Device-agnostic fuzzing of zigbee protocol implementation,' in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021, pp. 347–358.
- [43] Michal Zalewski. (2021). 'American Fuzzy Lop,' [Online]. Available: <https://lcamtuf.coredump.cx/afl/> (visited on 12/12/2021).
- [44] Pentest-Tools. (2021). 'Pentest-Tools,' [Online]. Available: <https://pentest-tools.com/> (visited on 12/12/2021).
- [45] Google. (2021). 'Google ML Kit,' [Online]. Available: <https://developers.google.com/ml-kit/> (visited on 12/12/2021).

- [46] TensorFlow. (2021). 'TensorFlow,' [Online]. Available: <https://www.tensorflow.org/> (visited on 13/12/2021).
- [47] SonarQube. (2021). 'SonarQube,' [Online]. Available: <https://www.sonarqube.org/> (visited on 23/11/2021).
- [48] I. de Oliveira Nunes, K. Eldefrawy, N. Rattanavipanon and G. Tsudik, 'Pure: Using verified remote attestation to obtain proofs of update, reset and erasure in low-end embedded systems,' in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8. DOI: 10.1109/ICCAD45719.2019.8942118.
- [49] I. D. O. Nunes, K. Eldefrawy, N. Rattanavipanon, M. Steiner and G. Tsudik, 'Vrased: A verified hardware/software co-design for remote attestation,' in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1429–1446.
- [50] KaliLinux. (2021). 'KaliLinux,' [Online]. Available: <https://www.kali.org/> (visited on 12/12/2021).
- [51] PortSwigger. (2021). 'Burp Suite Professional,' [Online]. Available: <https://portswigger.net/burp/pro> (visited on 12/12/2021).
- [52] Jonathan Foote. (2021). 'GDB Exploitable plugin,' [Online]. Available: <https://github.com/jfoote/exploitable> (visited on 14/12/2021).
- [53] Google AFL. (2021). 'Crash Triage,' [Online]. Available: <https://github.com/google/AFL#10-crash-triage> (visited on 14/12/2021).
- [54] bnagy. (2021). 'Crashwalk,' [Online]. Available: <https://github.com/bnagy/crashwalk> (visited on 14/12/2021).
- [55] KLEE. (2021). 'KLEE - Symbolic Virtual Machine,' [Online]. Available: <https://hub.docker.com/r/klee/klee> (visited on 14/12/2021).

