# A modern approach for Threat Modelling in agile environments: redesigning the process in a SaaS company

Master's Degree Programme in Information and Communication Technology
Department of Computing, Faculty of Technology
Master of Science in Technology Thesis
Cyber Security

Author:
Emanuele Beozzo

Supervisors:
Antti Hakkala

University of Turku - September 2023

UNIVERSITY OF TURKU
Department of Computing, Faculty of Technology

Emanuele Beozzo: A modern approach for Threat Modelling in agile environments: redesigning the process in a SaaS company

Master of Science in Technology Thesis, 87 p., 15 app. p.
September 2023

---

Dealing with security aspects has become one of the priorities for companies operating in every sector. In the software industry building security requires being proactive and preventive by incorporating requirements right from the ideation and design of the product. Threat modelling has been consistently proven as one of the most effective and rewarding security activities in doing that, being able to uncover threats and vulnerabilities before they are even introduced into the codebase. Numerous approaches to conduct such exercise have been proposed over time, however, most of them can not be adopted in intricate corporate environments with multiple development teams.
This is clear by analysing the case of Company Z, which introduced a well-documented process in 2019 but scalability, governance and knowledge issues blocked a widespread adoption. The main goal of the Thesis was to overcome these problems by designing a novel threat modelling approach, able to fit the company's Agile environment and capable of closing the current gaps.
As a result, a complete description of the redefined workflow and a structured set of suggestions was proposed. The solution is flexible enough to be adopted in multiple different contexts while meeting the requirements of Company Z. Achieving this result was possible only by analysing the industry's best practices and solutions, understanding the current process, identifying the pain points, and gathering feedback from stakeholders. The solution proposed includes, alongside the new threat modelling process, a comprehensive method for evaluating and verifying the effectiveness of the proposed solution.

Keywords: threats, threat modelling process, evaluation process, agile, automation

# Contents

**1 Introduction** **1**

1.1 Research Objectives and Industrial Context . . . . . 2

1.2 Research Contributions . . . . . 4

1.3 Thesis Outlines . . . . . 5

**2 Literature Review** **7**

2.1 Introduction to Threat Modeling . . . . . 7

2.1.1 History . . . . . 8

2.1.2 Why do Threat Modelling . . . . . 9

2.1.3 Misconceptions about Threat Modelling . . . . . 11

2.1.4 Steps of Threat Modelling . . . . . 12

2.1.5 People involved . . . . . 16

2.2 Threat Modeling Frameworks . . . . . 18

2.2.1 Why are there so many frameworks and methodologies? . . . . 19

2.2.2 List of well-known frameworks . . . . . 19

2.3 Problems with Threat Modeling Frameworks . . . . . 24

2.3.1 Changing of the development paradigm . . . . . 24

2.3.2 Security that Understands Development . . . . . 25

2.4 New ways to Threat Model . . . . . 26

2.4.1 Threat Modelling Tools . . . . . 26

2.4.2 New Threat Modelling Approaches . . . . . . . . . . . . . . . 31

2.4.3 Suggestions for implementing Threat Modelling . . . . . . . . 36

2.4.4 Use case in a real corporate environment . . . . . . . . . . . . 38

**3 Current SDLC at Company Z 40**

3.1 Software Development Life Cycle (SDLC) . . . . . . . . . . . . . . . . 41

3.1.1 Pre-Development . . . . . . . . . . . . . . . . . . . . . . . . . 42

3.1.2 Development . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

3.1.3 Post-Development . . . . . . . . . . . . . . . . . . . . . . . . 43

3.2 Secure SDLC . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44

3.2.1 Pre-Development Security . . . . . . . . . . . . . . . . . . . . 44

3.2.2 Development Security . . . . . . . . . . . . . . . . . . . . . . 46

3.2.3 Post-Development Security . . . . . . . . . . . . . . . . . . . 46

3.3 Considerations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

**4 Process requirements 50**

4.1 Data collection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50

4.1.1 Participants . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51

4.1.2 Style and questions . . . . . . . . . . . . . . . . . . . . . . . . 52

4.2 Data analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

4.2.1 Threat Modelling process . . . . . . . . . . . . . . . . . . . . 53

4.2.2 Threat Modelling session . . . . . . . . . . . . . . . . . . . . . 54

4.2.3 Suggestions . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55

4.3 Requirements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56

**5 Design and verification methods 58**

5.1 Design of the process . . . . . . . . . . . . . . . . . . . . . . . . . . . 58

5.1.1 Governance . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58

5.1.2 Process management . . . . . . . . . . . . . . . . . . . . . . . 62

5.1.3 Time and Automation . . . . . . . . . . . . . . . . . . . . . . . . 63
5.1.4 Knowledge . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
5.1.5 Final process . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
5.2 Review of verification methods . . . . . . . . . . . . . . . . . . . . . 69
5.3 Design of the evaluation process . . . . . . . . . . . . . . . . . . . . 70
5.3.1 Qualitative data . . . . . . . . . . . . . . . . . . . . . . . . . . 71
5.3.2 Quantitative metrics . . . . . . . . . . . . . . . . . . . . . . . . 72
5.4 Test implementation and Experiments . . . . . . . . . . . . . . . . . 75

**6 Conclusion** **85**
6.1 Future work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86

**References** **88**

**Appendices**

**A Templates for the semi-structured interviews** **A-1**
A.1 Template for Software Engineers . . . . . . . . . . . . . . . . . . . . A-1
A.2 Template for internal Ethical Hacker . . . . . . . . . . . . . . . . . . A-2
A.3 Template for Product Owner . . . . . . . . . . . . . . . . . . . . . . . A-4

**B OWASP Threat Dragon Report** **B-1**

**C pytm Report** **C-1**

# List of Figures

3.1 SDLC workflow . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
3.2 Secure SDLC workflow - part 1 . . . . . . . . . . . . . . . . . . . . . . 47
3.3 Secure SDLC workflow - part 2 . . . . . . . . . . . . . . . . . . . . . . 48

5.1 Updated threat modelling process and Secure SDLC workflow . . . . 68
5.2 DFD automatically produced by pytm . . . . . . . . . . . . . . . . . . 81
5.3 DFD manually created with by OWASP Threat Dragon . . . . . . . . 81
5.4 Sequence diagram automatically generated by pytm . . . . . . . . . . 82

# List of Tables

2.1 Graphical tools comparison . . . . . . . . . . . . . . . . . . . . . . . . 27

2.2 Threat Modelling with code . . . . . . . . . . . . . . . . . . . . . . . . 30

2.3 Suggestions for implementation . . . . . . . . . . . . . . . . . . . . . . 37

4.1 Job title of the employees interviewed . . . . . . . . . . . . . . . . . . 51

4.2 Thematic analysis: Threat Modelling process findings . . . . . . . . 54

4.3 Thematic analysis: Threat Modelling sessions findings . . . . . . . . 55

4.4 Thematic analysis: Mapping between problems and suggestions . . . 56

5.1 Thematic analysis: Mapping between problems and suggestions . . . 59

5.2 Metrics for quantitative evaluation . . . . . . . . . . . . . . . . . . . . 73

5.3 Metrics for quantitative evaluation . . . . . . . . . . . . . . . . . . . . 74

5.4 Mapping between the threat detected . . . . . . . . . . . . . . . . . . 83

# 1 Introduction

Despite the increased attention in media and corporate towards cyber security and projected double-digit growth in the sector's expenditure in the coming years [1], it remains evident that an alarming trend persists: the annual count of compromised data continues to rise, reaching a frequency where an attack occurs every 39 seconds [2], [3]. For this reason, it is vital to be proactive when dealing with security, instead of only focusing on detection, response and recovery. Among all the prevention possibilities, threat modelling is one of the top security practices when talking about application security, for both defenders and builders, as stated in the 2015 SANS survey on the State of Application Security [4]. It has also been recognized as one of the most effective ROI (Return on Investment) activities for discovering and adjusting design issues prior to code implementation [5].

According to the manifesto published by Adam Shostack and other fourteen security experts, threat modelling simply analyses the representations of an IT system to highlight concerns about security and privacy characteristics [6], aiming at identifying the attacks that an application has to withstand and the practical defences that must be built, even before starting to write a single line of code. Threat modelling is part of the "Shift-Left Security" concept which suggests moving security checks as early and frequently as possible in the Software Development Life Cycle (SDLC), as vulnerabilities discovered earlier are much easier and less expensive to correct. [7].

Implementing threat modelling in a corporate or high-speed scale-up environment is not trivial and requires a systematic approach to analyze the system, document the security requirements, and create a "defensive thinking mindset" in the early stages of the development process. Thinking defensively entails considering how a new requirement or feature could be abused or defeated by adversaries [8]. Over time, different frameworks and tools have been created to make it easier to threat model, but it takes time to constantly teach staff how to use them, integrate them into the SDLC and enforce their usage.

Furthermore, the establishment of Agile Software Development frameworks, which led to the decline of older methods of software development and improved the velocity and frequency of delivery cycles and feedback, contrasts with the traditional method of performing threat modelling, which can be perceived as slow and time-consuming. Following the Agile mantra YAGNI ("You Ain't Gonna Need It"), development teams do not see the value in prioritising or even considering threat modelling, as the design phase is usually quick and not as intense as in other methodologies [5], [9].

## 1.1 Research Objectives and Industrial Context

The main goal of the Master's Thesis is to overcome some of the limitations, presented before, in the adoption of a highly-valuable process for application security, namely Threat Modeling, inside software companies. The idea behind this project originated from a business challenge inside "Company Z", as we will call it throughout the thesis, and the solution proposed was developed in that industrial context considering real-world requirements and implications that are usually not underestimated in academic environments.

Company Z is a leading Digital Asset Management (DAM) company that offers a cloud-based solution to empower businesses to manage, collaborate, and distribute their brand assets effectively. Founded in 2013, it helped revolutionise the way organizations handle their creative content and streamline their marketing processes. More than 550 international employees work for Company Z, in 5 different offices located around the globe, and the size keeps scaling up. Company Z is home to numerous departments (such as customer onboarding and success, marketing, finance, legal, IT, Information Security, R&D, and many more) that collaborate in synergy to offer a trusted solution for businesses looking to streamline their creative processes and elevate their brand experiences. At the moment, R&D at Company Z counts more than 20 agile development squads, each of which is in charge of maintaining and developing a certain component of the product and each of them has different needs.

In such a business context, it is fundamental to consider security requirements to avoid injecting flaws in the service provided to customers to build a trust relationship and avoid data leaks and reputation damage. So, Threat Modeling is an essential step to help improve the security posture as early as the design phase. The Information Security team of Company Z, in collaboration with some experienced software engineers, introduced a well-documented Threat Modeling phase inside the SDLC in 2019, however, the solution presented some of the aforementioned problems like scalability, integration with an agile development framework and some others affected the adoption rate.

The key research questions that the presented Master Thesis wants to answer are the following:

1. Are there effective and modern methods to perform threat modelling inside high-speed development environments, like the Agile ones? Did the methodologies to elicit threats evolve as the software development did?

2. What is the most useful information that can be collected from the stakeholders about the current threat modelling process? Can these data support the

identification of pain points, the elicitation of new process requirements, and the collection of valuable suggestions?

3. How can a newly proposed threat modelling workflow be seamlessly integrated into an existing software lifecycle? What other factors should be taken into account during process implementation, beyond the workflow?
4. What are the commonly employed methodologies for evaluating the effectiveness of the Threat Modeling processes? Do these methodologies fit in the context of this study and the solution proposed?

## 1.2 Research Contributions

Beyond the effects that the Master's thesis may have on the development teams at Company Z and its overall security posture, other businesses may find the newly proposed threat modelling process useful due to the continuously rising value of this activity and the widespread of Agile methodologies. In fact, the process was designed to be flexible, scalable and adaptable to the needs of different companies by suggesting tools that are open-source and can be integrated into different environments. Moreover, considering the way the solution is structured in different thematic areas, other businesses can decide to tackle only some of the suggestions proposed and not the process as a whole. This design choice is beneficial both in terms of generalization, as in some environments not all the elements need to be changed as some of them can be already well-established, and implementation, due to economic and time constraints.

Additionally, the conducted literature review and comparison of the various threat modelling frameworks, tools, and implementation strategies can contribute to the academic field and be utilized to swiftly catch up on the state of the art in the field.

Although the thesis proposes innovative approaches created from scratch for both threat modelling and evaluation of the process itself, neither of them was implemented in a production environment and tested with a decent amount of data to establish clear conclusions. The future work will include an implementation of the process starting with a pilot test (both inside Company Z or in another firm) and will proceed with a constantly increasing number of development teams. Throughout the implementation, a fine-tuning of the process based on the feedback and data collected will be required. Once the implementation is completed, and ideally extended to other businesses, it will be possible to create a complete comparison that includes the findings that have been gathered.

## 1.3 Thesis Outlines

The rest of the Master Thesis will be organized as follow.

Chapter 2 presents a more comprehensive overview of the concept of threat modelling, starting with a brief history, and continuing with presenting the various frameworks used in the past and in the present and the tools that are available to speed up the process. It will advance by presenting how threat modelling is adapting to the new agile development methods and the solutions that security experts are proposing.

The elaborate proceed with Chapter 3, which covers how threat modelling is currently conducted inside Company Z and its integration inside the SDLC, after analysing the internal documentation and talking with multiple stakeholders.

Chapter 4 illustrates the methodologies used for the data collection and presents the results of stakeholder interviews, focusing on the problems, limitations and suggestions. After analysing and categorizing the information collected, the requirements and specifications of the new solution are presented.

In Chapter 5 the design of the new threat modelling process is defined, and a test implementation, alongside a comparison with the previous solution, is presented. A brief literature review of the evaluation methods for threat modelling is also described and some suggestions and metrics on how to actually evaluate the new methodology proposed.

Lastly, Chapter 5 briefly summarizes the findings and defines the relevance of the contribution to the Security Industry. Potential and future expansions of the presented work are also discussed.

# 2 Literature Review

Before diving into the various frameworks and methodologies to perform threat modelling, it is necessary to clarify and properly present more concepts about threat modelling that can be useful throughout the reading.

## 2.1 Introduction to Threat Modeling

In the literature, a universally accepted and recognised definition of **Threat Modelling** does not exist. For sure, the one specified inside the Threat Modeling Manifesto [6] is a good starting point, but at the same time is too broad and vague. Combining different sources, it is possible to find that the goal is to "Identify the likely threats to a system to inform the design of security countermeasures" [10], by performing "a collaborative security exercise where we evaluate and validate the design and task planning for a new or existing service" [11]. This exercise entails structured thinking and systematic procedures to undercover potential security weaknesses. It also necessitates examining the system through the eyes of a potential attacker rather than adopting a defender's stance and successfully integrating as a core component of the SDLC [12]. Threat modelling finds applicability across diverse domains, such as software, applications, systems, networks, distributed systems, Internet of Things (IoT) devices, and even business processes.

Threat Modelling is, of course, not the sole security practice available, nor the simplest one to integrate into the SDLC (or even better, the SSDLC - Secure SDLC) and other approaches like penetration testing, fuzzing or Static Analysis Security Testing (SAST)/Dynamic Analysis Security Testing (DAST) can yield more tangible results and wins by leveraging automated tools and outsourcing techniques. However, threat modelling plays a crucial role in achieving resilience: without contemplating what could go wrong, it is challenging to instil confidence that software or service will remain free from unexpected and hard-to-fix issues, due to the involvement of human judgement and decisions that may not be detected by tools [13].

Threat Modelling can be as simple as asking yourself or your team the following four questions proposed by Adam Shostack:

1. What are we working on?
2. What can go wrong?
3. What are we going to do about it?
4. Did we do a good job?

Nowadays, this simple framework serves as a base for most of the modern threat modelling approaches.

### 2.1.1 History

Militaries have been familiar with the concept of threat modelling since antiquity and the first extensive treaty about the topic can be dated around 512 BC. For centuries threat modelling was used only to define military defensive readiness, but in the early 1960s, with the advent of computing a new form of threat - the cyber ones - started to spread around. Initially, the academic research focused on the concept of architectural patterns (Christopher Alexander [14]), attacker profiles (Robert Barnard) and attack and threat trees (Edward Amoroso [15]). The evolution of threat modelling to the current methodologies received a significant boost in 1998 through the introduction of attack trees for cyber-risk analysis, a milestone documented in Bruce Schneier's publication titled "Toward a Secure System Engineering Methodology" [16]. Based on this work, both STRIDE and OCTAVE methodologies were created, respectively by Loren Kohnfelder and Praerit Garg [17] and Carnegie Mellon University [18]. STRIDE is aimed at helping Microsoft security professionals systematically analyze potential attacks targeting components of a computer system. At the same time, OCTAVE was conceived as a risk-centric assessment approach that harmonizes technological and organizational facets of potential threats with established security measures. Remarkably, both of these methodologies continue to hold relevance in the present times and are described in section 2.2. [19]

### 2.1.2 Why do Threat Modelling

In 2022, different forefront organisations in establishing security standards, such as the Food & Drug Administration (FDA), Center for Internet Security (CIS) and National Institute of Standards and Technology (NIST) raised significantly the bar. In particular, NIST ranked threat modelling as first on the "Recommended Minimum Standard for Vendor or Developer Verification of Code" on the list of activities for software verification, beating more widespread methods, including automated testing and external dependencies analysis. This means threat modelling will be part of government procurement processes, aligning with the growing interest in this security practice. This will have a trickle-down effect throughout the software industry and the security measures adopted by companies to remain competitive. [20]

Despite the new compliance requirements, there are tons of other reasons that need to be considered when choosing threat modelling for a project or inside a corporate environment. In the following list, some of the major ones are presented.

- Security requirements elicitation: Requirements represent the foundation of the systems and without them, they can not be developed. Defining the security ones is a huge part of the process and using only generic industry's best practices is not enough to discover all the threats of a particular system. With threat modelling it is possible to overcome this limitation, define what to do and justify security countermeasures and technical choices [21], [22].

- Proactive design guidance: Along the lines of the shift left security concept mentioned before, conducting threat modelling as soon as possible helps engineer a better product right from the start, reducing the necessity to perform security bug fixing at later stages and detecting typical issues that other techniques will not. Being proactive with threat modelling during the design stage is as costly as doing it later, however, the effort and the costs required to mitigate threats in an existing production system due to implementation and architectural constraints can be 100 times higher [21]–[24].
- Risk reduction: Guaranteeing an impenetrable system is impossible, but, working toward 100% risk management is feasible. With threat modelling, insights about threat sources, scenarios and impact on assets are collected so it is clearer to understand the risk appetite and risk tolerance in a particular environment and define the appropriate countermeasures based on the effort required, and budgeting [21], [22], [24].
- Help to think about attacks: Threat modelling supports the creation of the complete attack kill chain by modelling the actors, profile and motivations, simulating possible attack strategies and scenarios and understanding how the defences act in such cases. That information is useful for exercises like tabletop or red teaming, to identify single points of failures and bottlenecks and reduce the attack surface [21], [23], [24].
- System-wide and company-wide security improvement: Threat modelling is a great opportunity to take time to discuss and review the system/feature

under development, evaluate security processes and tools, bring together different teams (security and engineering for example) and share knowledge and awareness [21], [24].

### 2.1.3 Misconceptions about Threat Modelling

Alongside many reasons to perform threat modelling in a systematic and organized way, there is an equal number of misconceptions about it and failures in the implementation of the process that limit the diffusion and create bad fame around it. Again, a non-exhaustive list of these "problems" is presented below, based on the findings presented by Shostack in his whitepaper [25], by Kirtley in its blogs [24] and by Licata in the SAFECode publication [5].

- Heavy, complicated process: Threat modelling is one of the most flexible security activities that can be performed and can be adapted to any possible environment and requirements. The final characteristics of the process deeply rely on how it was designed and, unfortunately, security experts, who are responsible for designing security processes, are often guilty of choosing the most standard and structured way, without appropriate testing and customizations, to overcome the lack of specific expertise of developers and operations teams. This led to the misconception that threat modelling is always tedious and lengthy, but different approaches that can fit environments like agile ones exist and they can be as rewarding as the classical ones, without requiring too much experience or being too complex.

- No useful results: Another problem that is related to the process implementation more than threat modelling per se, is not having a clear goal. Starting a session without a clear scope, context and expected outcome can result in failures on multiple levels, not completely addressing the attack surface and not having concrete results.

- Waste of time: Some can argue that threat modelling takes a lot of time, and that is partially true. But performing one of the most rewarding activities in terms of ROI should never be seen as a waste of time. Choosing the right technique, and setting time boundaries and goals is determinant.
- Occurs only at the design phase: threat modelling is never a one-time activity, it needs to occur throughout the entire lifetime of a system/components. An already built and deployed system can be threat-modelled for different reasons, namely monitoring flaws, adjusting deploying and architectural problems and many others.
- Can replace and be replaced with penetration testing, code review or questionnaire: as said in subsection 2.1.2, threat modelling is the top suggested security practice to perform while coding and can not be replaced with testing and code review. At the same time, it can not replace these activities: it is important to have in place multiple security layers at the same time. Also, security questionnaires can not replace real threat modelling sessions, because a single individual can not be such an expert and know all the answers.

### 2.1.4 Steps of Threat Modelling

Accordingly to AWS, when talking about Threat Modelling and when to perform it, it is possible to say: "The earlier the better. The more often the better. It's never too late." [26]. In other words, it is better to start threat modelling as early as possible in the life cycle of your system and apply it continuously throughout the development as the information gets more and more granular with time. A high-level threat model should be defined during the concept ideation or planning phase when there is still the flexibility to define the most suitable solution to the identified threats, and then constantly refine it.

It is possible to talk about three main types of threat modelling based on when and what they analyse. The classical, which is created from scratch for new or existing systems, is the called "base threat model". The differential is used when building up on the base threat one, due to the creation of a new feature, an architectural change that impacts how data are produced and consumed or the troubleshooting of security concerns. The last class is the blueprint one, employed for recurring patterns, but not used as it is harder to scope and execute [23].

But, despite all the different types of threat models and the enormous number of techniques available to conduct such sessions and integrate them inside the SDLC, it is possible to re-conduct all the activities to four main steps, that answer the questions designed by Shostack [21]. Other sources present may present a slightly different organization [5], [12], [24], but the main ideas behind that remain the same.

### What you're working on

To start with, having a clear idea of what you are working on and your goals is fundamental for creating a successful threat model. In a corporate environment, this includes the selection of the feature the team wants to implement in the next sprint or quarter with their functional requirements well defined.

Once this is done, we need to gain knowledge of how the new component or system interacts with external entities (by defining entry points), the assets involved and the different trust levels they assure. An asset is something valuable within the system and for the business, like information or services, and that needs to be protected. Asset identification is useful not only for threat modelling but also required by many security standards for risk assessments.

When all the information is collected, the easiest way to use it is with models and graphical representations. They can be done at varying levels of formality, but the description must accurately depict suitably. In particular, an application and a net-worked system use different types of diagrams: here the focus is on software systems. For applications, two different categories of visualizations can be constructed. The Data Flow Diagram (DFD) deconstructs the application into functional elements and demonstrates the flow of data throughout the system's processing. It facilitates the elucidation of data entry and exit points for each component, along with the identification of data storage, processing, interactions, and trust boundaries. In contrast, the Process Flow Diagram (PFD) portrays the interactions between users and third parties with the system. They can be used in combination or alone, but only DFDs are widespread in the field.

### What can go wrong?

The next step is central to undercover the threats that can be discovered inside the system. With the help of the information acquired in the previous step and the diagrams created, it is possible to start reasoning about who are possible unintended users of the system analysed and what can happen in case they get access or exploit the company's assets in unintended ways. The goal can be achieved in a variety of different ways, such as STRIDE, that are presented in section 2.2 and are the core of the Thesis.

### What are we going to do?

The next question "What can go wrong" helps figure out the possible dangerous scenario and provide a list of threats and their related asset, actions and prospective attacker profiles. Now it is time to address the threats and analyse the risks connected to each of them. Some major possibilities are the following.

- Mitigating threats: add checks, controls or adjust the design to reduce the impact or the chances of its occurrence;

- Eliminating threats: deactivate the feature or interface or reduce the functionalities that create the threat;
- Transferring threats: pass on the responsibility to manage the threat to other parties if they are better equipped to handle it (examples include customers that have clear responsibilities listed in the licensing agreement);
- Accepting threats: the time and effort required to reduce or eradicate the threat disregard the purpose of the project, or the threat has too little impact or probability;
- Ignore the risk: pretend the threat was not discovered and it is not there. This led to possible compliance violations and it is never recommended.

**Did we do a good job?**

After defining the threats and countermeasures where needed, it is time to take a step back and validate the work done. This includes reviewing the diagram and each of the threats found to determine if the right mitigations and tests were proposed, if all the potential dangers were considered and if the residual risk was estimated correctly. This phase concludes by determining the next activities and possible iterations of the threat modelling sessions on the system, publishing the material about the findings and retrospectively analysing the work done.

**Output of a Threat Modelling process**

Once the various steps needed for threat modelling are clear, it necessary to do the same regarding what is the expected outcome. Apart from the classical information about the system modelled, the assumptions made and that need to be checked in future sessions, the main output consists of:

1. The list of threats found with an index to score their possible impact;

2. The action items that needed to be taken to mitigate the threats;
3. An extensive set of test cases.

These outputs should be used as a base to perform other security activities that are part of the SDLC like risk assessments, source code review, quality assurance and penetration testing [23], [27], [28].

### 2.1.5 People involved

As said, threat modelling is not a one-time task, but it is not also a one-person activity. It is often referred to as a team sport because it requires both knowledge and skills for technical and non-technical individuals who provide different mindsets and values to the session. Conducting threat modelling when multiple individuals are missing, can lower the quality of the outcome and can dismiss the point of view of some stakeholders. A suggested list of the roles that need to be involved is presented below, based on the suggestions of Shostack, Boyd and Licata, but keep in mind that one person can bring multiple perspectives and may cover multiple personas [5], [21], [26], [29].

#### Technical roles

- Software Development Teams and Testers: They are responsible for the actual implementation and testing of the product and the first responsible for building security directly into the code and assuring the effectiveness of the controls defined.
- Systems Architecture Teams: They designed the proposed workflow and can explain it and motivate the decisions made until that moment. They also know the "bigger picture", namely they are aware of how other parts of the system

work and interact with each other. In case of major security issues, they can lead a redesign phase and select the right components.

- Operations Teams: Knowing current threats and vulnerabilities can help during the preparation for deployment, vulnerability patching, security tools selection and deployment, and monitoring metrics definitions.
- Security Teams and AppSec (Application Security) SME (subject matter expert): Security teams are responsible for aligning the business security requirements, evaluating the threats discovered, assessing the risks and defining trade-offs. The AppSec SME is usually a member of such a team, familiar with threat modelling, and its practices, and moderates the session. Penetration testers should be informed about the outcome of the session and test if the mitigations work as intended.

Among all the qualified people involved, some need to play the role of the adversary persona which, as an unauthorized user, tries to take advantage of design flaws to achieve a particular objective, while some others can play the defender persona that tries to mitigate the threats devised by the adversary and evaluates whether the proposed mitigations are manageable in terms of ongoing operational support, monitoring, and incident response.

#### Non-Technical roles

- Project Managers and Project owners: They are in charge of protecting what is planned and under development, the timeline and the expected results and they do have to keep track of what the teams are doing. They hate activities that require a lot of time and unexpected delays that come from penetration tests, bug fixes or security patching, that require more work than planned. Threat modelling can provide results much earlier to them so they can incorporate them in the planning, they can also keep track of the security requirements through the whole development understand if they match the expectations, and avoid last-minute surprises from pen-testing and testing that can postpone a roll-out.

- Legal: In case the product/function is regulated by specific laws or needs, and privacy concerns arise, a legal counsel should be involved.
- Business Managers and Executives: they do not need to get involved in the technical intricacies of Threat Modelling, but they should be limited to making risk-related decisions in conjunction with the recommendations provided by the security teams. The business standpoint regarding threats holds immense significance as it focuses on customer interest and delivered value. Such a perspective can be used to support the prioritization of remediations.

## 2.2 Threat Modeling Frameworks

According to the Cambridge Dictionary, a framework is "a system of rules, ideas, or beliefs that is used to plan or decide something" [30]. Starting to build a solution or process every time from scratch and always trying to reinvent the wheel is time-consuming. For this reason, almost every area of information technology adopts design patterns, coding frames and libraries, and standard architectures. The same applies to threat modelling, where frameworks can be contextualized quite easily as sets of guidelines used to structure the process and improve an organization's ability to identify threats. Employing them helps speed up the implementation of the process and at the same time trusts an already known and well-tested solution. Many threat modelling frameworks were introduced throughout the last decades, as can be seen in the systematic review performed by Xiong et al. [31], but only some stood out and are not deprecated nowadays.

### 2.2.1 Why are there so many frameworks and methodologies?

Even though not all the frameworks survive the test of time, by looking at the literature on threat modelling we can find an infinite number of possibilities, focusing both on security and privacy threats. For Shostack, the reason is the same as such a variety of programming languages or agile methodologies were introduced: threat modelling should be adapted to the environment it is used and a one-size-fits-all can not align on particular needs. Developing a threat modelling process can be seen as an art, and as new information is discovered over time, it should be adjusted and fine-tuned to accommodate the specific needs [21].

### 2.2.2 List of well-known frameworks

The list presented below includes the most well-known frameworks that are still in use or provided a significant contribution to the field. As a baseline for the creation of the list, multiple sources were used to have a broader view of the different methodologies [21], [24], [27], [32], [33], [34]. Lightweight approaches, risk assessment models and scoring systems were intentionally excluded as their main focus is out of the scope of the Master Thesis.

#### STRIDE

STRIDE is a mnemonic approach developed by Microsoft that provides an easy-to-learn and mature way to define what can go wrong. Its acronym stands for a list of the possible threats that can be identified using the methodology, namely Spoofing (which violates Authenticity), Tampering (Integrity), Repudiation (Non-repudiation), Information disclosure (Confidentiality), Denial of Service (Availability) and Elevation of privileges (Authorization). STRIDE emphasizes the development part of the process and participants are required to derive abuse scenarios for each threat. Other specific variants like STRIDE-per-Element and STRIDE-per-

Interaction were created over time, but always with the same goal. Unfortunately, this approach is time-consuming and has a strict reliance on well-done DFDs to produce satisfactory results.

#### PASTA

PASTA, which stands for "Process for Attack Simulation and Threat Analysis", is a risk-based, attacker-centric threat modelling methodology composed of seven steps (Define objectives, Define technical scope, Decompose the application, Analyze threats, Analyze vulnerabilities, Analyze attack paths, Analyze risk and impact). Each of them includes various activities and requires different tools like DFD, attack trees, and use and abuse cases. Due to the incorporation of both business and technical objectives, PASTA is a highly collaborative framework that produces an asset-centric view of the system, identifies and prioritizes threats, and creates rich documentation.

#### Attack Trees

Attack Trees are one of the oldest threat modelling frameworks and they are used to demonstrate how an asset can be attacked, in a way that all the thought process is displayed. In complex systems, it is common to have multiple trees: an attacker can have several goals and each of them requires a different tree. They are represented as graphical diagrams with a hierarchical structure and their logic follows the same idea of every other decision tree. The root of the tree represents the goal of the attack, while the leaves are the ways to achieve that. It is possible to include multiple alternatives in the tree, so AND and OR constructs can be to enforce such requirements. As the creation of attack trees requires advanced security expertise and a deep understanding of the system, they are often used in combination with other techniques to discover if the system is vulnerable or makes security decisions.

### Persona non-Grata

The Persona non-Grata approach, also abbreviated as PnG, is a threat modelling methodology that focuses on human attackers by trying to define their motivations, skills and goals. By using such personas it is easier to define possible misuse cases and highlight possible attack vectors and vulnerabilities of the system from different points of view. While PnG produces consistent results, it is rarely used as tents to detect only a subset of all the threats.

### Security Cards

Security card games are not a usual process-driven methodology to elicit threats, but more brainstorming techniques to discover unusual attacks and ways to bring different stakeholders to reason about security. The main idea is to use a card deck to create threat scenarios and reason about adversary motivations, resources, methods and human impact. Different card games were introduced over time, each of them with its own rules, but the ones worth mentioning are Security Cards by Washington University[1], Elevation of Privilege (EoP) by Microsoft and Adam Shostack[2] and OWASP Cornucopia[3]. Although such games help identify most of the threats, the number of false positives they produce is very high.

### Trike

Trike is a highly structured, compliance-focused and risk management-based threat modelling and security audit framework. It uses a defensive perspective to generate threats and requires the following steps:

1. Definition of the system and the requirements model, by creating an actor-

[1]`https://securitycards.cs.washington.edu/`

[2]`https://www.microsoft.com/en-US/download/details.aspx?id=20303`

[3]`https://owasp.org/www-project-cornucopia/`

asset-action matrix (with information about which CRUD action is allowed/disallowed and the specific rules);

2. Define DFDs and map them to the actors and assets in the matrix;

3. Iterate through the DFDs to start the threat generation phase. Each of the threats discovered can be categorized either as an elevation of privilege or a denial of service and becomes the root node of an attack tree.

4. Using the information from the previous steps, assign a weight to each risk using particular mathematical functions.

The nature of Trike helps prioritize mitigation and achieve an overall risk reduction, but the process itself is extensive and the documentation insufficient.

**VAST Modelling**

The Visual, Agile, and Simple Threat (VAST) model was developed and first used by the company ThreatModeler. The foundations on which this methodology is built (automation, integration, and collaboration) allow it to be scalable and adopted in large organizations with several teams and products. Due to the different views on the system from the development and operation teams, VAST proposes the use of two different threat models: application and operational, respectively. The first one uses PFDs and operates on the architectural view and interaction with the external world, while the second is created from the attacker's perspective based on DFDs. Due to its duality, VAST can be used in an agile environment as different teams can work on different models. Among the disadvantages are the scarcity of available documentation and the need for a vendor-supplied tool to have a real and automated solution.

### OCTAVE

OCTAVE, which stands for Operationally Critical Threat, Asset, and Vulnerability Evaluation, is a risk-based assessment focused more on evaluating organizational aspects than technological ones. It is structured in these main phases: building company-wide security requirements based on assets owned, identifying potential vulnerabilities inside the infrastructure, analyzing potential threats, and defining a risk management strategy. As expected, OCTAVE is a time-consuming methodology, and for this reason, a revised and optimized version, called OCTAVE-ALEGRO, was released.

### LINDDUN

LINDDUN is a threat modelling approach with privacy and data security as the primary focus. Like STRIDE, it is a mnemonic method, meaning the threat categories considered in the evaluation are coded inside the name. In particular, LINDDUN looks for possible problems of Linkability, Identifiability, Non-Repudiation, Detectability, Disclosure of Information, Unawareness, and Non-Compliance (the desired properties are the negation of the ones listed).

### Attacks and Threat libraries

In numerous scenarios, STRIDE could be perceived as excessively theoretical and broad, and incorporating a more extensive range of common issues could improve the effectiveness of recognizing and mitigating threats. Hence, the introduction of attack and threat libraries was prompted by these considerations.

A threat library serves as an organized and searchable location for structured and unstructured security information. It can contain threat intelligence received by feeds and providers or vulnerabilities reported inside catalogues like MITRE CVE and NIST NVD (for publicly disclosed vulnerabilities) or the Snyk Vulnerability

Database (for open-source dependencies). Having such information easily accessible (from a native interface but also using APIs) and being able to automatically aggregate and normalize it can support a threat modelling session and help define the focus with real examples.

Alongside Threat Libraries, other sources of inspiration and guidelines for threat modelling are Attack Libraries. The most famous ones are OWASP Top 10, MITRE CAPEC and ATT&CK, and OSC&R, and their goal is to provide lists of attack patterns, exploits, and techniques used to compromise a system.

## 2.3 Problems with Threat Modeling Frameworks

The frameworks discussed in the preceding section are effective approaches for conducting threat modelling in simple and unstructured setups, suitable for small-scale projects. However, when addressing larger corporations characterized by hundreds of employees, multiple development teams, established procedures, and demanding schedules, the need for tools and solutions capable of speeding up the process and introducing a degree of automation arises. Most of the proposed approaches are only guidelines that can be used as a base to build more comprehensive solutions on top.

### 2.3.1 Changing of the development paradigm

Furthermore, the majority of the recommended frameworks were proposed before the significant shift in the development paradigm that occurred in recent decades, marked by the widespread diffusion of agile practices over traditional models like waterfall development. These changes have had a profound impact on how systems are implemented and deployed, but they particularly influenced the various design activities that precede coding. Threat modelling was notably affected as it was kick-started during that phase, which is short and with a limited emphasis on agile methodologies. Moreover, the fast realise cycle does not allow long and intense brainstorming sessions with multiple stakeholders, and the creation of extensive documentation and diagrams but prioritizes short update meetings and sketches. This led to a rapid disregard for the classical ways of performing threat modelling and the prioritization of other security activities. As discussed before, threat modelling should not be replaced and can be adapted to different SDLCs, and can even be done in sprints. While features are designed, threat models can happen in parallel and mitigations that do have not a high priority can be added to the project backlog [26]. Threat modelling can even fit the DevOps approach, which focuses on automation and everything as code if the right tools and approaches are selected.

### 2.3.2 Security that Understands Development

Developing a process that fits the SDLC and selecting the right tools are not the only challenges that the agile paradigm brings to threat modelling and security in general. According to Mike Milner, Vice President of Cloud Technology at Trend Micro, it is clear that builders and developers have conflicting goals compared to security people, with the results of having security disconnected from development. An example of this behaviour is vulnerabilities or misconfigurations detected in a production system by security teams: such problems should have never been injected into the application if developers focused on security during coding and building time. Instead, if security is integrated and understood development, it is possible to have security feedback throughout the whole SDLC and both the security and developers team can both find and fix issues, based on their expertise [35].

## 2.4 New ways to Threat Model

To overcome the limitations of the frameworks described in the previous section, comprehensive research was performed with the following results. The section is organized into three main parts that present modern tools, different approaches and suggestions articles, suitable for integrating threat modelling in a complex environment such as the one of a company. Also, a real use case on how threat modelling was implemented inside a company environment is presented.

### 2.4.1 Threat Modelling Tools

Threat modelling inside a structured environment with a significant amount of people involved requires a level of automation and cohesion between teams that only well-designed supporting tools can provide. Different types of tools exist and are presented below.

#### Graphical Tools

The first category of tools considered is the graphical one. Their main idea is to start designing a DFD by dragging stencils over a virtual whiteboard (instead of a physical one in a meeting room for example) and create a model of the system. They usually allow the insertion of various architecture components, define the trust boundaries and exclude elements out of the scope of the analysis. When the diagram is complete, the threats can be elicited manually or automatically and inserted inside the tool. The tools considered here are Microsoft Threat Modelling Tool (TMT)[4], OWASP Threat Dragon[5] and Mozilla Sea Sponge[6]. The main difference between TMT and the other tools is their main goal: while the Microsoft tool is something

[4]`https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-getting-started`

[5]`https://github.com/OWASP/threat-dragon`

[6]`https://github.com/mozilla/seasponge`

that "thinks" for the users by automating the process, the other ones are designed to help the user think. Other differences are highlighted in the Table 2.1

| | **Microsoft TM Tool** | **OWASP ThreatDragon** | **Mozilla SeaSponge** |
|---|---|---|---|
| **Automatic Threat Discovery** | Yes | Limited | No |
| **Custom Threat Libraries** | No | No | No |
| **Require DFD** | Yes | Yes | Yes |
| **Approach** | STRIDE per Element | STRIDE | STRIDE |
| **Reporting** | HTML, CSV | PDF | Unknown |
| **Output** | Threats list for interactions (with priority and state) | Threats for each element (with severity and state) | Unknown |
| **Open-Source** | No | Yes | Yes |
| **Platforms** | Windows | All desktop OSes, WebApp | WebApp |
| **Last update** | June 2023 | August 2023 | April 2015 |
| **Extra** | Open-source templates | Autodiscovery implemented only in version 1 | Project discontinued |

Table 2.1: Graphical tools comparison

**Tools and Coding**

Utilizing automated software can aid in the task of interpreting and evaluating system models, and comprehending vulnerabilities and possible issues within components, connections, and data. The primary challenge lies in articulating what needs to be modelled in a format that a computer can grasp, without constructing the actual system. Employing code stands out as the most straightforward method to circumvent this issue, presenting the following two key approaches.

- Threat Modelling from code: Use the program code/annotations inside the program and a taxonomy of threats to identify potential findings and produce results that should be interpreted by a human.
- Threat Modelling with code: Take an architecture or system description (with information about data, components and relationships) encoded in a form that resembles code and perform analysis for automated threat identification and reporting.

Usually, the output produced by both methods is a text document or PDF report [36].

For threat modelling from code, there are two major alternatives: Threatspec and ThreatPlaybook. Threatspec[7] provides a way for coders to document threat information alongside the code using annotations and generate detailed and useful documentation, including diagrams. Unfortunately, it requires code to exist and the design already solidified, shifting right threat modelling instead of left. Moreover, it requires security knowledge inside the development team or requires guidance from expertise, creating a scalability issue. ThreatPlaybook[8] is, instead, a framework to join threat modelling with Application Security Test Automation. It connects

[7]`https://threatspec.org/`

[8]`https://we45.gitbook.io/threatplaybook/`

a classical user story-driven approach (that includes user stories, abuse cases and threat scenarios, and security tests) with tools for the orchestration and validation of vulnerabilities. The main limitation of both of these tools is that they do not perform analysis or threat detection on their own [36] but require manual work.

To perform threat modelling with code there are different options, namely OWASP pytm[9], threagile[10] and TicTaaC[11] and a short comparison can be found in Table 2.2. The main idea is to perform pattern analysis on the system model using a database of threat information and a set of rules. This happens because computers can not infer or assume stuff, as it is common for humans after looking at a visual representation of a system. Using code to describe the application and automatically generate threats has multiple benefits like aligning with DevOps practices, allowing developers to be familiar with something they already know (coding) and tools that already use (IDE), placing security information where the code lives, tracking the changes, collaborating and having consistent results [36].

### Commercial Tools

For the sake of completeness, some commercial tools like ThreatModeler[12], Tutamen[13] and SecuriCAD[14] were included in the review performed, but their community versions are limited, no more supported or do not include competitive advantages compared to the other free or open-source tools described above.

[9]`https://github.com/izar/pytm`

[10]`https://github.com/Threagile/threagile`

[11]`https://github.com/rusakovichma/TicTaaC`

[12]`https://threatmodeler.com/`

[13]`https://www.tutamantic.com/`

[14]`https://nse.digital/pages/guides/Creating%20threat%20models/securiCAD.html`

| | **OWASP pytm** | **threagile** | **TicTaaC** |
|---|---|---|---|
| **Automatic Threat Discovery** | Yes | Yes | Yes |
| **Custom Threat Libraries** | Yes | Yes | Yes |
| **Require DFD** | No | No | No |
| **Reporting** | HTML, JSON | PDF, JSON, XLS | HTML, JSON |
| **Output (excluding threat list)** | DFD, Sequence Diagrams | DFD, Data Asset Risk Diagram, Various Threat Classifications | Various Threat Classifications |
| **Input Programming Language** | Python Objects | YAML | YML |
| **Open-Source** | Yes | Yes | Yes |
| **Platforms** | Linux, MacOS, Docker | Linux, MacOS, Docker, as REST-Server | Windows, Linux, MacOS, Docker |
| **Last update** | April 2021 | No official release, last commits around 2020 | May 2023 |
| **Extra** | 101 rules from Microsoft TMT and CAPEC, Object-oriented but without logic | RAA and DBP calculation, GitHub action available, CI/CD pipeline can stop deployment for unmitigated vulnerabilities | Jenkins integration, Quality Gates definition |

Table 2.2: Threat Modelling with code

### 2.4.2 New Threat Modelling Approaches

Alongside tools that help build threat modelling graphically or from code, it is possible to find new and innovative approaches that help perform such tasks in non-traditional ways, using totally different perspectives.

#### User Story Threat Modelling: It is the DevSecOps Way (Snyk)

DFDs and long threat modelling sessions do not work in DevSecOps as the design phase is limited and security can not be integrated any further left in the pipeline compared to when User Stories are defined. The main idea is to bring in business people during the sprint planning and let them discuss, for each story, what are the worst things that could happen. The process should start by defining the assets involved and then thinking about threats in a non-technical way (for example, instead of using STRIDE as classification, it is possible to think about malicious actions like theft, fraud, exposed data, or interrupted business). Only after the brainstorming phase, the high-level requirements are translated into technical countermeasures. This process helps the inclusion of product owners, who are the ones that safeguard the business and customer interests, to be part of the definition of security measures [10].

#### Integrating threat modelling with DevOps (Microsoft)

DevOps stands for Development and Operation and, when building threat models, it is fundamental to consider both those aspects. Starting with the Development part, mitigations and threat lists are the most valuable items that can be created during the threat modelling process. But, while mitigations are easy to store in a Task and Bug Tracking tool, as they can be treated as Task/Feature or Bug, it is not the same for the threat discovered. A solution can be achieved by using a different approach: the user stories should be extended in order to include a WITH-

OUT clause to the usual formulation "As a [who am I] I want to [what I want] so that I can [do something]." that can be mapped to one or multiple threats. In this way, they are somewhat encoded in the statement and security requirements as well are explicated in the stories. Including more and more security information inside the development tools used every day is beneficial as it is more convenient to add mitigation and satisfy security requirements during the actual implementation of the function than including them in a later stage. Talking about Operations, threat modelling can provide security-related information to facilitate Root Cause Analysis by integrating it with monitoring tools. By doing that, it is possible to use threat information and monitoring and events data to design specific controls that can detect undergoing attacks on the system and improve the incident response plan. Other pillars of such a solution are the introduction of the figure of Security Champions, interested in security and responsible for leading the threat modelling sessions, and knowledge bases that contain information about security in the specific environment (such as attack patterns and standard mitigations) and reference material for the sessions [37].

### Continuous Threat Modeling (AutoDesk)

Accordingly to AutoDesk, products evolve at a fast pace and also threat models should do so. It is important that they evolve together and that threat models become part of the Definition of Done of every User Story that includes security-notable events[15]. Even in this case, a figure called the "Curator" has the delegated of responsibility towards threat modelling activities queuing and updating. User stories related to threat modelling and security activities should be marked with appropriate labels in the tracking tool throughout the lifecycle. The process starts, as usual, with the creation of the DFD, and then the team is required to iterate over

[15]`https://github.com/Autodesk/continuous-threat-modeling/blob/master/Secure_Developer_Checklist.md`

the model and a some questions at the same time. The list is designed to help the developers focus on elements that require attention when dealing with security. The list can be found in the handbook provided by AutoDesk[16] and it is not exhaustive, but only a starting point. For each finding, a possible attack scenario should be defined, and a ranking, using CVSS or risk value, should be created. The threat model should be reviewed at least once a year in case of events such as architecture changes, additional input vectors, new services or components[38], [39].

### hTMM (Carnegie Mellon)

The hybrid Threat Modeling Method (hTMM) is a newly introduced framework developed by Carnegie Mellon's Software Engineering Institute in 2018. The innovative element here combines three techniques already presented: STRIDE, Security cards and Persona non grata (PnG). The main idea is to limit the disadvantages that the approaches present when used singularly to create a framework able to reduce the false positives and with consistent results while being cost-effective. hTMM consists of 5 distinct steps: identify the system to be threat-modelled; use Security Cards to brainstorm potential threats; filter the found attack vectors and scenario based on realistic PnGs; summarize and categorized the findings using STRIDE; conduct a risk assessment with a formal method [21], [32], [34].

### Hybrid Approach (SANS)

Another Hybrid approach that combined multiple classic techniques was developed by Sriram Krishnan, with the idea of designing a structured approach, that includes the optimum level of detail and is readable by all the stakeholders that need to be involved. By combining STRIDE, Attack Trees and Attach Libraries it is possible to overcome limitations such as the lack of countermeasures development, missing

[16]`https://github.com/Autodesk/continuous-threat-modeling/blob/master/Continuous_Threat_Modeling_Handbook.md`

abstraction levels about scenarios and completeness respectively. The process proposed is linear and starts with a Design Analysis, followed by Threat Identification and Categorization, where STRIDE and Attack Trees kick in, and it is concluded by a Threat Mitigation phase supported by Attack Libraries [40].

### NIST Data-driven approach

NIST Special Publication 800-154 proposes a data-centric approach for System threat modelling. The idea is to concentrate the effort on protecting data rather than the systems, as part of the risk management process. The publication includes a set of principles that can be integrated into other methodologies rather than suggesting a novel methodology to replace them. The steps included are mainly four [34], [41]:

1. Identify and characterize the system and data of interest (by including authorized data locations and how the data move inside the system, security objectives for the data and authorized actors);
2. Identify and select the attack vectors to be included in the model (attack vectors are essentially content, often of a malicious nature, originating from a source and then exploited by a processor);
3. Characterize the security controls for mitigating the attack vectors (for each attack vector identify mitigation controls, evaluate the effectiveness and estimate negative implications);
4. Analyze the threat model (using for example risk-scoring approaches).

### Rapid Threat Modelling Prototyping (RTMP)

RTMP approach, proposed by Geoffrey Hill and based on business aspects and focuses on critical assets, fits well in an agile environment as is guided by the Pareto principle (80% of the outcomes can be done with 20% of the effort). The first step of RTMP is to model the system (no need for a complete DFD) or use an already existing representation, and assign trust zones using a numeric convention (from 0 to 9) where the more critical the system, the higher the assigned zone. Then, use STRIDE to annotate the model using the zone rules defined by the method (that considers the trust zone previously defined). In this context, pay particular attention to the Elevation of privilege as it allows to perform all the other threats. Use a mapping like the one from STRIDE to OWASP Top 10 to define the vulnerabilities related to the threats and define the mitigations. Verify that the mapped results apply to those situations and implement at least one mitigation for each of them. In an Agile setup, most of the process should be performed in sprint 0 and more information and threats should be added iteratively in the subsequent ones [42], [43].

### Agile Threat Modeling in 5 Simple Steps

Practical DevSecOps proposed a simple framework, called KISS, that tries to keep things simple and straightforward when deploying in an agile environment. KISS is composed of five steps, namely Define, Identify, Rank, Address, and Validate, which fits the classical four-question framework. The names of the steps are self-explicative in the context of threat modelling. For each step, the same set of questions is defined to help perform threat modelling and organize the process. The questions are:

- *What is needed?* Defines the prerequisites and the participants involved in each step;
- *When could it be done?* Identifies the most suitable agile events or ceremonies when the step can be performed;
- *How could it be done* Suggests ideas and tactics on how to perform the step;
- *When do we know we are done?* Tries to highlight when the step is complete and the output is satisfactory, even if the whole process is continuous and

should be performed periodically.

Regarding the identification of the threats, the method suggests an extensive list of options spacing from lightweight methods to attack trees and gaps identification with respect to security frameworks.

### 2.4.3 Suggestions for implementing Threat Modelling

Experts who deal with threat modelling every day and companies that implement successful security processes are the stakeholders to consider to get suggestions and different opinions.

Starting with one of the most authoritative sources, the Threat Modelling Manifesto mentioned before, defines a complete set of principles, values, patterns and anti-patterns to follow when dealing with such a security process. The concept of threat modelling should be perceived as a mindset centred around fixing design problems rather a compliance requirements, as a collaboration effort rather than a list of steps to perform on a tool, and as an ongoing process over a one-time procedure [6].

Implementing threat modelling in the real world is not only about technical skills and elements but also requires interpersonal skills and organisational support. Adam Shostack, in one of his white papers [13], compared threat modelling to the Jenga game where each block is a skill, technique or tool needed to build a stable tower, namely process. Each block has an associated cost and a key question is to define how many building blocks the company should have or can afford to create a program that stands up on its own without collapsing. Identifying and balancing those elements is not a trivial task, and the suggestions listed in Table 2.3 can direct the focus [29], [44], [45].

| **Focus** | **Suggestion** | **Activities and Benefits** |
|---|---|---|
| Interpersonal | Assemble the right team | Include multiple personas with different skill sets<br>Initially invite a security expert<br>Involve remote team members |
| Organisational | Use consistent and systematic approach | Allow knowledge sharing<br>Improve scalability and reusability<br>Make the process independent from the people involved |
| Organisational | Align delivery methodology | Improve the overall workflow of software delivery and development |
| Organisational | Use existing workflow tooling | Integrate already used security and management tools<br>Choose carefully threat modelling methodologies and tools<br>Treat threats like other risks and bugs |
| Organisational | Break the workload down into smaller parts | Perform threat modelling at the feature level |
| Organisational | Distribute ownership | Avoid centralizing governance and responsibilities to a single person/team |
| Interpersonal | Motivate the Team | Provide education, using real examples, and explain the purpose of threat modelling |
| Organisational | Do not let paralysis stop you before you start | Start with new features and pick up later already existing ones |
| Organisational | Know the "Definition of Done" | Use a risk-based perspective to create a balanced process<br>Include a moderator to steer the discussion on the right topic and avoid overfocus<br>Schedule short, time-bounded and efficient meetings |
| Organisational | Document the results | Store and share the outcomes in a centralized knowledge base |

Table 2.3: Suggestions for implementation

### 2.4.4 Use case in a real corporate environment

Implementing threat modelling in a big corporate environment, such the one of an international bank, with 630 development teams (mostly following DevOps practices), 4000 engineers and 3000 active applications is a non-trivial task. However, according to a security expert with experience in both penetration testing and DevSecOps practices and who is working in such a context as a DevSecOps enabler, it is still possible.

Starting with the figure of DevSecOps enabler, this role is more related to people and business compared to classical DevOps engineers, with the goal to help create good DevOps processes and security mentally inside the development teams. The enablement can happen on multiple levels, but building a pipeline that includes security elements is the priority. In fact, designing secure applications in a corporation that deals with high-sensitivity data and guarantees maximum integrity should be the starting point and threat modelling is the first step to achieve that. The following points include some suggestions on how a threat modelling process can be designed, without disclosing too many details due to confidentiality.

- Use tools that are already familiar to developers to share knowledge and examples, and store the results;
- Create guidelines, checklists, and learning materials that can be consulted both with the guidance of the enablers, during team meetings or by the developers on their own;
- Include the enablers in the first sessions and at the same time develop templates to guide the teams when performing the tasks without security experts. Templates can include threat tables (with targets, classification, controls, and countermeasures), DFDs and sketches, access control, dependencies and penetration testing scoping;

- Enforce the use of the template for creating threat modelling of the application/part of the application the team is developing, but let the engineers decide the pace of update based on the release cycle and the features created. The enablers just verify that the model reflects the real state of the system periodically;

- Create interactive enabling quiz to break the ice and test developer about security concepts and threat modelling;

- Deploy a pipeline that reports security issues, monitors metrics about vulnerabilities and blocks the release in case of major problems. These security gates should be created for every team;

- Formalize governance, translate the security requirements into policies and make the CEO or C-levels approve them. Every engineer and product owner should sign and agree with those obligations that define guardrails and a baseline for security.

# 3 Current SDLC at Company Z

Before proposing a novel solution for threat modelling inside Company Z, it is relevant to analyse the current situation and implementation of the SDLC and threat modelling process.

Company Z operates as a mid-sized business within the IT sector, counting a workforce of approximately 150 technical employees in the R&D team. The flagship product offered is a web application delivered through the Software-as-a-Service (SaaS) model to a broad range of businesses, ranging from Small and medium-sized enterprises (SMEs) to big corporations. The core feature of the software, namely the DAM, can be extended by additional modules to extend and improve the experience. Despite being totally designed and developed in-house, the application is deployed in one of the major public cloud services.

As common in the software industry, the development methodology in Company Z follows the Agile principles and it is divided into sprints, following a release cycle of six weeks. The R&D department that is in charge of all the technical aspects of the web application is divided into multiple teams, each of them in charge of a single component or aspect of the service. In total, it is possible to find more than 20 teams, called squads, across the company and located in different locations. Inside the same team, skills are covered, in order to be able to rely on other squad members compared to being dependent on others, and among the roles, it is possible to find:

- A product manager and a product owner;

- A user experience (UX) researcher and a product designer;
- An agile and a delivery leader;
- Various front and back-end developers, DevOps and quality assurance (QA) engineers.

For each team, among the experienced engineers, a technical champion is nominated the maximize the benefits of the technology used, support other members and improve knowledge and skills development. Excluding the developers, almost all the other people usually work or support multiple teams at the same time.

## 3.1 Software Development Life Cycle (SDLC)

Before starting the development process, it is necessary to define what features the team needs to work on: they can be new features, bug fixing or improvements of existing parts of the system. Different stakeholders and customers can populate such a list and make different requests, that are prioritized based on the company goals and strategy. Once the features are selected, the SDLC can start.

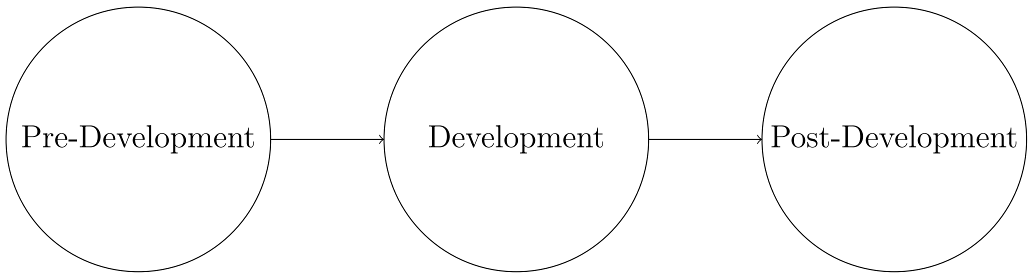


Figure 3.1: SDLC workflow

As can be seen from in Figure 3.1, the process is composed of three main phases in Company Z.

### 3.1.1 Pre-Development

The process starts way ahead of the development phase: at the beginning of each quarter, the product owner collaborates with the members of the squad to outline a high-level draft of the features' requirements that will be developed in the next couple of months. For this reason, it is called kick-off and usually lasts a couple of days. Once this preliminary step is finalized, the team should have a general overview of the goals for the next months are should start working on them. The work is organized using Agile methodology, so the time is divided into sprints. For each of them, two standard ceremonies take place: sprint refinement and planning.

In the first one, led by the agile leader, the uncertainties of the features developed in the upcoming sprint(s) are discussed and particular attention is given to elucidating the problem, the motivation and the outcome of each task. The overarching aim is to ensure that the developer assigned to the ticket comprehends exactly what needs to be done. Moreover, the story points, that estimate the effort required for the task, are assigned.

In the second meeting, the planning session, the team determines the actual tickets that need to be completed in the next sprint, based on its velocity. Threat modelling sessions are also conducted for the tickets that require it, but more details about the security tasks are provided in section 3.2.

When the requirements are clear, the designs, both architectural and graphical, start to be developed.

### 3.1.2 Development

The central phase of the process is the actual development.

The engineers can start to pick up the tickets as soon as the designs are ready. Code is created and changes are tracked using a distributed versioning control system. When programming is done, the review process starts with the creation of a pull request (PR), that automatically enforces some activities:

- Code quality, employing a checklist, to detect common issues in secrets and log management, error handling, data storage, input validation, and access control;
- Static code analysis to automatically check security vulnerabilities and bugs;
- Unit tests, written by the QA engineers and performed automatically, to check the correct functionality. They should cover the majority part of the new code;
- Dependency checks, to control known vulnerabilities in the third-party libraries and issues related to licences;
- Presence of documentation.

In any case, the PR is reviewed manually by at least two other developers. In case of issues, the new code is rejected and, after the fix, the same checks are performed again until the quality meets the standard defined. If all looks fine, the code can be prepared to be deployed and released on a test environment, called state, which is a replica of the production one. This practice is typical in DevOps environment and it is useful to test how features perform in a similar context. QA engineers are responsible for verifying the release in such an environment. If needed, penetration testing of the new code is performed.

### 3.1.3 Post-Development

After successful deployment on stage, without problems of security and quality testing, the release is ready to be deployed to production. In case of problems, instead, the code is reverted, fixed, and added to the next release, following again the same cycle. To make sure the features and bug fixes are released correctly and working properly, they are constantly monitored with metrics and triggers, using dashboards to check anomalous situations.

## 3.2 Secure SDLC

As can be seen, many quality gates are defined throughout the life cycle in order to avoid the release of low-quality code. Having controls such as code review, automatic and manual testing, static code analysis smoke test and automated checks with building pipelines, reduces significantly the possibility of including bugs or security vulnerabilities inside a release and is as important as the actual development of the code. Narrowing down the focus on the security activities included, it is possible to recall the previous three steps depicted in Figure 3.1 and break down the various security-related activities of each phase.

### 3.2.1 Pre-Development Security

As evident from the preceding chapters, the primary security task preceding the development phase is threat modelling. Enhancing the existing process at Company Z constitutes the central objective of this Master's Thesis so it is vital to have a clear vision of what it looks like and how it is currently performed.

The current process was introduced in 2019 from a joint effort of the Information Security team and the most security-minded Software Engineers of the company, with the initial recommendation to use it only for major changes or new features developed from scratch. For small improvements and minimum changes, a small brainstorming session using the 4 questions of Adam Shostack and with a couple of team members is needed, with the goal of gathering and listing the threats with their acceptance criteria (e.g. conditions to be fulfilled, like mitigations).

The whole process is triggered when a new ticket is picked up for the next sprint during the pre-development ceremonies described in subsection 3.1.1. For every story, the team reasons whether threat modelling is genuinely necessary, guided by specific security and privacy criteria, specifically designed to assist software engineers. To prevent undue complexity, they consist of only two straightforward, easy-to-answer, questions about the existence of user interactions and the involvement of Personal Identifiable Data (PII) within the workflow. In practice, these two inquiries are narrowed down to more specific ones, alongside examples, in the internal documentation of the company. Based on the combination of the results of the questions, a final outcome regarding the need for the threat modelling session is produced. The decision is reviewed by the security team and stored, using an appropriate label, inside the ticket tracking system. If the decision was not to build the threat model, the process continue as described before, otherwise, the session should be organized during the sprint planning and must include the whole team and a member of the security one. The use of labels throughout the process is enforced to simplify the communication between the development squads and the security people.

The structure of the meeting can vary based on the participants and the designated leader. However, the fundamental stages encompass identifying threats, precisely defining them, in a manner that offers clarity to developers and quality assurance personnel, assigning a severity and formulating a strategy to mitigate these threats. For creating DFDs and facilitating the identification of threats following the STRIDE classification, OWASP Threat Dragon is recommended.

A repository is used to store the developed diagrams and serves as a central location for development teams to share findings and knowledge. The repository includes scripts to automatically upload the diagram and threats identified to the ticket associated with the function/story. Once the model and all the documentation are created and reviewed, the label on the ticketing system can be updated to reflect the current status.

### 3.2.2 Development Security

During the development, all the best security practices are followed, like the one included in the OWASP Top 10, and a particular focus is placed on implementing the mitigations defined during the threat modelling session.

### 3.2.3 Post-Development Security

After the development phase, the initial security assessment occurs through the code review process. Adhering to the 4-eyes principle and involving other engineers to scrutinize the code represents a crucial method for preventing the introduction of vulnerabilities into the codebase. Integrating automated security checks into the DevOps deployment pipeline is another essential practice. In the context of Company Z, one of the industry's leading tools is employed, which offers the capabilities of:

- SAST (Static Application Security Testing) that analyses the code as it is written, without executing it;
- SCA (Software Composition Analysis) that analyses the dependencies declared inside the application;
- Container analysis that examines the running environment where the application is deployed.

If none of the security checks indicates issues with medium, high or critical CVSS value, the deployment can proceed and the low-risk findings are added to the development queue, and prioritized accordingly to the Service Level Agreement (SLA). In case of major discoveries, the pipeline is blocked, the code reverted and the problems should be fixed by returning to the development phase.

The last step before releasing the code, namely penetration testing (pen-testing), is performed by the internal security team when needed. For every ticket that underwent threat modelling, penetration testing is conducted to assess the quality of implemented vulnerabilities, however, there are instances where other stories might necessitate it. If the outcome is satisfactory, all the security checks needed are complete and the code is ready for release. Otherwise, it should be adjusted to address the last findings.

A more schematic overview of the security tasks and the Secure SDLC can be found in Figure 3.2 and Figure 3.3.

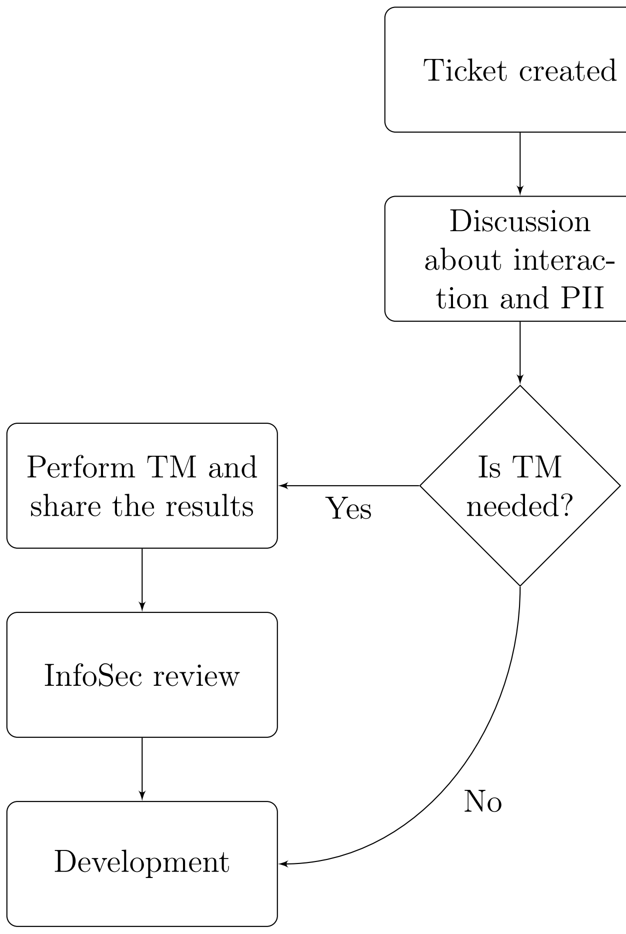

Ticket created
Discussion about interaction and PII
Is TM needed?
Yes
Perform TM and share the results
InfoSec review
Development
No

Figure 3.2: Secure SDLC workflow - part 1

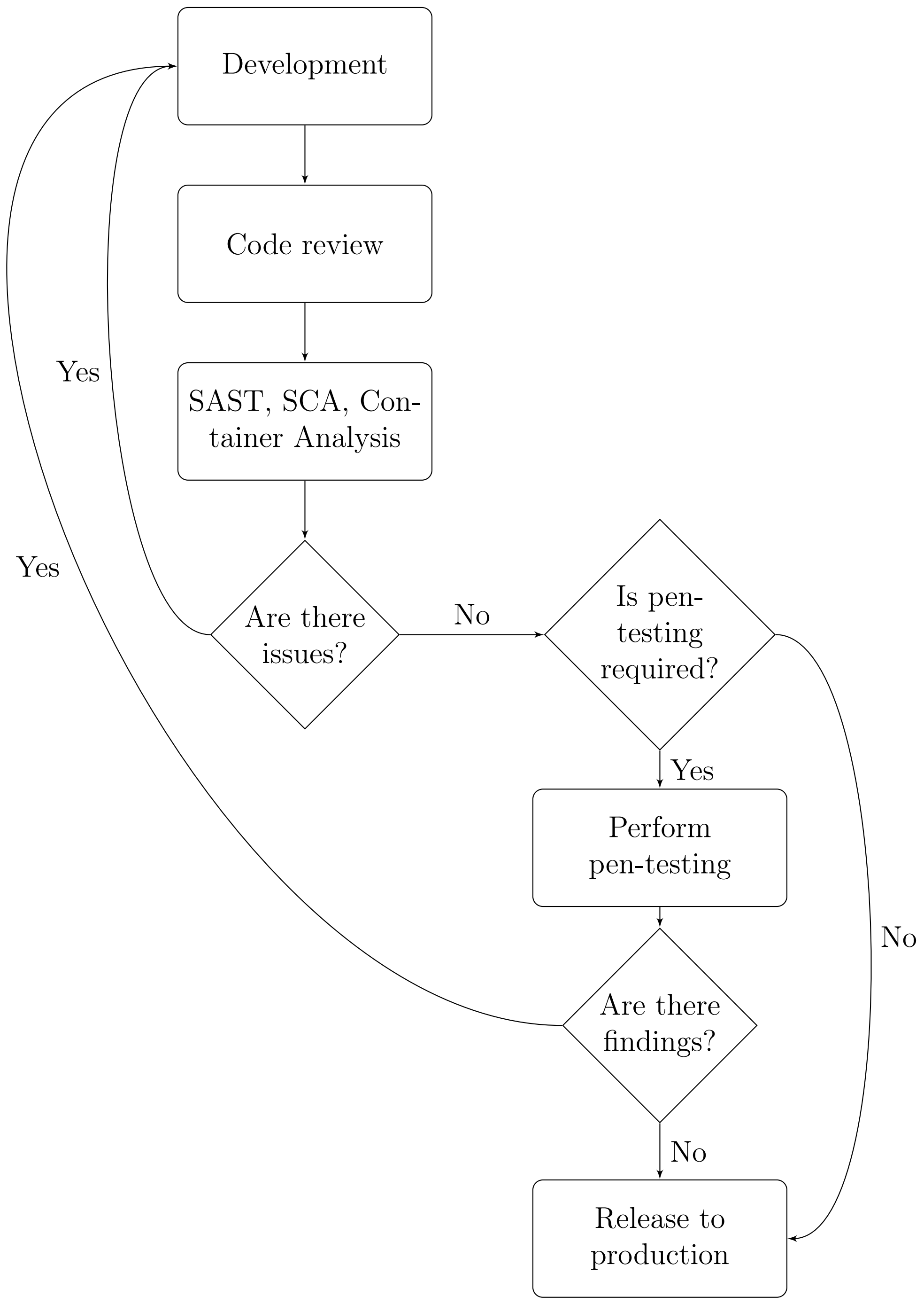


Figure 3.3: Secure SDLC workflow - part 2

## 3.3 Considerations

The approach chosen by Company Z for the threat modelling part, and application security in general, is quite traditional as described in this chapter. The incorporation of tools like OWASP Threat Dragon brings added value to the process, however STRIDE model might not offer a swift identification of threats as it demands a certain level of technical expertise and can potentially lead to extended and resource-intensive meetings. Moreover, the process itself includes different discussion topics during sprint meetings and comprehensive reviews conducted by the Information Security team. The combination of all these elements results in a process that does not meet the expectations of the company, does not fit the SDLC and slows down the release cycle.

# 4 Process requirements

The considerations included in the previous chapter highlight some of the limitations of the current threat modelling process implemented at the moment on Company Z. They were derived just by looking at the process and the various phases, but this is usually not enough to understand the real paint point of the stakeholders that are actually using the process. The initial reservations regarding the process's inefficiency and the general lack of engagement were voiced by the company's security employees, due to the bland involvement of the team in threat modelling sessions and reviews, as required by the process. This situation led to their promotion of the project behind this Master's Thesis.

## 4.1 Data collection

To validate the accuracy of the security team's concerns, the most effective approach was to address the core issue directly. This involved gathering insights from individuals engaged in the actual development of the application and those overseeing the coordination of work, namely employees of the R&D squads. To gain real insights over a large quantity of static data, qualitative methods were preferred over quantitative ones, and in particular interviews over questionnaires.

### 4.1.1 Participants

The interviewees were carefully chosen based on their experience inside the organization, their attitude toward security and the squad to which they belong, and, most significantly, their job titles. Collecting different points of view from a set of heterogeneous people was important to be able to balance and ponder the opinions and suggestions.

| **Role** | **Experience** |
|---|---|
| Back-end engineer | Senior, Staff |
| Front-end engineer | Senior |
| Product owner | Intermediate |
| Ethical Hacker | Senior |

Table 4.1: Job title of the employees interviewed

The Table 4.1 illustrates the roles of the interviewed people alongside their experience. It is worth noticing that some of the people selected were the ones responsible for the creation and implementation of the current process.

The number of interviews was not established a priori, but dynamically during the process: the approach was selected following the idea of Jakob Nielson. For him, when testing the usability of an interface, conducting interviews with an increasing number of individuals does not necessarily result in a greater understanding, but rather can lead to diminished insights, as the same information is observed multiple times [46]. Following this logic, as soon as the same details were repeated a couple of times during the interviews, the process was concluded. In total, given the limited number of development teams within Company Z, a small number of interviews was needed (four with software engineers belonging to different squads, one with the internal ethical hacker and one with a product owner) to have a comprehensive snapshot of the situation.

### 4.1.2 Style and questions

The type of style chosen for the interviews was the semi-structured one because provides a good balance between the richness of data collection of the unstructured ones and the replicability of the structured ones. They are guided by a script however they enable the interviewer to explore more in-depth interesting topics using additional questions created on the fly. Three different scripts were created, one for each of the roles of the interviewees, and the completed list of questions can be found in Appendix A.

The questions were designed with particular goals in mind and their motivations are listed in the following bullet points.

- The questions inserted in the point "SDLC, threat modelling and deployment process" of each template are needed to comprehend the current development process and its security elements, from the perspective of directly involved stakeholders. They also try to grasp information related to how threat modelling is perceived, and how much the process is known and used in reality.
- The set named "Experience with threat modelling" aims at analyzing the experience of the stakeholder in organizing and performing the sessions and, at the same time, gaining insights without directly asking for feedback about the process.
- The final questions try to gather explicit feedback regarding the efficacy of the present process, identifying both its strengths and weaknesses, and acquiring some suggestions based on previous working experiences.

The interview template for the project owner is differently organized compared to the other two while maintaining the same topics and their motivations, as it aims to understand the perspective of non-technical people. The key distinction is the additional set of questions introduced to collect information on management's perspective of security and the balance of security requirements and features implementation.

## 4.2 Data analysis

Raw data coming from interviews must be processed and analysed to extract knowledge. In this instance, the interview transcripts underwent processing with a qualitative methodology, and more precisely, a thematic analysis helped to extrapolate the recurring pattern and point out the most important information. To simplify the discussion, three macro categories were defined and a summary of the results can be found in the following subsections.

Before presenting the findings, it is worth noticing that the focus was on the main limitations and the pain points of the stakeholders when dealing with threat modelling and the main objective was to collect information to overcome them. All the positive feedback related to the other security practices included in the SDLC is not reported for brevity.

### 4.2.1 Threat Modelling process

Starting the examination with the threat modelling workflow provides insight into the overall perceptions of the structure of the process itself. An overview of the findings can be found in Table 4.2.

As evident, the impressions on the threat modelling process among software engineers are not uniformly positive, and there exists significant variation among different teams in terms of adoption and security mentality. Additionally, product owners tend to deprioritize such activity due to the lack of understanding of its ROI.

Despite these premises, it is notable that threat modelling is still conducted by the engineers for new and substantial features.

| **Observation** | **Role of the supporters** |
|---|---|
| The process is known or partially known | Software Engineer, Ethical Hacker |
| Knowledge and awareness gap about the process and the value of threat modelling in general | Product Owner, Software Engineer |
| Threat modelling is not required for the majority of tickets or features in development by particular squads | Software Engineer, Product Owner |
| The process is not used as it is and more lightweight and unstructured methods are preferred | Software Engineer, Ethical Hacker |
| The questions included in each ticket to determine whether or not to execute the session are annoying and not used | Software Engineer |
| The current workflow is not well integrated, structured and enforced | Software Engineer |
| The current way of sharing knowledge does not work | Software Engineer |

Table 4.2: Thematic analysis: Threat Modelling process findings

### 4.2.2 Threat Modelling session

After analysing the process, it is interesting to discover what are the thoughts about the threat modelling sessions. The results are presented in Table 4.3.

By having a quick look, it is clear that not enough guidelines are available regarding how the session should be performed. It also emerges that the squads are not trained enough to complete the activity without the involvement of an expert, who is usually not included due to schedule constraints.

| **Observation** | **Role of the supporters** |
|---|---|
| The sessions are not performed frequently | Software Engineer, Ethical Hacker |
| The InfoSec team is rarely involved actively, only the material is reviewed | Software Engineer, Ethical Hacker |
| The findings are always the basic ones | Software Engineer |
| A lot of manual work is required and the sessions are time-consuming | Software Engineer |
| Even different approaches (e.g. card games) lead to basic results | Software Engineer |
| It is not clear what to do, where to start and what to expect | Software Engineer |

Table 4.3: Thematic analysis: Threat Modelling sessions findings

### 4.2.3 Suggestions

The suggestions and recommendations collected by the stakeholders are mainly solutions to alleviate the problems highlighted in the previous sections. To clearly express such a relation, Table 4.4 presents a mapping between the problems and suggestions.

| **Suggestion** | **Related problems** |
|---|---|
| Automation | Manual work, time-consuming |
| Guidelines and checklists | No information of where to start and what is the expected outcome |
| Central knowledge base | Limitations of knowledge sharing and relative gap |
| Increase the scope | Questions are annoying, not every ticket requires TM |
| Improve awareness and governance | Awareness gap, disparity among different teams |
| Introduce Security Champions | Awareness gap, governance problem |

Table 4.4: Thematic analysis: Mapping between problems and suggestions

## 4.3 Requirements

The comprehensive analysis described in section 4.2 is an excellent starting point for defining the requirements of the new threat modelling process. At the same time, the approaches and suggestions defined in the literature review can help refine and fine-tune them. The requirements identified were organized into four main pillars, which are presented in the following list. For each of them, it is necessary to work on different sub-tasks and activities.

- Governance: The value of threat modelling should be clear, the process uniform among the teams and the ownership distributed correctly among the stakeholders.

- Process management: The method for determining whether the session is required should be enhanced, and the whole threat modelling workflow should be enforced, as happening for other security activities.

- Time and Automation: Choosing the right tools that integrate well in the SDLC, automating the process following DevOps and Agile principles and saving time, should be a top-level priority.
- Knowledge: Building guidelines and consistently sharing the results should lead to improvement of the overall outcomes and experience.

The redesign of the process should also incorporate aspects to promote scalability and adaptability to various needs and should be easy to update.

# 5 Design and verification methods

Once the literature is reviewed, the current system is clear and the requirements are defined, the only missing part is to design the new threat modelling process. A set of guidelines, suggestions and recommendations to overcome the limitations should be established and structured into a novel approach for threat modelling that can be applied in Agile environments.

## 5.1 Design of the process

The ideation of the solution started by considering the main four categories of requirements defined in section 4.3. For each of them, several strategies for addressing the gaps are put forth, alongside their motivations.

### 5.1.1 Governance

As reported by the expert in subsection 2.4.4, governance is a frequently overlooked topic when discussing security, but it is critical within a corporation with hundreds of employees. Three of its crucial components come into play and should be considered: responsibilities, accountability and awareness.

#### Responsibilities

Not every engineer or product manager is as security-minded and responsible toward data protection as the other. It is human nature to have different interests and prioritize some activities compared to others, but inside a company, a security baseline should be established and followed by everyone. Formally delineating security requirements, activities and guardrails, obtaining approval from the executives of the corporation, and securing the developers' signature represents the simplest method to establish concrete and binding obligations.

Among all the security activities that should be included in this document, the focus here is on sole threat modelling. To ensure clarity regarding this activity, the roles and responsibilities of both company teams and individual members within development squads should be defined, and the RASCI matrix is the perfect framework for this task. The name is a mnemonic way to describe the roles that should be assigned to each employee, namely Responsible, Accountable, Supportive, Consulted, Informed [47]. The matrix depicted in Table 5.1 is the specific one proposed for threat modelling inside Company Z.

| **Role** | **Team(s) or Individual(s)** |
|---|---|
| Responsible | Security Champion |
| Accountable | Product Owner |
| Supporting | Information Security team, Development squad |
| Consulted | Information Security team |
| Informed | Information Security team, R&D management team |

Table 5.1: Thematic analysis: Mapping between problems and suggestions

As can be noticed, the figure of Security Champion is introduced in the matrix as the main responsible for the threat modelling process, with a job that resembles for some aspects the one of a project/product manager with a focus on security aspects. This concept was already mentioned in the literature review and can be seen as a specific evolution of the Technological Champion, that acts as a bridge between security and development teams. Currently, Company Z's squads do not encompass this specific role, however, with the ongoing expansion of the number of developers, its necessity is escalating in parallel. While designating these responsibilities to a Technical Champion is feasible in smaller setups, it becomes impractical in situations where this individual follows different development teams and coordinates multiple activities. The other main actors involved in the process and sessions are the supporting ones.

Along this line of defining the responsibilities, it is also worth mentioning that the ownership of each piece of code should be defined and assigned to a specific squad.

**Accountability**

Defining responsibilities is not totally useful without tracking the progress and the results. In work environments, it is common to define KIP (Key Performance Indicator) or OKR (Objectives Key Results) for monitoring the advancement of projects and personal development and employees are accountable for the performances to the management. The same indicators can be applied in the domain of secure application development to establish objectives and track accomplishments. Defining metrics like:

- frequency of threat modelling sessions;
- number of features that require threat modelling versus the number of sessions organized;
- number of critical/high/medium/low vulnerabilities discovered versus the one already fixed;

and collecting these data for each development team should be the strategy. The information gathered should be used to build dashboards and reports and monitor the overall security posture of the application, not only during the deployment but also during development. Both current and prospective customers and external auditors can take advantage of these tools to certify the validity of the security program.

All of this evaluation system comes down to the cost of building a new pipeline, integrating it into the already existing one or employing a tool. Various types of solutions, already in use by numerous businesses including Company Z, can facilitate the collection and storage of such information. Examples include Human Resources and Team Performance Management systems, as well as GRC (Governance, Risk, and Compliance) tools, which might even be capable of automating the gathering of pieces of evidence.

Leveraging reports and dashboards is essential to periodically assess and audit both the security status of individual teams and the overall process and, when necessary, to update the process or implement corrective measures.

### Awareness

Threat modelling software, new feature or even improvements of old functions is not possible if not enough knowledge about the process are available or no instructions are provided. Training, right from the start, all the members of development squads about security and threat modelling is vital. Mandatory online lectures and material with real case scenarios, workshops and hands-on during threat modelling sessions are only some examples of what it is possible to offer to improve awareness about threat modelling. At the same time, the role and the impact (even of financial aspects) that security has nowadays on a business should be clear to everyone.

More intensive and detailed training should be provided to the designated Security Champions when needed, as they should cover the role of "enablers" for threat modelling and all the security tasks.

### 5.1.2 Process management

Improving the management of the threat modelling process requires shifting left security, increasing the scope and enforcing the workflow.

To contextualize and better explain, a development ticket represents either a single short task to be performed or a small subpart of something bigger, like a feature. Collections of related sub-tasks are called epics in Agile terminology. Triggering the discussion to determine whether threat modelling is required using the two broad questions for every single ticket, as pointed out during the interviews, is ineffective. This is particularly true when dealing with sub-tasks of an epic, as their scope is too small and the goals are too related to each other to consider performing threat modelling separately. Moreover, having to deal with such questions and discussions every couple of weeks during the sprint ceremonies only makes the meetings longer and upsets people who are willing to start the development. The whole design of the pre-development workflow seems limited and does not scale well in the considered context.

The first suggestion is to get rid of the questions*per-ticket* and move them to a *per-epic* level. In this way, the reasoning should happen for each new feature, and not for single tasks, meaning that the scope is increased. The second one is to move the decision process away from the sprint ceremonies and place it at the kick-off. At that point, the requirements of the features should be already drafted and determining if enhanced security is needed is possible. This anticipation means shifting even more left the first decisions about security. Additionally, only new features and big architectural changes should be part of the discussion. Defining security requirements this early has the benefit of helping project managers define threat modelling sessions way in advance and shorten the sprint meetings.

Going back to governance about the process, the whole workflow of threat modelling should be enforced in the development and building pipeline and the code should not be released if insufficient evidence is provided. Not performing threat modelling, when required, should be considered as dangerous as intentionally inserting a bug in a production system.

### 5.1.3 Time and Automation

Automating and implementing everything-as-code are two of the pillars on which DevOps is built. Among all the solutions presented in the literature review, only threat modelling with code is able to offer both aspects at the same time. Employing one of those tools over Threat Dragon has the benefits of speeding up the sessions, bringing in automatic threat discovery and simplifying the maintenance of the model, as it can be stored alongside the code.

Among the three main tools available for threat modelling with code mentioned during the literature review, namely pytm, threagile and TicTaaC, the decision felt on the first one for the following reasons:

- support by the OWASP Foundation and the community;
- number of rules already included (more than 100 from CAPEC);
- simplicity of integrating custom threat libraries;
- availability in Docker Containers;
- use of Python objects only, without the need for real coding skills.

As a bonus point, some of the Software Engineers interviewed already tried the tool and suggested it as a possible option.

The main goal of pytm is to help development squads create diagrams, identify threats and keep threat models updated with minimum effort. The steps for achieving this are the following.

1. Include pytm Python 3 library;
2. Instantiate a threat modelling object that will contain the description of the whole system;
3. Model the various components by instantiating a collection of objects such as Process, Server, Datastore, Lambda, Trust Boundary, and Actor;
4. Identify the specifics of the system and set them by using the attributes included in each object;
5. Define how the data flows inside the model, the protocols used and the ports;
6. Run the tools.

A simple command enables automated threat detection, report and diagram generation, and risk analysis. By following these simple steps, developers can continue to write code for threat modelling, free of the usage of graphical tools, that require manual manipulation of stencils. In case of changes to the model over time or additions to the threats library, by simply executing the tool, is possible to obtain the updated material and threat list. More information about pytm and its usage is presented in section 5.4.

### 5.1.4 Knowledge

Awareness is not the only way to increase knowledge about a process or a technology. Learning new information and understanding the value of an activity can also happen by looking at examples and reports of results. For these reasons, it is important to consider how knowledge is built and shared when dealing with threat modelling. In this area, three different types of improvements can be identified compared to the actual process.

### Documentation

Incorporating user-friendly documentation that encompasses examples, recordings and detailed explanations of previously conducted sessions should be considered a fundamental requirement. This documentation should also outline the process, its distinct phases, and the expected outcomes. Particular emphasis should be placed on the utilization of the results of threat modelling to formulate security requirements, implement security mitigations, and develop tests. Recommendations on how to implement the identified countermeasures should also be presented using code snippets following the recommendations of well-known standards such as OWASP and CWE.

### Central knowledge base

Defining a centralized place to store the results and the documentation in a way that is easily accessible by all the members of R&D teams can facilitate the consultation of the material, with benefits on the process. Storing the results inside the tickets was proven ineffective, as they are difficult to find and not centralized, and using a repository inside the development environment is not easy for non-technical individuals. A compromise can be the use of a wiki tool.

### Guidelines for threat identification and Threat Library

To facilitate the sessions and the threat elicitation phase the first suggestion is to use a sort of checklist with security topics that should be considered with particular attention when performing threat modelling. A good example of that is the set of questions included in the AutoDesk Continuous Threat Modelling Handbook presented in section 2.4.2. The list includes questions related to topics like Authentication and Authorization, Access Control, Auditing, Cryptography, Injection and many others that can be beneficial to avoid forgetting some security practices and start the discussion.

Once the process is more mature, the AutoDesk questions can be tailored to the context of Company Z and a Threat Library could be built. Different sources can be used to populate the library, like the major threats collected in the previous sessions, the latest vulnerabilities that affect the technologies used by the application, and the requirements defined by the security certifications. The library itself can facilitate the sessions in the first place, but also be fed to pytm to improve its performance in automatic threat detection by providing more contextualized results.

### 5.1.5 Final process

By analysing all the previous points, a genuine question arises about the necessity of guidelines for the sessions, considering the automation provided through pytm. The motivation behind that can be understood by putting together all the elements and defining the new process as a whole. The newly proposed approach is hybrid and combines aspects of multiple techniques and all the suggestions presented before.

The main idea of the hybrid solution is to use the results provided by the automated tool as a starting point for the threat modelling session, as the results are not perfect yet and can contain false positives. The meeting has the goal of fine-tuning the findings and, eventually, enhancing them using the questions proposed by AutoDesk. The proposed solution can be easily integrated into the remaining part of the Secure SDLC described before and depicted in Figure 3.3. As suggestions, additional controls, like the implementation of the mitigation, can be included in the code review and the way threats and mitigations are handled can be improved. Treating threats as bugs and countermeasures as features can simplify their prioritization and management as both concepts are well-grasped by developers.

The overall workflow is depicted in Figure 5.1. The colours of the blocks represent when the particular activity should be performed.

- White: during the kick-off;
- Purple: during the Pre-Development phase and Agile ceremonies;
- Gray: during any of the Agile ceremonies or whenever it is possible to schedule the session in the Pre-Development phase;
- Green: during the development.

The process should be launched at the beginning of every quarter during the kick-off, by every R&D team. In case of high-priority requests coming in after that, the workflow is flexible enough to be started right away, without waiting months. In terms of participants, the process of converting the design into Python objects and executing pytm falls under the responsibility of the Security Champion, supported by architecture experts. However, the fine-tuning and improvement of the outcomes should be carried out in a session involving the entire development squad and information security personnel, as specified in the RASCI matrix.

It is important to notice that the process outlined here focuses only on the initial creation of the model. However, as emphasized throughout this comprehensive document, the model should not remain static, but rather, it should be regularly updated when code changes or alterations in requirements happen. The Security Champion should be responsible for checking that the updates are performed and defining which of the activities should be performed.

Regarding the implementation, the new process should be used at first for new features and tasks, and later on, all the legacy codebases should have a model and a comprehensive list of threats. The workflow depicted here does not include the activities related to governance and knowledge described in the previous point, but their implementation has the same importance.

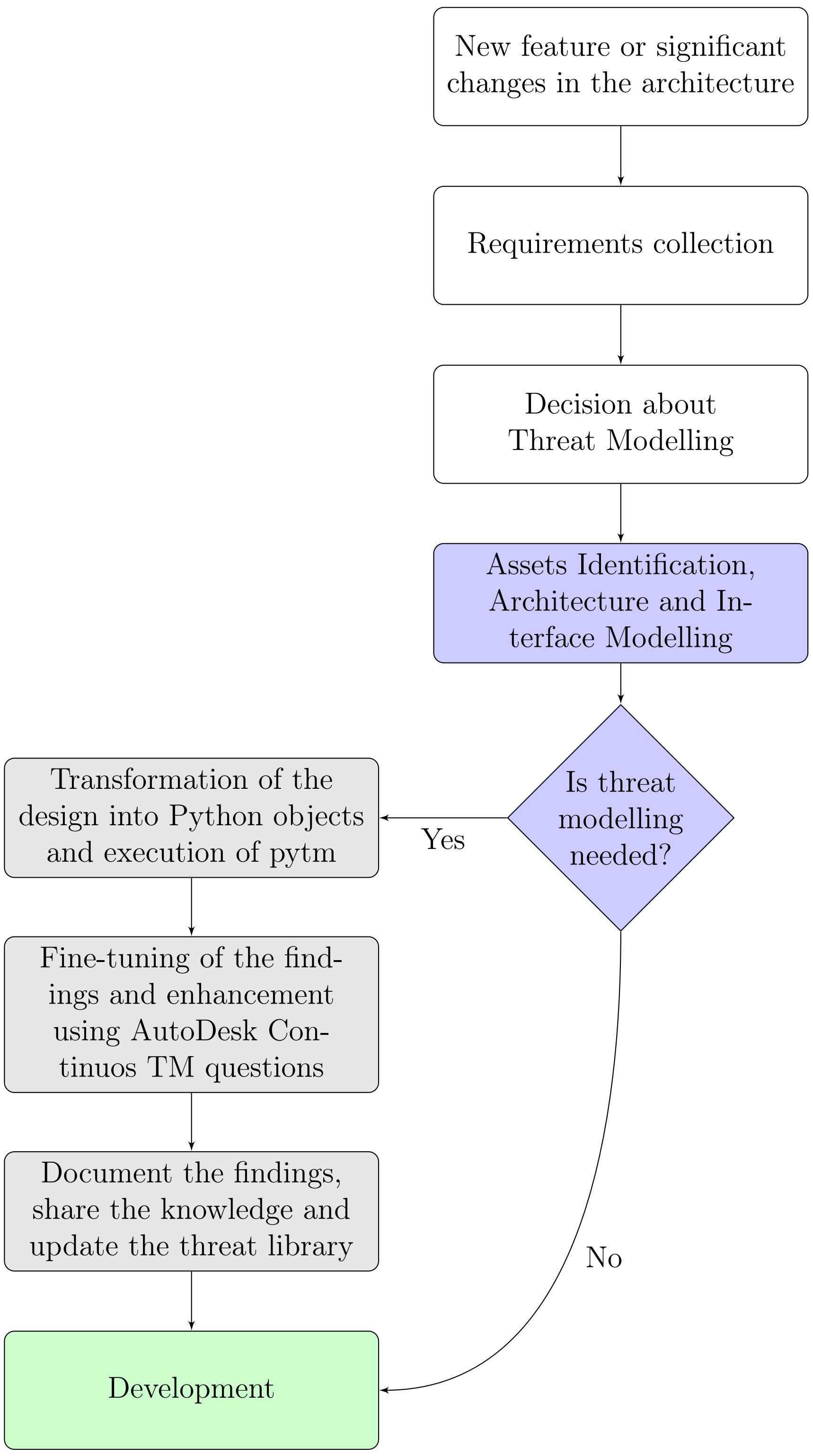

New feature or significant changes in the architecture
Requirements collection
Decision about Threat Modelling
Assets Identification, Architecture and Interface Modelling
Is threat modelling needed?
Yes
Transformation of the design into Python objects and execution of pytm
Fine-tuning of the findings and enhancement using AutoDesk Continuos TM questions
Document the findings, share the knowledge and update the threat library
No
Development

Figure 5.1: Updated threat modelling process and Secure SDLC workflow

## 5.2 Review of verification methods

Once the new process is designed, the natural subsequent phase involves creating a mechanism for assessing its effectiveness in comparison to the previous approach. Without building a way to collect metrics and data about performances and properly evaluate them, a POC in a real environment does not make sense as is not possible to determine the value of the solution. The whole assessment narrative can also be recalled by thinking about the four questions frameworks of Shostack and, in particular, the last one: "Did we do a good job?" [45].

Unfortunately, scientific evaluations of threat modelling approaches and processes are rarely performed and widely used techniques for quality assurance do not exist. This poses a challenge in a metrics-driven environment like DevOps, where the collection of data, even pertaining to threat modelling and its process, is crucial [45]. A general lack of interest and research around the topic was already evident for Shostack in 2008 [48] but the same sensations were still perceived by Bygdas in 2021 [49], [50]. They both posed several possible research questions, but the majority never found an answer. Some inquiries try to discover how various aspects of threat modelling processes contribute to achieving security objectives and meeting quality assurance standards, as well as to what extent they do so. Others focus on questioning which metrics can be defined for discovering the barrier to the adoption of a new process, the area of improvement and the costs of implementation.

Despite standardised methodologies does not exist to perform an evaluation of the threat modelling process, is still possible to find specific and tailor-made ways in the literature.

The systematic literature review performed by Xiong et al. clearly presents an overview of the different techniques used by others. In particular, two types of evaluation are usually employed: theoretical and empirical. Among the first category, it is possible to find simulation analysis and examples conducted in a laboratory environment. For the empirical one, instead, case studies, real implementations, expert reviews and interviews are used [31].

Other sources focus only on evaluating a singular tool/framework and comparing it with others, more than including an analysis of the performances of the whole process. This is the case of the works proposed by Scandariato [51] and Williams [52] that present an evaluation, conducted with computer science students of different techniques, respectively STRIDE and Microsoft Threat Modelling tool, developed by Microsoft.

In contrast, Suhas et al. introduced a comprehensive set of criteria for evaluating threat modelling tools, including aspects such as the identification and comparison of strengths and weaknesses, adoptability costs based on available documentation, and applicability within the given context [53]. The metrics proposed by Shostack are slightly different and incorporate three primary dimensions (cost, expertise prerequisites, and quality of output), although additional ones can also be taken into consideration [25].

## 5.3 Design of the evaluation process

Based on the suggestions and techniques available in the literature and discussed in the previous section, a novel evaluation system was ideated for the proposed threat modelling workflow. Before discussing it and its phases, it is worth noticing that some activities to validate the process itself were already performed, even if no real implementation and deployment exist yet. The first one is discussed in section 5.4 and is a combination of an experiment conducted in a laboratory environment with fictitious data and a use case executed in a production environment with real data. The second one was to evaluate the design of the workflow with some of the stakeholders interviewed, who are familiar with the topic and security in general. The feedback collected was positive and the motivations behind the choices were approved.

Going back to the design of the validation process, the tasks that need to be performed can be grouped by the way data are collected. In fact, one of the driving elements of the evaluation methodology design is information. The two categories were defined: qualitative data collected from the stakeholders and quantitative information collected from tools.

### 5.3.1 Qualitative data

The first set of activities requires testing the knowledge of the stakeholders and collecting information directly from them about awareness.

To begin with, it is strongly recommended to extend the data collected during the interviews and presented in chapter 4. Using a questionnaire with multiple choice questions is sufficient this time as the goal is to collect only information about the currently used process (like an estimated date of the last usage, how much awareness of the process is there, and when the last training was) without the need of getting insights or more broad answer. The questionnaire should be shared with every member of the R&D team and at least a couple of responses from each team should be received. However, collecting only more data about the old process is not sufficient: the same questionnaire should be resubmitted to the stakeholders at least six months or one year after the complete deployment of the new process. In this way, it is possible to discover the diffusion of the process, test the knowledge of the people involved and collect opinions on long-term usage without the bias and the unrealistic usage data of the first period. A comparison should be created to understand if the adoption rate and satisfaction improved or not with the new solution.

Alongside these brief surveys, constant opinions about the sessions, the process, and the strengths and weaknesses of the tools should be collected by the Security

Champions and reported periodically to the Security team, which is in charge of modifying and updating the process when needed.

To conclude a complete POC with a couple of highly collaborative teams willing to provide frequent and honest comments and observations, before a large-scale deployment, should be conducted. During the POC both the new and old processes should be used in order to have the possibility to compare the results.

### 5.3.2 Quantitative metrics

The data-driven approach of DevOps, mentioned before, requires a lot of data that should be collected automatically and over time, whenever it is possible. Those data are also required for implementing the accountability part of the process, defined section 5.1.1. The following Table 5.2 and Table 5.3 present the various metrics that should be collected, divided by category. More specific details are dependent on the development pipeline and specific to all the tools used.

| **Category** | **Metric** |
|---|---|
| Scope and code coverage | Which parts of the code have been modelled? |
| | Are there parts of the codebase that are excluded? |
| | Are the updates of the code and the threat model synchronized? |
| Adoptability | How much time is required for a team to adopt the process? For a new team member? |
| | How helpful is the documentation in both cases? |
| Applicability and maintainability | Is the method able to be applied recursively and account for the relationship among subsystems? |
| | Is the model easy to update and is it possible to keep track of the updates? |
| Consistency of the process | Is the processes the same in all the teams after 6 months? 1 year? |
| | Are the results consistent among the team? Are some teams overdoing and some underdoing? |
| Documentation quality and Knowledge sharing | Is the quality and completeness of the documentation increased? |
| | Are the results easily accessible to everyone? |

Table 5.2: Metrics for quantitative evaluation

| **Category** | **Metric** |
|---|---|
| Effectiveness of threat identification | How many vulnerabilities were reported by the bug bounty and discovered by the pentest compared to before? |
| | How many risks were successfully mitigated in the established period? |
| | Were there security incidents related to threats missed during threat modelling? |
| | How many high, medium, and low-risk threats are identified and addressed over time? |
| | How many security controls have been implemented as a result of threat modelling? |
| Automation percentage | How many threats were discovered manually? How many of them were added to the library? |
| | How many false positives do the tools produce usually? |
| Planning and goals of the sessions | How many times the time booked was enough to discuss both the threats and mitigations |
| | How often were the objectives of the sessions met? |
| Cost-effectiveness | How many resources were invested in the implementation of the new process? |
| | Did the new process have an ROI (e.g. increased number of prospectors)? |

Table 5.3: Metrics for quantitative evaluation

## 5.4 Test implementation and Experiments

As stated in the introduction, the thesis does not include real results about the effectiveness as it was not possible to implement the process and perform a pilot test inside Company Z due to time constraints. Despite that, as mentioned previously, the tools and the workflow were tested in a trial environment, which is described in a later section, using real scenarios of Company Z. This is the reason why this type of evaluation can be seen as midway between an experiment and a use case. The obtained results were also compared to the ones obtained by one of the development teams using the current process. For the sake of brevity, only one example is presented here even if multiple tests were conducted.

### Environment preparation

The tests were performed on a device running a Unix-based operating system and with the latest release of pytm available on GitHub (version 1.2.0). The installation procedure of the tools was smooth using Docker and the documentation was clear enough. The first test with the demo highlighted some minor issues in the standard configuration of the reporting part, but after fixing some paths, everything worked fine.

### Familiarization with the command line interface (CLI)

Running the tool is straightforward due to the presence of a Makefile. By default, when the command "*make*" is executed, all the elements are generated (threats list, diagrams, report) based on the content of *tm.py* file. It is possible to obtain only some of the elements by specifying the correct parameter to the command. It is possible to run the tool even without the use of the Makefile: the commands available are clearly defined and well-documented.

### Understanding the different objects and building the first DFD

Looking at the example first and at the documentation right after can be a good strategy to learn how to transform the architecture of a system into a set of pytm objects quickly. People who are familiar with the concept of OOP (Object Oriented Programming), can get acquainted with the logic in a couple of minutes, even if Python was never used. This is true because no logic should be implemented in the code of the architecture, so the prerequisite knowledge about the language is nonexistent.

The available pool of objects of pytm is limited, which could be constraining if specific environments need to be modelled; nevertheless, there is always the option to expand this collection. At the same time, the scarce number is beneficial for the learning curve, which is definitely steep. To discover the properties available for each object, the documentation is the best way as it provides a good overview of the options[1].

To better understand, the following objects are available: Server, ExternalEntity, DataFlow, Datastore, Actor, Asset, Process, SetOfProcesses, Boundary, Lambda, Data, Threat, Threats and Finding. Regarding the properties, they vary significantly depending on the specific object under consideration, and due to their large number, not all of them can be detailed here. However, some of the most noteworthy properties include *isPII* and *classification* for the Data object, *validatesHeaders* and *implementsCSRFToken* for the Server, *usesStrongSessionIdentifiers* and *encryptsCookies* for Process, and many others.

Among all the use cases chosen for the tests, the one presented here was from the latest threat modelling session performed inside Company Z. The main reason behind the initiative was the development of a new service for retrieving and handling configurations related to certain data, which is referred to as "Q", to adhere to

[1] `https://github.com/izar/pytm/wiki`

confidentiality requirements. For the same reason, some details and specifications of the architecture are obfuscated and intentionally omitted.

The first step of the process was to replicate the DFD of the new feature starting from the one built with OWASP Threat Dragon. The final Python code developed is inserted in Listing 5.1. It's worth mentioning that certain sections were omitted due to their similarities to others, in order to shorten the presented content.

```python
#!/usr/bin/env python3

from pytm import (TM, Actor, Boundary, Classification, Data,
    Dataflow, Datastore, Lambda, Server,)

tm = TM("Functionality Y test")
tm.description = "This is a test of pytm using one of the last
    threat models performed inside Company Z. The service retrieve
    and manage configurations about Q"
tm.isOrdered = True
tm.mergeResponses = True

# Definition of trust boundaries
internet = Boundary("Internet")
server = Boundary("Server Processes")
server.levels = [2]

# Definition of the only actor
user = Actor("Admin")
user.isAdmin = True
user.inBoundary = internet
user.levels = [2]

# Definition of the services
Q_management = Server("Service 1")
Q_management.isHardened = True
Q_management.encodesOutput = True
Q_management.isEncrypted = True
Q_management.protocol = "HTTPS"
Q_management.inBoundary = server

Q_event_handler = Server("Service 2")
# ...
# The same properties set for Service 1 apply also to Service 2

Q_converter = Server("Service 3")
Q_converter.protocol = "Proprietary protocol"
# ...
# The same properties set for Service 1 apply also to Service 3

# Definition of Datastores
no_sql_db = Datastore("NoSQL Database")
no_sql_db.inScope = False
no_sql_db.isHardened = True
no_sql_db.isSQL = False
no_sql_db.maxClassification = Classification.RESTRICTED
no_sql_db.description = "NoSQL database to store Q information"
no_sql_db.levels = [2]
no_sql_db.isEncrypted = True

work_unit_ds = Datastore("Work Unit Data Store")
work_unit_ds.description = "DS to save information about work units
    "
# ... - The same configurations used for the Datastore "NoSQL
    Database" apply also to the Datastore "Work Unit"

# Definition of q data and its flows between the various elements
q = Data(
    name = "Q information",
    classification=Classification.RESTRICTED,
    traverses=[Q_management],
    processedBy=[no_sql_db, Q_event_handler],
    description="Data about Q configurations",
)

user_to_Q_management = Dataflow(user, Q_management, "User create/
    update/delete Q information")
user_to_Q_management.protocol = "HTTPS"
user_to_Q_management.dstPort = 443
user_to_Q_management.isEncrypted = True
user_to_Q_management.data = q
user_to_Q_management.note = "Data on public network"

Q_management_to_user = Dataflow(Q_management, user, "User retrive Q
     information remotely")
Q_management_to_user.note = "Data on public network"
# ... - The same properties of user_to_Q_management are also set
    for Q_management_to_user flow

Q_management_to_no_sql_db = Dataflow(Q_management, no_sql_db, "Save
     Q information to NoSQL DB")
Q_management_to_user.note = "Data on private network"
# ... - The same properties of user_to_Q_management are also set
    for Q_management_to_no_sql_db flow

no_sql_db_to_Q_event_handler = Dataflow(no_sql_db, Q_event_handler,
     "Retrieve Q information from NoSQL DB")
# ... - The same properties of user_to_Q_management are also set
    for no_sql_db_to_Q_event_handler flow

# Definition of work_unit data and its flows between the various
    elements
```

```python
work_unit = Data(
    "Work unit information",
    classification=Classification.RESTRICTED,
    description="Data about work units"
)

Q_event_handler_to_work_unit = Dataflow(Q_event_handler,
   work_unit_ds, "Send work unit information to the data store")
Q_event_handler_to_work_unit.protocol = "Proprietary protocol"
Q_event_handler_to_work_unit.dstPort = 6379
Q_event_handler_to_work_unit.isEncrypted = True
Q_event_handler_to_work_unit.data = work_unit

work_unit_to_Q_converter = Dataflow(work_unit_ds, Q_converter, "
   Send work unit information to Q Converter Service")
work_unit_to_Q_converter.protocol = "Proprietary protocol"
work_unit_to_Q_converter.dstPort = 6379
work_unit_to_Q_converter.isEncrypted = True
work_unit_to_Q_converter.data = work_unit

if __name__ == "__main__":
    tm.process()
```

Listing 5.1: Python code for building the DFD

The final output can be seen in Figure 5.2. As a comparison, the DFD created using Threat Dragon is reported in Figure 5.3: it is evident that both tools can produce overlapping results. As previously stated, pytm is also capable of automatically generating sequence diagrams that may be shared with non-technical people in order to ease comprehension of data flows and where threats may exist and act. The graph generated for the specific use case is depicted in Figure 5.4.

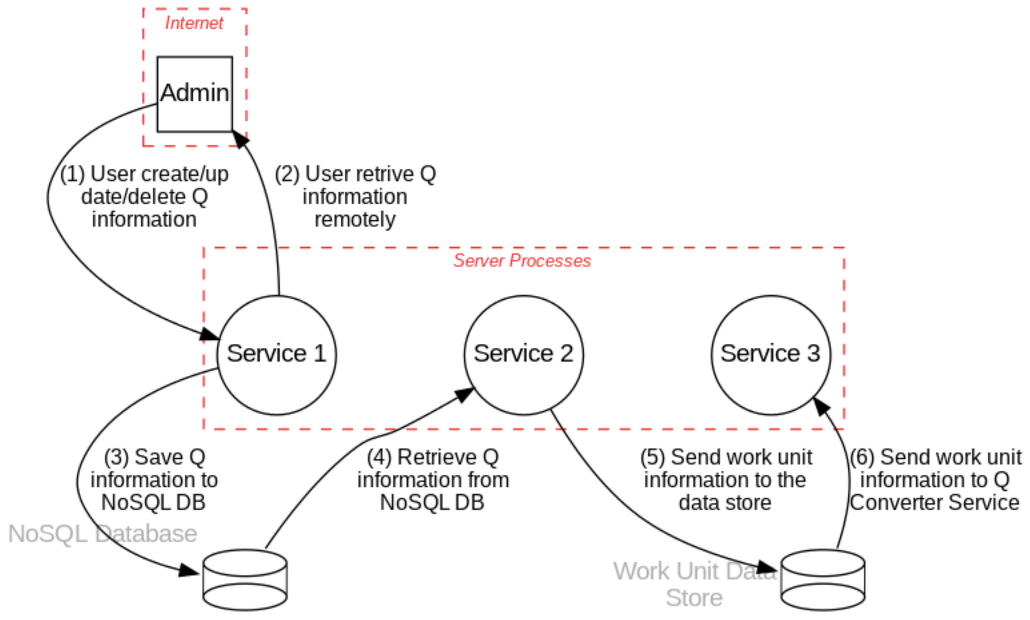

Internet
Admin
(1) User create/up date/delete Q information
(2) User retrive Q information remotely
Server Processes
Service 1
Service 2
Service 3
(3) Save Q information to NoSQL DB
(4) Retrieve Q information from NoSQL DB
(5) Send work unit information to the data store
(6) Send work unit information to Q Converter Service
NoSQL Database
Work Unit Data Store

Figure 5.2: DFD automatically produced by pytm

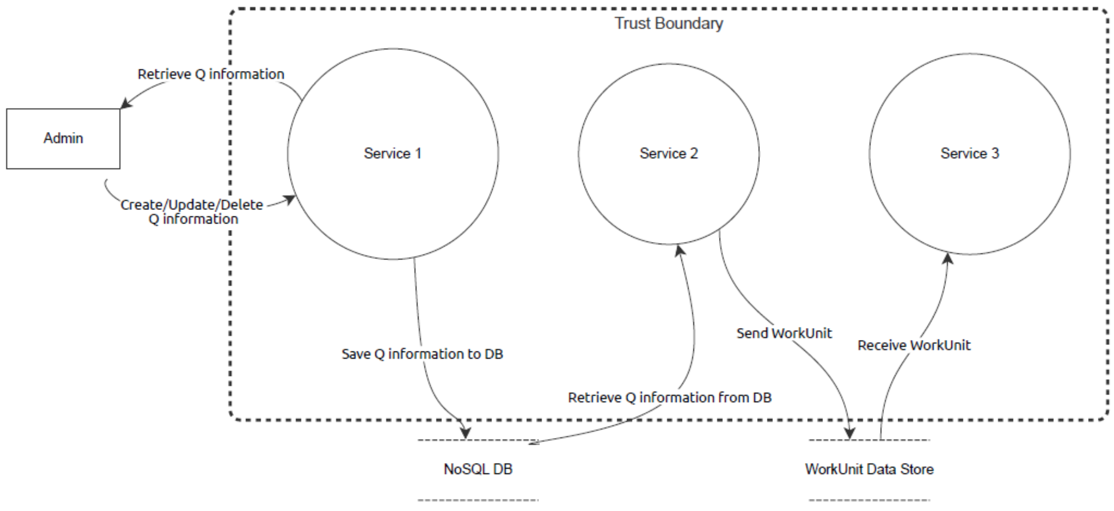


Figure 5.3: DFD manually created with by OWASP Threat Dragon

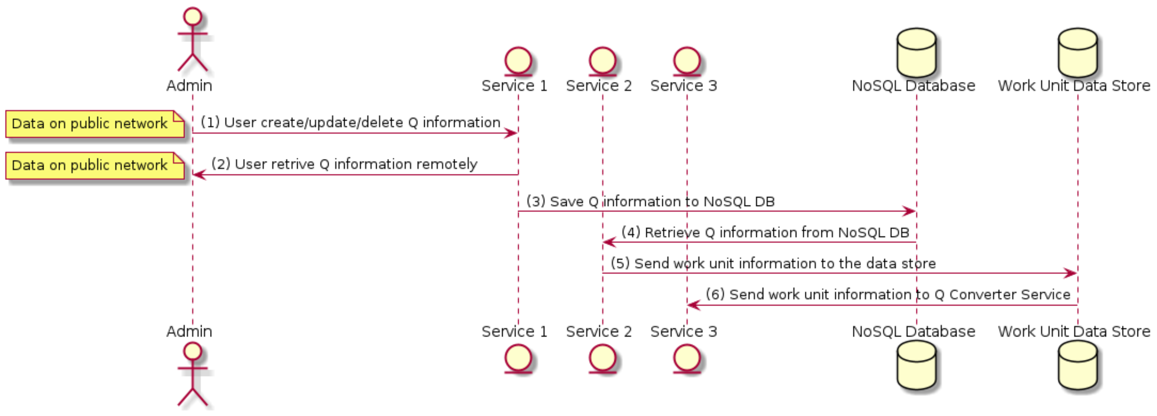

Admin
Service 1
Service 2
Service 3
NoSQL Database
Work Unit Data Store
Data on public network
(1) User create/update/delete Q information
Data on public network
(2) User retrive Q information remotely
(3) Save Q information to NoSQL DB
(4) Retrieve Q information from NoSQL DB
(5) Send work unit information to the data store
(6) Send work unit information to Q Converter Service

Figure 5.4: Sequence diagram automatically generated by pytm

### Results comparison

The number of threats discovered by pytm by simply running a command is surprisingly high. In fact, 155 possible threats were automatically discovered, compared to only two threats discovered after the session conducted with OWASP Threat Dragon. Having such a high number and huge disparity is not good, but different motivations can be the cause:

- Not all of the threats were included in Threat Dragon, as the squad may have previously deliberated on them during the session and excluded the less probable ones, along with those for which mitigations are already in place.
- The knowledge about the part of the system modelled was limited (as based only on the DFD of Threat Dragon) and not all the details were included in the Python objects due to confidentiality reasons. Including a small amount of details makes pattern marching underperforming;
- The three services are described in a similar way using pytm for the reasons above, so most of the threats discovered are repeated for each of them.

It is worth noticing that threats identified manually with Threat Dragon require deciding which elements generate the threats and manually inserting information like

Title, Type (STRIDE), Status (N/A, Open, Mitigated), Score (that can be CVSS), Priority (Low, Medium, High), Description, Mitigations. On the other end, pytm automatically includes details like the targeted element, the severity, an example of instances, the possible mitigations and references to CAPEC and CWE websites.

Both of the threats discovered manually using the old process are detected automatically by the tool. In particular, the Table 5.4 presents a mapping between them.

| **Target** | **Threat of OWASP Threat Dragon** | **Threat of pytm** |
|---|---|---|
| Service 1 | Unauthorized access (Tampering) | Authentication Abuse/Bypass (number 3) |
| Service 3 | Injection attacks (Elevation of privilege) | Format String injection (96), Parameter Injection (98), ... |

Table 5.4: Mapping between the threat detected

The threats defined by pytm related to the injection attack are more specific than the ones created manually as there are multiple ways to disrupt the system: in this case, a one-to-one mapping is difficult to define. The complete reports generated by the tools, which include all the threats and show how the output is presented, are included respectively in Appendix B and Appendix C.

### Refinement of results

In the process proposed, the results will be fine-tuned and enhanced during a team session using the questions proposed in the Countinuos Treat Modelling handbook by AutoDesk. Unfortunately, it was not possible to organize such a test session, but it is still possible to review if all the categories of threat proposed in the "checklist" were considered by the tool.

By iterating over it, the subsequent categories were identified as lacking coverage.

- Data retention, Data minimization and privacy: the different aspects related to data management lifecycle were not parts of by the identified threats;
- Resiliency, Denial of Service, Configuration management, Hardening, Cloud Services, Dev/Stage/Prod practices, Third Party Libraries and Components: most of their questions are more related to deployment and release pipeline than the development per se and can be difficult to integrate them in the description of the architecture. DevOps should assist Software Engineers throughout this part of the discussion as they are more knowledgeable about these topics. Additional automated checks in the pipeline, conducted at a later stage, such as Infrastructure as Code (IaaC) analysis and dependency checks, can be beneficial in corroborating the accurate identification of threats related to these specific areas.

### Considerations

The conducted tests, taking into account both the presented scenario and the omitted ones, demonstrate that automatic threat detection serves as a potent helper, capable of speeding up tasks and supporting the development team. However, in its current state, they still necessitate the inclusion of as many details and a subsequent refinement session to validate the results and identify any false positives. The creation of a threat library specific to the context of Company Z can be a partial solution to enhance the results and reduce the need for manual work.

# 6 Conclusion

The Master's Thesis aimed at proposing a novel approach to threat modelling tailored for adoption by a software development company employing Agile methodologies. As the project was born in an industrial context and was created following the needs of Company Z, it was important not only to consider the best tools, procedures and suggestions to design the new security workflow but also the feasibility of implementation, the implications of the process in a real environment, the costs and the organizational matters.

Achieving the goals was not possible without performing an extensive review of the literature available for threat modelling. Unfortunately, a single and updated source that summarizes all the relevant aspects and approaches of threat modelling is not available and it was first required to get familiar with all the traditional frameworks and methodologies for performing threat modelling and then analyse all the more modern tools, possibilities and approaches that can be employed these days. For this reason, the research conducted and the results reported inchapter 2 can be easily used by others to establish a baseline about the topic. Other aspects that helped reach the goals were the insight and information collected from the stakeholders and the internal documentation available about the current process. Without them, understanding the pain points and limitations would not have been possible.

As a result, a thoroughly revamped threat modelling process has been proposed, designed to automate tasks and streamline efficiency through the use of cutting-edge tools and checklists for expediting sessions. A new way to kick-start the process was also defined and an extensive set of suggestions was created, ranging from guidelines to newly defined requirements related to governance. A minimum verification of the proposed design was performed by presenting the results to the stakeholders and running some examples on real scenarios of Company Z. In both cases, the impressions were good and the results were in line with the expectations, but a more comprehensive and systematic way to evaluate the effectiveness of the process was needed. For this reason, based on the literature research, a way to perform the verification of the process was proposed. It includes qualitative metrics that require the collection of data from the people involved in the process, and quantitative metrics that use evidence and data collected objectively from the development pipeline.

Even though the solution designed was tailored to the needs of Company Z, it is certainly feasible to adapt the entire process to other medium-sized companies employing agile methodologies. In fact, the foundations of the process lie on requirements that are aligned with the ones of industry operating in the sector of software development, the tools we suggest are compatible across multiple platforms and do not necessitate licences, and they can be seamlessly integrated into the development stack. Moreover, the majority of the suggestions proposed are self-contained, allowing a broader applicability even of sub-parts of the process.

## 6.1 Future work

As already mentioned in the introduction, the natural evolution of the work includes the implementation of the designed processes inside Company Z. Although the value of this work is clear, and sometimes the importance of the ideation of processes is undervalued, a technique like threat modelling needs to be tested in a real environment, and specifically in the context for which it has been created, to determine its effectiveness. The implementation of the process should start with the definition of the specific tools to be used. To avoid overspecialization on the use case of Company Z, the name of specific solutions for the ticketing system, knowledge sharing, progress and goal tracking was not mentioned. After that, a pilot test should be conducted with at least two different development teams (to have enough results to compare and avoid biases). An implementation plan, for completing a large-scale deployment strategy, should be created at that point. Alongside all these activities, the verification system should be implemented and the data collected and analysed right from the POC. Once the process is well established at Company Z, enough data are collected and the process is proven effective, it can be extended to other companies and contexts.

# References

[1] M. Kapko. “Cybersecurity spending on pace to surpass $260b by 2026”. (Oct. 18, 2022), [Online]. Available: https://www.cybersecuritydive.com/news/security-spending-balloons/634365/.

[2] A. Petrosyan. “Number of data breaches and victims u.s. 2022”, Statista. (Apr. 1, 2023), [Online]. Available: https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/.

[3] M. Cukier. “Study: Hackers attack every 39 seconds”. (Feb. 9, 2007), [Online]. Available: https://eng.umd.edu/news/story/study-hackers-attack-every-39-seconds.

[4] J. Bird, E. Johnson, and F. Kim, “2015 state of application security: Closing the GapSANS”, *Closing the Gap*, May 2015. [Online]. Available: https://scadahacker.com/library/Documents/White_Papers/SANS%20-%20State%20Application%20Security%20-%202015.pdf.

[5] S. Licata, *Tactical Threat Modeling.* SAFECode, May 8, 2019. [Online]. Available: https://safecode.org/resource-secure-development-practices/tactical-threat-modeling/.

[6] A. Shostack, Z. Braiterman, J. Marcil, *et al.* “Threat modeling manifesto”, Threat Modeling Manifesto. (2023), [Online]. Available: https://github.

com/Threat-Modeling-Manifesto/threat-modeling-manifesto/releases/latest/download/threat-modeling-manifesto.pdf.

[7] Snyk. "5 benefits of shift left security", Snyk. (2023), [Online]. Available: https://snyk.io/learn/shift-left-security/.

[8] N. Mead, F. Shull, K. Vemuru, and O. Villadsen. "A hybrid threat modeling method", Software Engineering Institute - Carnegie Mellon University. (Mar. 2018), [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=516617.

[9] P. DevSecOps, *Agile Threat Modeling in 5 Simple Steps*. Hysn Technologies Inc, 2023. [Online]. Available: https://www.practical-devsecops.com/wp-content/uploads/2023/03/Agile-Threat-Modeling-ebook.pdf.

[10] A. Miller, "User story threat modeling: It's the DevSecOps way", SnykCon 2020, 2020. [Online]. Available: https://snyk.io/learn/snykcon-user-story-threat-modeling-its-the-devsecops-way/.

[11] R. Reichel. "What is threat modeling and GitHub's process - GitHub blog", The GitHub Blog. (Sep. 2, 2020), [Online]. Available: https://github.blog/2020-09-02-how-we-threat-model/.

[12] L. Conklin, S. Strittmatter, and V. Drake. "Threat modeling process", OWASP Foundatio. (Aug. 16, 2021), [Online]. Available: https://owasp.org/www-community/Threat_Modeling_Process.

[13] A. Shostack. "The jenga view of threat modeling - supporting delivery of resilient software", Shostack + Associates. (Jun. 1, 2020), [Online]. Available: https://shostack.org/files/papers/The_Jenga_View_of_Threat_Modeling.pdf.

[14] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction.* OUP USA, 1977, 1216 pp., Google-Books-ID: hwAHmktpk5IC, ISBN: 978-0-19-501919-3.

[15] E. G. Amoroso, *Fundamentals of Computer Security Technology.* PTR Prentice Hall, 1994, 440 pp., Google-Books-ID: f95QAAAAMAAJ, ISBN: 978-0-13-108929-7.

[16] C. Slater, O. Saydjari, B. Schneier, and J. Wallner, "Toward a secure system engineering methodolgy.", Jan. 1998, pp. 2–10. DOI: 10.1145/310889.310900.

[17] L. Kohnfelde and P. Garg. "The threats to our products", Microsoft Interface. (Apr. 1, 1999), [Online]. Available: https://shostack.org/files/microsoft/The-Threats-To-Our-Products.docx.

[18] C. Alberts, A. Dorofee, J. Stevens, and C. Woody. "Introduction to the OCTAVE approach", Software Engineering Institute - Carnegie Mellon University. (Aug. 2003), [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=51546.

[19] threatmodeler. "The evolution of threat modeling - from manual to enterprise strategic", ThreatModeler. (Oct. 14, 2016), [Online]. Available: https://threatmodeler.com/evolution-of-threat-modeling/.

[20] A. Shostack. "NIST brings threat modeling into the spotlight", Dark Reading. Section: threat-intelligence. (Sep. 23, 2021), [Online]. Available: https://www.darkreading.com/threat-intelligence/nist-brings-threat-modeling-into-the-spotlight.

[21] A. Shostack. "The ultimate beginner's guide to threat modeling", Shostack + Associates. (2023), [Online]. Available: https://shostack.org/resources/threat-modeling.html.

[22] S. Myagmar, A. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements", Aug. 1, 2005.

[23] J. Becker, "A pragmatic approach to threat modeling", OWASP Frankfurt Chapter Meetup 2022, Frankfurt, Jun. 25, 2022. [Online]. Available: `https://owasp.org/www-chapter-frankfurt/assets/slides/55_OWASP_Frankfurt_Stammtisch_2.pdf`.

[24] N. Kirtley. "What is threat modeling", Threat-Modeling.com. Section: Threat Modeling. (Aug. 1, 2022), [Online]. Available: `https://threat-modeling.com/what-is-threat-modeling/`.

[25] A. Shostack. "Fast, cheap and good whitepaper - an unusual trade-off available in threat modeling", Shostack + Associates. (Dec. 15, 2021), [Online]. Available: `https://shostack.org/blog/fast-cheap-good/`.

[26] AWS. "Threat modeling for builders", AWS Workshshops. (2023), [Online]. Available: `https://catalog.workshops.aws/threatmodel/en-US`.

[27] A. Shostack, *Threat modeling: Designing for security.* John Wiley & Sons, 2014.

[28] V. Drake. "Threat modeling | OWASP foundation". (Aug. 16, 2021), [Online]. Available: `https://owasp.org/www-community/Threat_Modeling`.

[29] D. Boyd. "How to approach threat modeling", AWS Security Blog. Section: AWS Well-Architected. (Jan. 11, 2021), [Online]. Available: `https://aws.amazon.com/blogs/security/how-to-approach-threat-modeling/`.

[30] C. Dictionary. "Framework Definition", Cambridge Dictionary. (Aug. 9, 2023), [Online]. Available: `https://dictionary.cambridge.org/it/dizionario/inglese/framework`.

[31] W. Xiong and R. Lagerström, "Threat modeling – a systematic literature review", *Computers & Security*, vol. 84, pp. 53–69, Jul. 1, 2019, ISSN: 0167-4048. DOI: 10.1016/j.cose.2019.03.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818307478.

[32] N. Shevchenko. "Threat modeling: 12 available methods", Software Engineering Institute - Carnegie Mellon University. (Dec. 3, 2018), [Online]. Available: https://insights.sei.cmu.edu/blog/threat-modeling-12-available-methods/.

[33] N. Humphrey. "What is a threat library?", ThreatQuotient. (Dec. 12, 2017), [Online]. Available: https://www.threatq.com/threat-library/.

[34] M. Sass. "A threat modeling field guide". (Jul. 30, 2022), [Online]. Available: https://shellsharks.com/threat-modeling.

[35] M. Milner, "A new approach to cloud security for developers & security teams-trend micro", AWS Summit 2023, Amsterdam, Jun. 1, 2023.

[36] I. Tarandach and M. J. Coles, *Threat modeling: a practical guide for development teams*, First edition. Sebastapol, CA: O'Reilly Media, 2021, 1 p., OCLC: 1220993828, ISBN: 978-1-4920-5652-2. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2680271.

[37] S. Curzi, A. Nevico, J. Devis, R. P. Rodriguez, and B. Hanson. "Integrating threat modeling with DevOps -microsoft security engineering documentation", Microsoft Learn. (Dec. 6, 2022), [Online]. Available: https://learn.microsoft.com/en-us/security/engineering/threat-modeling-with-dev-ops.

[38] Autodesk. "Autodesk continuous threat modeling handbook", GitHub. (May 31, 2019), [Online]. Available: https://github.com/Autodesk/continuous-

threat-modeling/blob/master/Continuous_Threat_Modeling_Handbook.md.

[39] I. Tarandach, "(continuous) threat modeling: What works?", O'Reilly Software Architecture Conference, Feb. 5, 2019. [Online]. Available: https://conferences.oreilly.com/software-architecture/sa-ny-2019/public/schedule/detail/71585.html.

[40] S. Krishnan, "Threat modeling: A hybrid approach | SANS institute", Mar. 9, 2017. [Online]. Available: https://www.sans.org/blog/threat-modeling-hybrid-approach/.

[41] M. Souppaya and K. Scarfone, "Guide to data-centric system threat modeling", National Institute of Standards and Technology, NIST Special Publication (SP) 800-154 (Draft), Mar. 14, 2016. [Online]. Available: https://csrc.nist.gov/Pubs/sp/800/154/IPD.

[42] G. Hill, *Rapid threat model prototyping (RTMP) documents*, original-date: 2018-09-08T07:49:59Z, Jul. 19, 2023. [Online]. Available: https://github.com/geoffrey-hill-tutamantic/rapid-threat-model-prototyping-docs.

[43] R. Tame. "Rapid threat model prototyping", Blue Hat Security. (Jan. 14, 2019), [Online]. Available: https://www.bluehatsecurity.net/blog/rapid-threat-model-prototyping/.

[44] M. G. Jaatun, K. Bernsmed, D. S. Cruzes, and I. A. Tøndel, *Exploring Security in Software Architecture and Design: Threat Modeling in Agile Software Development (Chapter 1)* (Advances in Information Security, Privacy, and Ethics), M. Felderer and R. Scandariato, Eds., red. by M. Gupta. IGI Global, 2019. DOI: 10.4018/978-1-5225-6313-6. [Online]. Available: http://services.igi-

global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-5225-6313-6.

[45] K. Yskout, T. Heyman, D. Van Landuyt, L. Sion, K. Wuyts, and W. Joosen, "Threat modeling: From infancy to maturity", in *2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, Oct. 2020, pp. 9–12.

[46] J. Nielsen. "Why you only need to test with 5 users", Nielsen Norman Group. (Mar. 18, 2000), [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

[47] M. Brulotte. "What is RASCI / RACI", Interfacing Technologies Corporation. (Aug. 28, 2021), [Online]. Available: https://www.interfacing.com/what-is-rasci-raci.

[48] A. Shostack, "Experiences threat modeling at microsoft", Jan. 1, 2008.

[49] E. Bygdås, L. A. Jaatun, S. B. Antonsen, A. Ringen, and E. Eiring, "Evaluating threat modeling tools: Microsoft TMT versus OWASP threat dragon", in *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, Jun. 2021, pp. 1–7. DOI: 10.1109/CyberSA52016.2021.9478215.

[50] C-MRiC ORG. "Evaluating threat modeling tools: Microsoft TMT versus OWASP threat dragon". in collab. with L. Jaatun. (Jun. 18, 2021), [Online]. Available: https://www.youtube.com/watch?v=SHWwwhVHtug.

[51] R. Scandariato, K. Wuyts, and W. Joosen, "A descriptive study of microsoft's threat modeling technique", *Requirements Engineering*, vol. 20, no. 2, pp. 163–180, Jun. 1, 2015, ISSN: 1432-010X. DOI: 10.1007/s00766-013-0195-2. [Online]. Available: https://doi.org/10.1007/s00766-013-0195-2.

[52] I. Williams and X. Yuan, “Evaluating the effectiveness of microsoft threat modeling tool”, in *Proceedings of the 2015 Information Security Curriculum Development Conference*, ser. InfoSec ’15, New York, NY, USA: Association for Computing Machinery, Oct. 10, 2015, pp. 1–6, ISBN: 978-1-4503-4049-6. DOI: 10.1145/2885990.2885999. [Online]. Available: https://dl.acm.org/doi/10.1145/2885990.2885999.

[53] K. S. Suhas and N. Shah, “Evaluation of threat models”, *International Journal for Research in Applied Science and Engineering Technology*, vol. 11, no. 3, p. 809, Mar. 1, 2023, ISSN: 2321-9653. [Online]. Available: https://www.academia.edu/99253732/Evaluation_of_Threat_Models.

# Appendix A Templates for the semi-structured interviews

## A.1 Template for Software Engineers

1. Personal presentation and context introduction
2. SDLC, threat modelling, and deployment process
   - Can you briefly describe how the SDLC works from the creation of the ticket to the deployment in the production environment? Consider each phase, also development and testing.
   - How familiar are you with threat modelling? And with the current threat modelling process defined within Company Z?
   - Did you ever have to do it? When was the last time? How often do you have to do it?
3. Experience with threat modelling
   - Who is typically involved in a threat modelling session? Who is leading the session? Is someone from the Information Security team always invited?
   - Was it easy to organize the session? Have you encountered any difficulties in coordinating and collaborating with other teams?

- Did you follow the actual defined process? If not, which methodologies or frameworks did you use for threat modelling? Are there knowledge gaps regarding threat modelling inside Company Z?
- Were the results of the threat modelling session documented and communicated? Did you measure the success or effectiveness of the threat modelling session in any way?
- How well does the current process integrate with the development life cycle? Do threat modelling activities impact project timelines or delivery schedules?
- Who do you think is responsible for making threat modelling happen? Product owners or developers?

4. Challenges and suggestions
    - What are the strengths of the current threat modelling process? And the biggest pain point? Are there any challenges or limitations?
    - Do you see any areas of improvement regarding the current threat modelling process? What changes would you make to the process?
    - Did you perform threat modelling in other ways during previous experiences?
    - From your perspective and past experiences, how do you think that thread modelling within Company Z should be?

## A.2 Template for internal Ethical Hacker

1. SDLC, threat modelling, and deployment process
    - How did you design the current process and choose the current tools (e.g. OWASP Threat Dragon)?

- Can you briefly describe how the threat modelling process is integrated into the current SDLC? I saw that in the documentation it mentioned the existence of two fields in the ticketing system, are they used?
- Apart from threat modelling, are there any other tools integrated into the pipeline?

2. Experience with threat modelling

- When was the last time you were involved in a threat modelling session? How often are you involved in that? Did you do threat modelling in other ways?
- Do you feel like there are knowledge gaps regarding threat modelling inside Company Z? Do you think that threat modelling is enough taken into consideration during development?
- Are the results of the threat modelling session documented somewhere and communicated?
- How well do you think the threat modelling process fits the current development life cycle?

3. Challenges and suggestions

- What are the strengths of the current threat modelling process? And the biggest pain point? Are there any challenges or limitations?
- Do you see any areas of improvement regarding the current threat modelling process? What changes would you make to the process?
- Did you perform threat modelling in other ways during previous experiences?

## A.3 Template for Product Owner

1. Personal presentation and context introduction

2. SDLC, threat modelling, and deployment process

   - What is the role of the product owner inside the SDLC?
   - Does your team use threat modelling?
   - Are you aware of the current process? Did someone introduce the process to you when you joined?
   - How involved are you in the threat modelling activities for the product you own? Do threat modelling activities impact project timelines or delivery schedules?
   - As a product owner, how do you currently perceive the role of threat modelling in the software development life cycle?
   - Are there any specific metrics or indicators you use/would like to use to assess the effectiveness of the threat modelling process?

3. Security requirements/features balance

   - Do you make an attempt to create a security mentality/requirements for the people developing the product you own?
   - How do you prioritize security concerns alongside other product requirements and features? How are you balancing security requirements with product functionality or delivery timelines? Are there any specific security objectives or compliance requirements that you consider during the product road-map planning?
   - How do you communicate and collaborate with the development team to ensure that security considerations are adequately addressed?

- Are there any challenges or concerns you have observed in integrating security considerations into the product's user stories and acceptance criteria?

4. Governance

   - Who do you think is responsible for making threat modelling happen? Product owner or developers?

5. Suggestions

   - Did you perform threat modelling in other ways during previous experiences?

# Appendix B OWASP Threat Dragon Report

# Q configurations

**Owner**: Squad1
**Reviewer**: InfoSec1
**Contributors**:
**Date Generated**: Wed Aug 30 2023

# Executive Summary

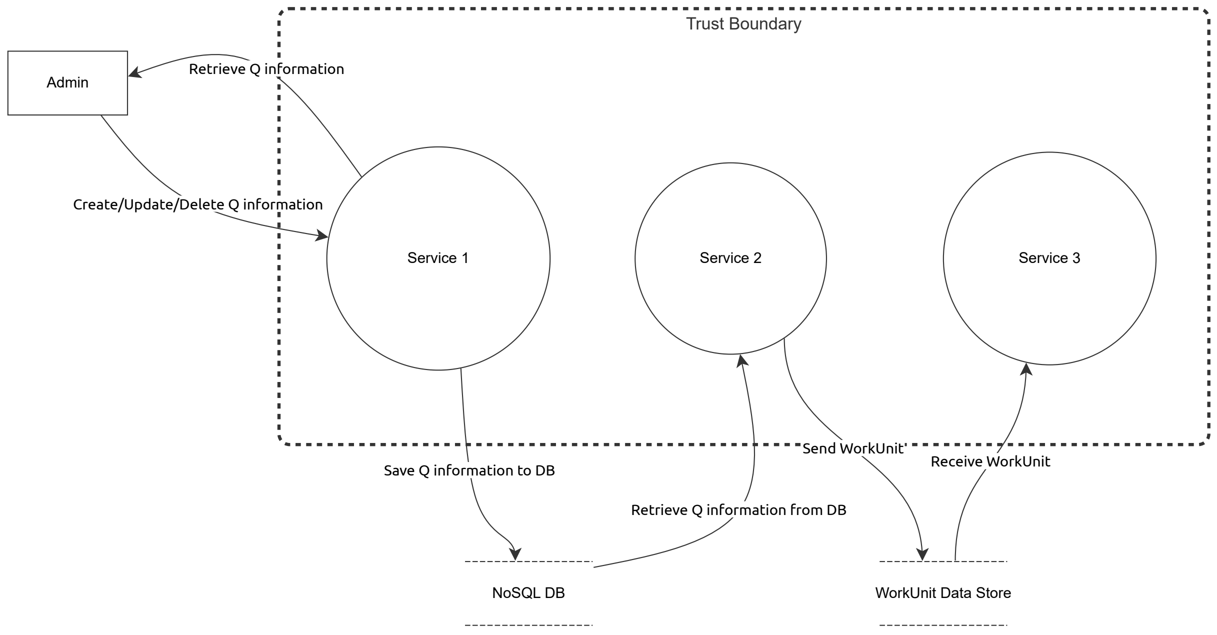
## High level system description

The service retrieve and manage configurations about Q.

## Summary

| | |
|---|---|
| **Total Threats** | 2 |
| **Total Mitigated** | 2 |
| **Not Mitigated** | 0 |
| **Open / High Priority** | 0 |
| **Open / Medium Priority** | 0 |
| **Open / Low Priority** | 0 |
| **Open / Unknown Priority** | 0 |

## Threats


Trust Boundary
Admin
Retrieve Q information
Create/Update/Delete Q information
Service 1
Service 2
Service 3
Save Q information to DB
Retrieve Q information from DB
Send WorkUnit'
Receive WorkUnit
NoSQL DB
WorkUnit Data Store

# Threats

## Service 1 (Process)

| Number | Title | Type | Priority | Status | Score | Description | Mitigations |
|---|---|---|---|---|---|---|---|
| 1 | Unauthorized access | Tampering | High | Mitigated |  | Some actor who does not have access changes the values of Q | All communication requires authentication and all create/update/delete requires superadmin authorization |

## Service 3 (Process)

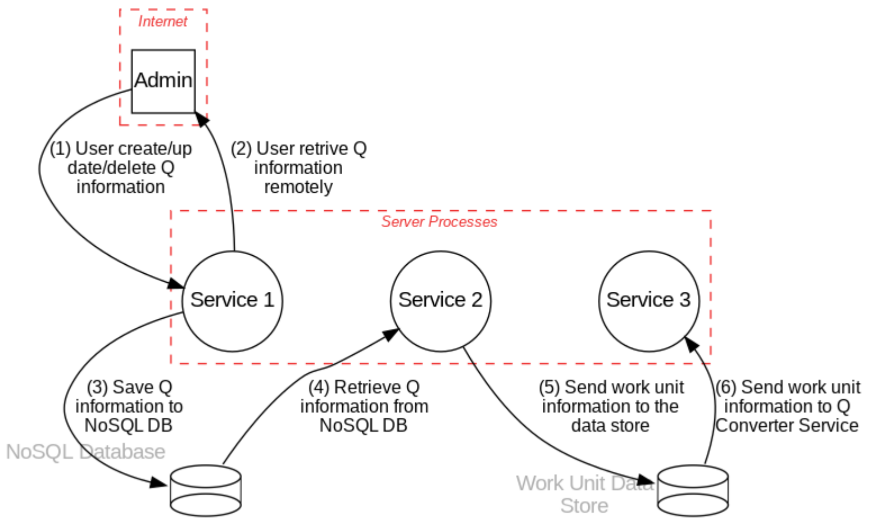
| Number | Title | Type | Priority | Status | Score | Description | Mitigations |
|---|---|---|---|---|---|---|---|
| 2 | Injection attacks | Elevation of privilege | High | Mitigated |  | Some input from the Q configurations allows you to inject something into the command used to perform the task of Service 3. | Validation of our data, both at the level of presets and at the command itself. |

# Appendix C pytm Report

## System Description

This is a test of pytm using one of the last threat models performed inside Company Z. The service retrieve and manage configurations about Q

## Dataflow Diagram - Level 0 DFD


Internet
Admin
(1) User create/update/delete Q information
(2) User retrive Q information remotely
Server Processes
Service 1
Service 2
Service 3
(3) Save Q information to NoSQL DB
(4) Retrieve Q information from NoSQL DB
(5) Send work unit information to the data store
(6) Send work unit information to Q Converter Service
NoSQL Database
Work Unit Data Store

## Dataflows

| Name | From | To | Data | Protocol | Port |
|---|---|---|---|---|---|
| User create/update/delete Q information | Admin | Service 1 | Q information | HTTPS | 443 |
| User retrive Q information remotely | Service 1 | Admin | Q information | HTTPS | 443 |
| Save Q information to NoSQL DB | Service 1 | NoSQL Database | Q information | HTTPS | 443 |
| Retrieve Q information from NoSQL DB | NoSQL Database | Service 2 | Q information | HTTPS | 443 |
| Send work unit information to the data store | Service 2 | Work Unit Data Store | Work unit information | Proprietary protocol | 6379 |
| Send work unit information to Q Converter Service | Work Unit Data Store | Service 3 | Work unit information | Proprietary protocol | 6379 |

## Data Dictionary

| Name | Description | Classification |
|---|---|---|
| Q information | Data about Q configurations | RESTRICTED |
| Work unit information | Data about work units | RESTRICTED |

## Potential Threats

▶ 1 – Server Side Include (SSI) Injection
▶ 2 – Command Line Execution through SQL Injection
▼ 3 – Authentication Abuse/ByPass

**Targeted Element**

Service 1

**Severity**

Medium

**Example Instances**

An adversary that has previously obtained unauthorized access to certain device resources, uses that access to obtain information such as location and network information.

**Mitigations**

Use strong authentication and authorization mechanisms. A proven protocol is OAuth 2.0, which enables a third-party application to obtain limited access to an API.

**References**

https://capec.mitre.org/data/definitions/114.html, http://cwe.mitre.org/data/definitions/287.html

- ▶ 4 – Excavation
- ▶ 5 – Double Encoding
- ▶ 6 – Privilege Abuse
- ▶ 7 – Flooding
- ▶ 8 – Path Traversal
- ▶ 9 – Excessive Allocation
- ▶ 10 – Format String Injection
- ▶ 11 – LDAP Injection
- ▶ 12 – Parameter Injection
- ▶ 13 – Relative Path Traversal
- ▶ 14 – Input Data Manipulation
- ▶ 15 – Dictionary-based Password Attack
- ▶ 16 – Using Malicious Files
- ▶ 17 – Web Application Fingerprinting
- ▶ 18 – XSS Targeting Non-Script Elements
- ▶ 19 – Exploiting Incorrectly Configured Access Control Security Levels
- ▶ 20 – Embedding Scripts within Scripts
- ▶ 21 – PHP Remote File Inclusion
- ▶ 22 – Principal Spoof
- ▶ 23 – XSS Targeting Error Pages
- ▶ 24 – XSS Using Alternate Syntax
- ▶ 25 – Encryption Brute Forcing
- ▶ 26 – Manipulate Registry Information
- ▶ 27 – Removing Important Client Functionality
- ▶ 28 – XSS Using MIME Type Mismatch
- ▶ 29 – Exploitation of Trusted Credentials
- ▶ 30 – Functionality Misuse
- ▶ 31 – Fuzzing and observing application log data/errors for application mapping
- ▶ 32 – Exploiting Trust in Client
- ▶ 33 – XML External Entities Blowup
- ▶ 34 – Session Credential Falsification through Manipulation
- ▶ 35 – DTD Injection
- ▶ 36 – XML Attribute Blowup
- ▶ 37 – XSS Targeting URI Placeholders
- ▶ 38 – XSS Using Doubled Characters
- ▶ 39 – SOAP Array Overflow
- ▶ 40 – HTTP Response Smuggling
- ▶ 41 – HTTP Request Smuggling
- ▶ 42 – Session Credential Falsification through Prediction
- ▶ 43 – Session Hijacking - ServerSide
- ▶ 44 – Server Side Include (SSI) Injection
- ▶ 45 – Command Line Execution through SQL Injection
- ▶ 46 – Authentication Abuse/ByPass
- ▶ 47 – Excavation
- ▶ 48 – Double Encoding
- ▶ 49 – Privilege Abuse
- ▶ 50 – Flooding
- ▶ 51 – Path Traversal
- ▶ 52 – Excessive Allocation
- ▶ 53 – Format String Injection
- ▶ 54 – LDAP Injection
- ▶ 55 – Parameter Injection
- ▶ 56 – Relative Path Traversal
- ▶ 57 – Input Data Manipulation
- ▶ 58 – Dictionary-based Password Attack
- ▶ 59 – Using Malicious Files
- ▶ 60 – Web Application Fingerprinting
- ▶ 61 – XSS Targeting Non-Script Elements
- ▶ 62 – Exploiting Incorrectly Configured Access Control Security Levels
- ▶ 63 – Embedding Scripts within Scripts
- ▶ 64 – PHP Remote File Inclusion
- ▶ 65 – Principal Spoof
- ▶ 66 – XSS Targeting Error Pages
- ▶ 67 – XSS Using Alternate Syntax
- ▶ 68 – Encryption Brute Forcing
- ▶ 69 – Manipulate Registry Information
- ▶ 70 – Removing Important Client Functionality
- ▶ 71 – XSS Using MIME Type Mismatch
- ▶ 72 – Exploitation of Trusted Credentials
- ▶ 73 – Functionality Misuse
- ▶ 74 – Fuzzing and observing application log data/errors for application mapping
- ▶ 75 – Exploiting Trust in Client
- ▶ 76 – XML External Entities Blowup
- ▶ 77 – Session Credential Falsification through Manipulation
- ▶ 78 – DTD Injection
- ▶ 79 – XML Attribute Blowup
- ▶ 80 – XSS Targeting URI Placeholders
- ▶ 81 – XSS Using Doubled Characters
- ▶ 82 – SOAP Array Overflow
- ▶ 83 – HTTP Response Smuggling
- ▶ 84 – HTTP Request Smuggling
- ▶ 85 – Session Credential Falsification through Prediction
- ▶ 86 – Session Hijacking - ServerSide

▶ 87 – Server Side Include (SSI) Injection
▶ 88 – Command Line Execution through SQL Injection
▶ 89 – Authentication Abuse/ByPass
▶ 90 – Excavation
▶ 91 – Double Encoding
▶ 92 – Privilege Abuse
▶ 93 – Flooding
▶ 94 – Path Traversal
▶ 95 – Excessive Allocation
▼ 96 – Format String Injection

**Targeted Element**

Service 3

**Severity**

High

**Example Instances**

Untrusted search path vulnerability in the add_filename_to_string function in intl/gettext/loadmsgcat.c for Elinks 0.11.1 allows local users to cause Elinks to use an untrusted gettext message catalog (.po file) in a ../po directory, which can be leveraged to conduct format string attacks.

**Mitigations**

Limit the usage of formatting string functions. Strong input validation - All user-controllable input must be validated and filtered for illegal formatting characters.

**References**

https://capec.mitre.org/data/definitions/135.html, http://cwe.mitre.org/data/definitions/134.html, http://cwe.mitre.org/data/definitions/133.html

▶ 97 – LDAP Injection
▼ 98 – Parameter Injection

**Targeted Element**

Service 3

**Severity**

Medium

**Example Instances**

The target application accepts a string as user input, fails to sanitize characters that have a special meaning in the parameter encoding, and inserts the user-supplied string in an encoding which is then processed.

**Mitigations**

Implement an audit log written to a separate host. In the event of a compromise, the audit log may be able to provide evidence and details of the compromise. Treat all user input as untrusted data that must be validated before use.

**References**

https://capec.mitre.org/data/definitions/137.html, http://cwe.mitre.org/data/definitions/88.html

▶ 99 – Relative Path Traversal
▶ 100 – Input Data Manipulation
▶ 101 – Dictionary-based Password Attack
▶ 102 – Using Malicious Files
▶ 103 – Web Application Fingerprinting
▶ 104 – XSS Targeting Non-Script Elements
▶ 105 – Exploiting Incorrectly Configured Access Control Security Levels
▶ 106 – Embedding Scripts within Scripts
▶ 107 – PHP Remote File Inclusion
▶ 108 – Principal Spoof
▶ 109 – XSS Targeting Error Pages
▶ 110 – XSS Using Alternate Syntax
▶ 111 – Encryption Brute Forcing
▶ 112 – Manipulate Registry Information
▶ 113 – Removing Important Client Functionality
▶ 114 – XSS Using MIME Type Mismatch
▶ 115 – Exploitation of Trusted Credentials
▶ 116 – Functionality Misuse
▶ 117 – Fuzzing and observing application log data/errors for application mapping
▶ 118 – Exploiting Trust in Client
▶ 119 – XML External Entities Blowup
▶ 120 – Session Credential Falsification through Manipulation
▶ 121 – DTD Injection
▶ 122 – XML Attribute Blowup
▶ 123 – XSS Targeting URI Placeholders
▶ 124 – XSS Using Doubled Characters
▶ 125 – SOAP Array Overflow
▶ 126 – HTTP Response Smuggling
▶ 127 – HTTP Request Smuggling
▶ 128 – Session Credential Falsification through Prediction
▶ 129 – Session Hijacking - ServerSide
▶ 130 – Interception
▶ 131 – Content Spoofing
▶ 132 – Sniffing Attacks

- ▶ 133 – Communication Channel Manipulation
- ▶ 134 – Data Leak
- ▶ 135 – Interception
- ▶ 136 – Content Spoofing
- ▶ 137 – Sniffing Attacks
- ▶ 138 – Communication Channel Manipulation
- ▶ 139 – Data Leak
- ▶ 140 – Interception
- ▶ 141 – Content Spoofing
- ▶ 142 – Sniffing Attacks
- ▶ 143 – Communication Channel Manipulation
- ▶ 144 – Data Leak
- ▶ 145 – Sniffing Attacks
- ▶ 146 – Communication Channel Manipulation
- ▶ 147 – Data Leak
- ▶ 148 – Interception
- ▶ 149 – Content Spoofing
- ▶ 150 – Sniffing Attacks
- ▶ 151 – Communication Channel Manipulation
- ▶ 152 – Data Leak
- ▶ 153 – Sniffing Attacks
- ▶ 154 – Communication Channel Manipulation
- ▶ 155 – Data Leak