Abstract

Threat Management in Agile Organisations

Oscar Sandén

A threat analysis of a computer system identifies and analyses threats to the systems and its assets. The process of handling the identified threats, verify the mitigations and to continuously discover new threats during agile development is difficult.

By making use of the backlog to track threats and security-related tasks a transparent connection between the threats and their security controls is established. In combination with other tools, a method of integrating the threat analysis into an agile development method is created.

The method proposed in this thesis is a solution to the problem of integrating a threat analysis into a agile organisation and presents tools that can aid in a continuous threat-driven security work.
## Contents

1 **Introduction**  

2 **Background**  
   2.1 Agile software development  
      2.1.1 Security in agile  
   2.2 Scaled Agile Framework  
      2.2.1 The SAFe backlogs  
      2.2.2 Kanban boards  
      2.2.3 Security in SAFe  
   2.3 Threat analysis  
      2.3.1 Threat analysis in agile development  
      2.3.2 Threat priority  
      2.3.3 Security controls  
   2.4 Nixu  

3 **Purpose, aims, and motivation**  
   3.1 Delimitations  

4 **Related work**  

5 **Method**  
   5.1 Interviews  
   5.2 Identified problems  
      5.2.1 Changing attack-surface  
      5.2.2 Security as overhead
5.2.3 Separation of duties ........................................... 13
5.2.4 Risk owners .................................................. 14
5.2.5 Traceability ................................................... 14
5.2.6 Residual risk .................................................. 15
5.2.7 Penetration tests ............................................. 15
5.2.8 Funding ....................................................... 15

6 Requirements ..................................................... 15

7 Evaluation method ............................................... 16

8 Current threat analyses by Nixu ............................... 17
  8.1 Tools used ..................................................... 18

9 Adapting threat analyses for use in SAFe ..................... 20
  9.1 Categorization of threats .................................... 20
  9.2 Documenting security controls ............................... 21
  9.3 Link between security controls and threats ................ 23
  9.4 Threat scenarios ............................................. 24
  9.5 Evaluation of security controls .............................. 25
  9.6 Identify roles and responsibilities .......................... 26
    9.6.1 Risk management role .................................. 27

10 Threat-driven development in SAFe ......................... 27
  10.1 Identify new threats and adding them to the backlog during development 27
    10.1.1 Checklist and generic security templates ............... 28
    10.1.2 Abuse stories .......................................... 31
1 Introduction

Security in computer systems today focus largely on compliance with regulations, frameworks and practices [12]. This thesis presents a way of building the security of systems based on identified threats and by doing so time and resources are being spent where it matters most.

The software development industry has been revolutionised by the implementation of agile development methods [26]. These methods of development are designed to help software developers quickly create systems that better meet the expectations of the customers by building software in small iterations with constant feedback and involvement of the customer. There is currently no consensus on how security engineering should be done in agile software development. The Information Security Forum is a non-profit and independent organisation for information exchange between business leaders and information security experts around the world. In a survey amongst their members, 90% had adopted agile methodologies but 57% of those have no experience in secure agile development [26].

Nixu is a cybersecurity company that amongst other things deliver threat analysis to its customers. A threat analysis identifies assets, threats to these assets, possible attack vectors against the system that is being analysed and recommended protection against those attack vectors. This thesis presents a method of how to use these threat analyses in organisations using agile development and how to create an agile threat-driven development process.

This thesis presents Nixu’s current threat analysis techniques, suggests how to align the threat analysis to better fit the requirements of the threat-driven method later proposed in this thesis and finally how to apply this threat-driven method on an agile development framework. The suggested method is evaluated through interviews and a workshop.

2 Background

This section describes some of the concepts of agile software development and it introduces Scaled Agile Framework (SAFe) that is the agile development framework that the method presented in this thesis will focus on. It also describes threat analysis and the company Nixu.
2 Background

2.1 Agile software development

Agile software development is a collective term for software development methods and practices that follow the values and principles that are gathered in the Manifesto for Agile Software Development [1]. This manifesto was written by seventeen software developers in 2001 and contains these four statements of what to value in software development:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The manifesto in addition to these values contains twelve principles. One thing that these principles argue for is a continuous delivery of working software during the development process. This is normally done in iterations and the first delivery of software is a software that only have a very basic functionality. The software is demonstrated to the people that have ordered the software, from now on called stakeholders, and they are given a chance to provide feedback and discuss what the next functionality that should be added to the software should be. After the next iteration some new functionality will be demonstrated and the iterations continue. These iterations of delivering working software are the basic building blocks of agile software development.

An agile project starts with an initiation phase and various amount of project planning, how much depends on the developers. The development iterations then start with an iteration planning session were the features that should be implemented in this iteration are decided and put in a backlog. A backlog is an accumulation of work that is waiting to be done. To include the stakeholders in the process of selecting features for the backlog a representative is chosen, this representative is called a product owner. After the iteration planning the code is written, tested and demonstrated before the next iterations begins. Progress is measured in the new software features developed during the iterations. The project life-cycle is shown in Figure 1.

![Figure 1](example.png)

**Figure 1** Example of an agile project life cycle. With the three middle blocks being one development iteration.
In agile development, the software is constantly evolving throughout the entire development process. New features and requirements are established by the development team in close collaboration with the product owner and this is happening during all stages of the development process. The short duration of the iterations provides a short feedback-loop and minimize the work that might need to be discarded in terms of code, documentation, modelling and planning if requirements change or features are discarded. This does not mean that requirements can be changed at any time without costs but it facilitates changing the requirements.

Agile development teams are self-organizing, cross-functional and cover all parts of the development process from planning to testing. Being cross-functional implies that an agile development team should have the knowledge and skills needed to complete the project within the team. Team sizes are often between 5 and 11 people [3] and organisations that claims to be agile tends to be using either Scrum, Extreme programming, Kanban, Lean Development or they combine or pick the most suitable techniques from these methods.

### 2.1 Security in agile

Security is not mentioned in the agile manifesto and there is no consensus on how security should be integrated into agile development. There have been several ideas on how to use different security processes in the agile methodology. Building security architecture iteratively in parallel with the software [7] or integrating existing and introducing new security tools into the agile processes [25] [3] are some examples but many more exist.

Since agile development teams should be cross-functional, meaning that the required expertise needed to build the system should be available within the team itself, one way of increasing security in agile development is to improve the security knowledge of the development team. Other methods argue that even if increased security knowledge is good, putting all the security responsibility on the developers is in contradiction to practices of separation of duties, a well-known practice in many larger companies [8]. Instead, they call for security reviews and analysis conducted by teams independent of the development team [11] as well as improving the security skills of the developers.

### 2.2 Scaled Agile Framework

Scaled Agile Framework is one of the most used frameworks for scaling agile methods in big development projects on an enterprise level [13]. It is described as a knowledge
base of proven, integrated patterns for implementing agile development [19]. SAFe is a very comprehensive framework and only the parts that are relevant to this thesis are mentioned in this section. An overview of SAFe is shown in Figure 2.

SAFe can be used at different scale deepening on what levels of the organization you choose to include in the framework. The minimal level of implementation is called Essential SAFe and consists of two levels, the team level and the program level. Essential SAFe can be used for smaller projects while full SAFe can be used for larger projects or parts of projects that can require a hundred people or more.

SAFe works with the concept of program increments (PI), a PI is a limited time period where several different groups of people from different parts of the organization work together in what is called an Agile Release Train (ART). The ART delivers working and tested software functionality to the customer. The technical goals and business goals that are intended to be achieved during a PI are called PI objectives and they are evaluated in an event called Inspect & Adapt at the end of every PI where the current state of the solution is presented to the stakeholders. In addition to presenting the progress to the stakeholder one purpose of the Inspect & Adapt is to dedicate time to things like innovation and training that might otherwise be put aside when there is a constant focus on delivery.

The PI itself contains several development iterations, these are shorter iterations that represent the standard building blocks of agile development. The development iterations also provide working and tested software with each iteration.

### 2.2.1 The SAFe backlogs

There are several backlogs in a SAFe environment. In this thesis we will focus on the backlogs used by the different teams involved in an ART, the team level backlogs and the program backlog that is shared with all parties involved in the ART. The program level backlog contains the features for the ART that the team backlog tasks are then decomposed from.

**Features** are the backlog items kept in the program backlog on the program level. These features address user needs that can be implemented during a single ART. A feature is later decomposed to smaller user stories and tasks in the team level backlogs.

**User stories** are tasks on the lowest level in SAFe, the team level. User stories are shorter descriptions of a desired feature or functionality. These are written in terms that
the user would understand without knowing the technicalities that will make the desires outcome possible, often on the form:

As a {type of user}
I want to {do something}
so that {I can achieve a goal}

The content of a user story is always so small that it can be done in a single iteration. These user stories have an acceptance criteria in the form of a list of specific functionality that the team need to demonstrate or activities that needs to be performed before the user story can be considered as done. User stories on the team level are defined and prioritized by the product owner. The product owner is the only person on the team that can accept a story as done.

2.2.2 Kanban boards

In SAFe each backlog is a part of a Kanban board. In software development, Kanban is a method that helps teams manage work and oversee progress during development. Originally Kanban comes from the Toyota Production System [27] and in software development it is known as a lean approach to software development. Lean software development is the adoption of lean manufacturing principles and practices and it is a method of minimizing waste in the production system [18]. Kanban visualizes the workflow, the amount of work in progress and the average time it takes to complete tasks (cycle time or lead time).

Kanban is often visualized using a Kanban board that can be either physical or digital. A Kanban board consists of different columns where work is represented, usually in the form of some type of card that can be moved between the columns to illustrate how the work with that task is progressing. In SAFe there are different Kanban boards at different levels and the columns can vary depending on what SAFe level Kanban board it is but also depend on the organisation that is using it. Figure 3 is an example of how a program level Kanban board can look in a SAFe environment.
Full implementation of SAFe with all four levels represented in the image. [22].

Figure 2
Figure 3 An example of a program level Kanban board in SAFe. Items in the board move from left to right starting as an idea and going through all the stages until it is released and done.

One important concept with Kanban is that there is a maximum number of items that can be in the same column at the same time. This means that items might have to stay in the backlog column if there are too many items already in the implement column. This means that the progress is never faster than its slowest component and if one part of the organization tries to push more work through the Kanban board it will be clear that the new task will only create waste or delays elsewhere in the process.

2.2.3 Security in SAFe

Security is not discussed in depth in the SAFe documentation. Since the ARTs are supposed to have all the necessary knowledge on how to create the product being developed security specialist are considered part of the ART. Security specialists are however also considered a shared service, meaning that they will be shared between multiple ARTs and teams [21]. The reason for this is that the security specialist role is not needed in every ART all the time and security specialist are a limited resource.

SAFe also touches on the subject of security as a part of compliance [20]. Products with high demands for security often have high demands on compliance with different regulations. How to work with compliance requirements are considered by SAFe. There are few other areas where security is mentioned in SAFe and it is generally as part of
2 Background

quality.

2.3 Threat analysis

There are two primary goals of a threat analysis [12]. The first goal is to provide a clear overview of critical assets, threats and possible attack-vectors in order to facilitate a dialogue and help in decision-making actions regarding risks. The second goal is to identify the appropriate security controls at application, system, infrastructure and enterprise level and determine any residual risks.

2.3.1 Threat analysis in agile development

In agile software development and especially in an early stage of the development, the functionality and requirements are constantly changing and so is the attack surface of the system [3]. Threat analysis is about looking at the design of the system from an attackers perspective and in an agile project it is possible that not much has been decided about the design in an early stage. This will produce a threat analysis with a high-level of abstraction. Threats will have to be reviewed as the design changes and for this to be effective an up-front threat analysis is done.

A very early threat analysis can still identify some threats and risks in the application or system being built. Identifying trust boundaries between the new system or application and already existing systems and their already existing security controls. It might be possible to identify what the assets being used are and how that data is going to be stored and handled, what new security controls might need to be implemented and if the developers have the knowledge of how to implement these security controls.

Knowledge amongst the developers of how to implement security controls is a good example of threats in the development process itself. The initial threat analysis should cover these kinds of threats and not only technical threats. The threat analysis should determine if the developers have the experience and security knowledge needed to work with their tools and languages safely. It can identify the need for additional security tools in the development pipeline like dependency analysis or static code analysis tools.

After the initial threat analysis, ideally, another threat analysis should be done whenever there are changes in the systems attack surface. In an agile development project, this means that threat analysis might be conducted often but the scope should be narrow if the changes are small.
2.3.2 Threat priority

The threats identified in a threat analysis are prioritized based on their impact and probability. This process might include many people as the knowledge required to determine the impact of certain actions might not be known by the same people who are responsible for managing the systems. The main focus is often given to impact as the probability is very difficult to estimate.

2.3.3 Security controls

The designed responses against the identified threats against a system are called security controls [12]. The purpose is to protect the system by either counter, mitigate or remove the threat. A security control can be either a technical detail like input validation against injection attacks, it can be a more physical control like restricted access to a workspace by key cards or it can be a more administrative control like the process of account deletion when employees quit their job. Security controls can be preventive, detective or reactive.

2.4 Nixu

Nixu is a cybersecurity company that provides services in IT- and information-security [16]. One service that Nixu provides is threat analyses against IT systems and applications. Nixu is interested in extending the service of providing threat analysis to customers by adding support for using these threat models in the customers agile development environments, SAFe in particular.

3 Purpose, aims, and motivation

In order to help developers use threat analysis and threat models to create software with effective security controls, the purpose of this thesis is to present a method of integrating threat analysis into agile development methods such as SAFe. The goal is a threat-driven development method where threats are used to determine required controls and security features and it is possible to verify that these security features covers one or several of the identified threats. Any residual risk could also be identified and then be used for go/no-go decision for release deployment. Ultimately solving the question of how to make use of a threat analysis in an agile organisation.
Security is not a fundamental part of agile development and the Scaled Agile Framework does not provide explicit instruction for how to deal with security and threats to the system being developed. At the same time, the 2017 Verizon Data Breach Investigations Report states that 30% of all data breaches target the application layer so there is a need for improved security in applications.

Security engineering is often driven by compliance requirements [12], this tends to put the focus on the controls rather than the threats. By identifying and prioritizing the threats against a system it is possible to use the identified threats as a foundation for security engineering. By focusing on the identified threats no resources are wasted on controls that do not mitigate an actual threat and it highlights any residual risks.

### 3.1 Delimitations

The proposed method in this paper does not try to provide a full security solution for agile software development but tries to provide a method of using a threat analysis in agile development processes.

The method proposed in this paper focuses not on the part of the threat analysis where threats and security controls are identified but on the part of a threat analysis where the security controls are actually implemented, validated and verified.

The proposed method does not consider other agile development methods than SAFe or threat analysis methods that differ from the ones used by Nixu.

### 4 Related work

This master thesis is related to studies on threat analysis and threat-driven security methods but also to studies on security in agile development. Many ideas in this thesis are based on concepts of the related work presented in this section.

Antti Vähä-Sipilä discusses how to integrate security in agile development during the OWASP AppSec Research conference 2010 [29]. The presented method is tested on Scrum but should also be applicable to other agile methods. The methods suggest making threat modeling visible as tasks in the backlog in the form of research spikes, a type of task in agile methodology that does not create any deliverable code. Tasks in the backlog will be prescreened using a filter checklist to determine if the task is risky. If a task is determined as risky, another task would be added to the backlog stating that there needs to be a threat analysis for that task. The focus on the proposed method is to
make threat analysis and threat modeling a part of the agile methodology and something that is visible and traceable rather than overhead or extra work. This presentation covers many of the areas that are covered in the method proposed in this paper but does not discuss how to integrate an external threat model into the process nor does it focus on SAFe.

Michael Muckin and Scott C. Fitch from Lockheed Martin published a white-paper where they propose a threat-driven approach to cyber-security [12]. They present a framework to select, implement and evaluate the effectiveness of security controls. They propose the use of several different tools, practices and methodologies. Like the proposed method here, threats are identified using threat modeling, attack trees and attack profiles and then the security work is based on those threats creating a threat-driven process. The threat-driven approach to build and manage your security controls is implemented in different extent in this thesis. The framework suggested does however focus more on a company-wide processes than development and does not claim to be agile.

A way to formally prove that software has been secured against anticipated security threats is presented by Dianxiang Xu and Kendall Nygard [32]. This is done using Aspect-oriented Petri Nets. Petri nets are a formal method used for specification and analysis of distributed systems and it provides both a mathematical and a graphical notation. This paper is related to the work in this thesis as the goal in both cases is to use a threat analysis, or model, to verify that the threats have been addressed. The difference between their paper and this thesis is the method proposed in this thesis is not a formal method and it does extend its focus outside of just software development.

Microsoft Security Development Lifecycle (SDL) [10] is a software development process that includes the use of threat models in agile methodology. Microsoft SDL suggests considering threat modeling as a reoccurring event for each iteration of software development. When the next iteration is being planned, the development team should discuss if any threat models need to be updated or added during the project based on the user stories, or epics, that will be implemented during that iteration. Microsoft SDL is very focused on Microsoft threat models and their own development process that differs from SAFe and the Nixu threat models.

Othmane et al. [4] propose a method with the goal of efficiently ensuring security at the end of each development iteration by integrating security engineering activities into an agile development method. One of these security engineering activities is risk processing and is integrated into agile processes by introducing a threat analysis with threat modeling in the initial phase of the software development process. Implementation of controls is not threat-driven in the same sense as the method proposed in this thesis.
5 Method

Interviews have been carried out with developers that have been using the threat analysis provided by Nixu and with Nixu personnel that conduct these threat analyses. The composition of the interviewees was three security analysts from Nixu where all had some background in software development and two developers that also had some experience in software security. The goal of the interviews has been to identify problems in projects when trying to handle the threats identified and implementing the suggested security controls.

The interviews are semi-structured, meaning that some questions were defined in advance but much room was given to follow-up questions and discussions [6]. This was necessary as the organisational structure behind the different development projects differ and the problems identified can require a further discussion to find the cause and possible solutions. Semi-structured interviews are a qualitative method of information gathering with more focus on interpreting what is being said than to collect a large set of data.

Problems that have been identified in the related work studied are also addressed in the proposed method.

Research, other documented methods or suggested methods have been studied and considered from the perspective of these identified problems. These solutions are then put together into a method of integrating Nixu’s threat analysis into SAFe.

5.1 Interviews

The interviews have been focusing on three major topics by discussing a list of predefined questions. The first topic discussed have been about what the interviewees previous experience and roles have been in different projects that somehow involve threat analysis, agile development or both. This was discussed in order to get a better understanding of the tools and methods used by the interviewees and their experience in relevant subjects.

The second topic has been about the outcome of different projects the interviewees had been involved in. This was discussed in order to get an understanding of the conditions of different projects and how much experience and knowledge the teams had in the related fields. What went well and what problems were encountered in these projects.

The third topic for discussion was what the interviewee could use to make work like this
more successful and more efficient in the future. This was discussed to bring forward any ideas of solutions the interviewee could have.

The goal of these interviews was to identify problems and solutions to working with threat analysis in agile development, this goal was described to the interviewees before the interview. Predetermined questions used in the interviews can be seen in Appendix A.

5.2 Identified problems

This section describes the identified problems regarding security and the use of threat analysis in agile development methods. These are problems that are considered by the method proposed in this thesis.

5.2.1 Changing attack-surface

One of the major problems with security in agile development is that the constant evolution of the systems is creating an ever-changing attack-surface [3]. The rapid development means that a threat analysis might be obsolete just iterations after it was done. How this is handled and how to be aware of the systems attack-surface and when it is changing is one of the major problems with security in agile development.

5.2.2 Security as overhead

Security work like threat analysis can easily get put aside in agile methods as is mentioned by Antti Vähä-Sipilä [29]. Since the agile methodologies focus on a quick and continuous release of working software any work that hinders deployment or is invisible in the process, consequently overhead work, gets put aside as progress is measured in delivered working software [2]. This problem is also related to the problem with funding (see section 5.2.8) as extra work cost extra money and as overhead work might not always be visible it is hard to say why time and money is spent and on what.

5.2.3 Separation of duties

Separation of duties, sometimes called segregation of duties, is used as a protection against fraud and errors. The idea is that by dividing a process among several people no
single person should be able to take advantage of a situation for personal gain or other misuse [9]. If considered from another security perspective this would also apply to security testing of software. There is a general agreement that software testing improves the quality of the software. One study also suggests that testing done by independent testing teams improve the quality of the software more than testing done by integrated testing teams [28].

During the interviews none of the interviewed persons thought that separation of duties was necessary for the reasons mentioned above but rather for the reason that a person not previously connected to the projects would have easier times questioning decisions made during the project development and would approach the project from a different perspective and thus identifying problems previously overlooked.

Separation of duties is also considered a fundamental control in security and governance frameworks such as ISO 27001 [5].

5.2.4 Risk owners

Once the threats are identified one problem that was identified during interviews is how to determine who actually has the responsibility and ownership of the risks this threat imposes and the implementation of its proposed mitigations. As security controls might need to be implemented on different levels within the company, or even outside the company, it is often hard to identify who owns the responsibility to protect against the threat. Without an owner of the threat identified or with an owner that does not have the ability to mitigate the threat they do not get mitigated and the problems remain.

5.2.5 Traceability

A problem during the process of mitigating the threats identified in the threat analysis was how to connect a specific mitigation to a specific asset and security control and vice-versa. This was identified during interviews and is also mentioned by Muckin and Fitch in their white-paper [12]. The problem is most apparent when security controls need to be verified and validated. If there is a list of threats and a list of security controls but no apparent connection between them it is difficult to know why security controls are in place, what their purpose is and if security controls exist against a threat or not. Often there is no list of security controls at all.
5.2.6 **Residual risk**

Related to the problem of traceability is the problem of determining residual risk. It is difficult to determine what threats have not been mitigated and what threats remain. It might not be possible to mitigate all threats and it will then be necessary to look at all residual risks as the list of residual risks might be an argument for deployment into the production environment or not.

5.2.7 **Penetration tests**

A penetration test is a simulated attack against a system that can be used to identify weaknesses and strengths. During the tests, penetration testers act like attackers and try to use the same tools and practices that a possible attacker could use to be able to break and exploit the system. This can be used either to identify vulnerabilities in the system or as a confirmation that the security controls in place work.

One problem with the rapid pace of agile development is that it requires advanced penetration test on a very short notice. Development iterations that are only weeks long can not spend weeks waiting for a penetration test but at the same time may require experienced penetration testers with good knowledge of the systems being used. This problem was identified during the interviews.

5.2.8 **Funding**

When threats are identified in a system the cost to address all threats can be very high, especially if they are identified late in the development process and will require major system changes. The problem here is to provide the information necessary for stakeholders to make decisions on what to address and why. The information needed is often not technical but have to cover both financial and business perspectives on the risks.

6 **Requirements**

The method proposed in this report should consider the problems that are identified during the interviews or in related work. The identified problems are listed under Section 5.2. The method does not need to solve the problems but should discuss possible solutions or explain why they are not solved. Advantages and disadvantages of using the proposed solution should be discussed in the thesis.
The method proposed in this report should have a list of entry-criteria and requirements that will be used to determine if the proposed method is applicable to an already existing agile development process or not. These should be minimum requirements and the method should not be possible to implement if these requirements are not meet. These entry-criteria are the following.

- During planning sessions of ART representatives from necessary parts of the organisation are present and all have access to the shared backlog on the program level of SAFe.
- Backlogs needs to be managed digitally with the ability to connect tasks to features across different backlogs.
- An initial threat analysis has been done for the project or organization and this threat analysis include a list of identified security controls.

The method proposed in this report should be applicable to the SAFe framework for scalable agile development. It should not be necessary to remove any parts of SAFe and suggested solutions should not make it difficult to use the practices and techniques that are a part of SAFe.

The method proposed should be understandable and implementable by developers who are not security experts and it is understandable for security experts with little knowledge of software development.

### 7 Evaluation method

The development method proposed in this thesis is evaluated using interviews with relevant individuals and by a workshop where parts of the method will be applied to an imaginary development scenario.

The evaluation interviews are conducted with developers and security analysts that have been involved in related projects, the interviewees were informed of the goal of the interview in advance. Five individuals were interviewed during the evaluation. Two software developers, one project manager and two security analyst. The method was presented to the interviewees before the interview began and was complemented by a short presentation at the beginning of the interview. The list of interview questions can be found in Appendix C. The questions are based on the requirements in Section 6 and the identified problems in Section 5.2. The interviews are semi-structured in the same manner as the interview conducted to identify the problems to allow a discussion about
the problems and benefits of the method proposed in this thesis. Additions to the method were also discussed during the interviews.

The workshop was an imaginary development scenario where an imaginary mobile application was created. The workshop started with a ready backlog of features to implement in this scenario and a suggested architecture. A threat analysis was conducted during the workshop with the participants. After the threat analysis was done parts of the method suggested in this thesis was evaluated by adding the threats to a digital backlog as abuser stories and security features with the suggested security controls linked to them. Once the workshop was done the method was discussed and evaluated with the participants. The workshop-group consisted of 7 individuals with different backgrounds in the area of software security. At the time of the workshop, the interviewees were either working as security consultants or developers in the area of identity and access management systems. Two security analysts from Nixu was also in place during the workshop to help perform the threat analysis.

The interviews evaluated all parts of the method proposed in this thesis while the workshop did not include a list of security controls, threat personas or security checklist it did focus on the problems with changing attack surface, security as overhead, traceability and residual risk acceptance.

The evaluation results are presented in section 12.

8 Current threat analyses by Nixu

From the interviews conducted it is clear that how a threat analysis is done in Nixu varies depending on the analyst who is conducting the analysis and also depending on the customers expectations. However, the common structure of an analysis is the same. Initially, documentation is gathered and reviewed before one or several workshops are used to identify the threats to the organization being analysed. The common outcome of a threat analysis from Nixu is a list of identified and prioritized threats with proposed mitigations in the form of suggested security controls. The final deliverable can then be a report, a slide presentation, a workshop or usually, a combination of several of these. After the final delivery, there are normally no follow-ups and feedback is normally not received.
8.1 Tools used

The different analysts use different tools in their workshops and reports. Two tools that are currently being used and that will be particularly useful in a SAFe environment are attack trees and security personas.

**Attack trees**

Attack trees are graphs that represent attacks against a system [23]. In these graphs, the root node is a goal that the attacker wants to achieve and the leaf nodes are different paths to achieving that goal. The attack trees do not need to be complete or completely accurate but they serve as a tool for discovering potential attack-vectors, weaknesses and required security features.

![Example of an attack tree](example.attack.tree.png)

**Figure 4** Example of an attack tree where the goal of the attack is to enable the attacker to repudiate actions or data.

Attack trees can be complemented by a list of threats, the priority of the threat and their proposed protection and security controls. An example of an entry in such a list would be:
<table>
<thead>
<tr>
<th>Priority</th>
<th>Protection</th>
<th>Against threat</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Ensure that all actions user can invoke are protected by a non-predictable token in the HTTP request.</td>
<td>Cross-Site Request Forgery (CSRF)</td>
</tr>
</tbody>
</table>

Table 1 Example of priority, threat and mitigation for one of the threats in the attack tree in figure 4.

The mitigating security controls can also be shown in the attack trees themselves to make the connection between security controls and threats even clearer. Attack trees can either be used in the workshop or as part of a report or both.

**Security personas**

One tool used during these workshops are security personas (or cyber boogies). The use of personas is a tool that is already used by many agile teams [3], they are descriptions of different types of people who will be using the system or application and they include descriptions of their experience and technical ability with the goal of getting the developers to think about how the users might want to use the system, what features they might want and what test scenarios that will need to be tested.

A security persona is a fictive person that might intentionally or unintentionally cause harm to the system and the developers should use these security personas to think about how to introduce design features and security controls that might prevent these users from reaching their goal. The number of security personas should be kept low and there should be one persona representing each type of user. One example of a security persona might be a compromised system administrator.

Security personas are designed to help people think about security and to identify risky features. The use of security personas help to identify current threats to the system but can also be kept in place during future development to keep developers thinking about threats during the development. By keeping the security personas in mind the developers might have an easier time realising when a new or changed feature affects the security of the system.
9 Adapting threat analyses for use in SAFe

In order to align Nixu’s threat analysis with the proposed method of how to use threat analysis in SAFe in Section 10 this section contains proposed changes and tools to be included in the threat analysis. The four main concepts are to categorize threats, list security controls, connect those two lists to make them easier to process and to create threat scenarios for evaluation.

9.1 Categorization of threats

The threats that are identified can be categorized into groups in order to aid in the connection between threats and what security controls are in place against this category of threats. There are several approaches on how to do this, one is using the CIA-triangle, dividing threats into the categories of what they are a threat to using the categories confidentiality, integrity and availability. Another popular method is STRIDE [24] that is an acronym for spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privileges. Each word in the STRIDE acronym represents a threat that matches a security property that they are a threat to. As an example tampering is a threat to integrity and denial of service is a threat to availability (see Table 4).

This relationship makes the connection between threats and their security controls in a threat-driven methodology clearer. An example is the threat of a user getting access to another users information and we will categorize it using STRIDE. In STRIDE this threat could be categorized under Spoofing (S), Tampering (T) and Information Disclosure (I) with the properties and controls listed in Table 4. The categories of controls here are an example and should be adapted to the system where they are being applied and should use the same categories later used to categorise the security controls.
### 9.2 Documenting security controls

Security controls need to be documented in order to effectively evaluate them. In a threat-driven environment, security controls can be categorized in regards to the security
function they want to accomplish. These security functions are the primary objective of the security controls categorized under them. The U.S National Institute of Standards and Technologies (NIST) suggest the use of Identify, Protect, Detect, Respond and Resolve as these functions in their Cybersecurity Framework [14] while Muckin and Fitch use Inventory, Collect, Detect, Protect, Manage and Respond [12] in what they call a Functional Control Hierarchy (FCH). These functions and their extending hierarchy should be adapted to the needs of the organization and its systems. The functions extend to categories, these categories should directly relate to either a property or a control in the threat categorization that is being used. The next level of the hierarchy is the actual security implementations, these are instances of technology, services, processes or products that implement the security features. The alignment between the security controls categories and the threat categories is essential in order to connect threats to security controls.

![Figure 5 An section of a security control categorisation.](image)

There are other categorizations for security controls that can be useful in the development process. Columns can be added to the different implementations to determine if the nature of the controls e.g. physical, organizational, technical. Another useful categorization would be what team within the organization that owns the responsibility for the security control. The purpose of adding any additional columns should always be to support using the lists as efficiently as possible.

Muckin and Fitch present an example of a basic FCH that can be used as the foundation to build organization specific FCHs [12]. Security controls on this or similar list can be used as a starting point for listing security controls in place. If controls exist on the list but currently not in the organization that might be a gap in the security controls. Security controls not in place can be kept and marked in the list to make it easy to identify what security controls that do not exist.

The list should be owned by someone who has an architect role in the organization and
when controls are either removed, added or changed the table should be updated. In order to make the table useful, it should represent the current state of the system at all times.

### 9.3 Link between security controls and threats

In the threat categorisation, the column of mitigating security control categories against a threat category should be using the same categories as in the list of categorised security controls as this makes using the list of security controls easier [12]. It should be easy to see what security controls are already in place against a threat in a certain threat category. If the list of security controls also shows suggested control categories that are not yet implemented it also helps to determine new security controls.

<table>
<thead>
<tr>
<th>Threat type</th>
<th>Property</th>
<th>Controls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spooiling</td>
<td>Authentication</td>
<td>Authentication and session management, digital certificates.</td>
</tr>
<tr>
<td>Tampering</td>
<td>Integrity and Access Controls</td>
<td>Digital signatures, checksums, sequence accounting, access control, auditing.</td>
</tr>
<tr>
<td>Repudiation</td>
<td>Non-repudiation</td>
<td>End-to-end auditing and logging, sequence accounting, digital signatures, fraud prevention.</td>
</tr>
<tr>
<td>Information disclosure</td>
<td>Confidentiality</td>
<td>Encryption, data tokenization.</td>
</tr>
<tr>
<td>Denial of service</td>
<td>Availability</td>
<td>Rate limiting, deception, elastic availability</td>
</tr>
<tr>
<td>Elevation of privileges</td>
<td>Authorization and least privilege</td>
<td>Authorization and least privilege.</td>
</tr>
</tbody>
</table>

![Table of security controls and threats](image)

**Figure 6** The link between control types against specific threats and the categories of listed security controls makes it easier to see what security controls are in place against a specific threat.
9.4 Threat scenarios

When the threats are prioritized they can be used to create threat scenarios for the most significant threats [12]. Threat scenarios represent an attack or something that should not be possible for an attacker to achieve [3] and in that sense, they are similar to abuse stories that are discussed in more detail in section 10.1.2. A useful tool when using threat scenarios is attack trees and other visual threat models as they help in visualising the threat scenarios. The threat scenario has to include the attack surface, the most common attack vectors and the asset that is the target. One example is the attack scenario where the threat is that users will be able to access other users information remotely. This might be considered as a high-level threat as the information might be used against the users and it will also be a privacy violation, both could have the consequence of reputation and financial loss.

**Threat scenario: User can access other users information remotely.**

<table>
<thead>
<tr>
<th>Asset</th>
<th>Threat categories</th>
<th>Attack surface</th>
<th>Attack vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>User information</td>
<td>• Spoofing</td>
<td>• User</td>
<td>• Attackers spoofs as the user.</td>
</tr>
<tr>
<td></td>
<td>• Tampering</td>
<td>• App server</td>
<td>• Exploitable vulnerability.</td>
</tr>
<tr>
<td></td>
<td>• Information disclosure</td>
<td>• App client</td>
<td>• Data is associated with wrong user.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• OS</td>
<td>• Authorisation data is wrong.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Network</td>
<td></td>
</tr>
</tbody>
</table>

*Table 4* Parts of a threat scenario categorized using STRIDE and containing assets, threat categories, attack surface and attack vectors.

Threat scenarios can be used as a tool to evaluate security in an organization. By creating threat scenarios for the most severe threats to the organization and evaluating how well the system is protected against them you get a measurement on the security implemented in the system. Saying that the 10 most serious threats to an organization have been mitigated is easier to understand for the product owners than just stating that there are
20 new security controls against threats in place. It also helps to illustrate residual risks as it shows what threat scenarios have been mitigated and to what extent. The threat scenarios should also show what additional security controls that could be implemented to mitigate the threat further.

The threat scenarios should be described in a way that the product owner can understand and documented with the attributes threat categories, attack surface and the identified attack vectors. The security controls that the threat analysis suggests as mitigations against the threat scenario or against the different attack vectors should also be documented in the threat scenario. The use of attack trees helps illustrate the different attack vectors and how the suggested security controls relate to these attack vectors and the threat scenario as a whole.

The number of threat scenarios should be kept manageable and should focus on the most significant threats identified.

### 9.5 Evaluation of security controls

In order to evaluate how well the controls mitigate the threat scenarios or other threats, their effectiveness has to be determined. This is done by comparing a threat scenario to the list of security controls and determining the effectiveness against that particular threat scenario (see Figure 7) [12]. The effectiveness of a security control is specific against a single threat.

![Figure 7 Example of effectiveness in regards to a specific threat scenario. Effectiveness is in this example rated as None, Partial or Full effectiveness.](image_url)

In order to verify that the controls are indeed effective, manual security tests can be conducted. In a stage of the development where many new security controls are being implemented at the same time, the problem with penetrations tests in agile development...
discussed in section 5.2.7 is not as apparent as it is when penetration tests need to be conducted to verify a single security control as there is more room for foresight.

The penetration testers can see what security controls that need to be tested from the list of security controls and in what context those security controls need to be tested from the threat scenarios. Since time is likely to be a critical factor the penetration testers should be given access to the source code in order to be more efficient in their assessment.

By using the attack cases designed for the most significant threats identified during the threat analysis and the list of categorized controls you get an overview of how well protected the organization is against its most severe threats.

9.6 Identify roles and responsibilities

In a threat analysis there need to be people from different parts of the organization and who have different roles within a project involved.

The risk of a threat being realized will always relate to an impact on the organization, as a consequence there should always be a person that has the ability to accept risks on behalf of the entire organization. This individual has the responsibility that the project meets the organizations requirements in regards to risks and audits and will have to be somewhat involved in the creation of security requirements. This person should preferably have some leadership role and should be responsible to make sure that the other people who need to be present at a threat analysis are present as this is paramount to the quality of the threat analysis.

The person who is responsible that the security requirements are actually implemented in an agile environment is always the product owner as the product owner is the only person that can accept a user story as done.

The person responsible for security threats regarding the development pipeline is the lead developer or scrum-master in the scrum method.

One or more persons who know about the environment where a system or application is being implemented should also be present as threats might need to be mitigated by security controls somewhere else in the production environment and security controls that are already in place in other areas of the environment might not be known by the other people involved in the project. This person is probably someone in the role of system architect and should preferably be the one responsible for the list of security controls discussed in section 2.3.3.
9.6.1 Risk management role

Nelson et al. find in their study that it is important to have a team member in the development teams with the role of risk management [15]. The person with this role would be responsible for keeping the team focusing on identifying and handling identified risks. This role could be rotated between team members and does not need to be assigned to a specific member of the team.

This can be implemented in SAFe by adding another risk management role to the program level as well as in the team level suggested in the study. The risk manager on the program level would be responsible not only for risks what will be handled by internal development, the risk manager would also delegate risk handling to other parts of the organisation like operations and IT and would be the person responsible for risks that are outside of the organization and contact with third-parties to handle these risks.

The tracking of risks and their current status is supported by keeping them stored and tracked in the backlog, this is further discussed in section 10.3 and section 10.4.

10 Threat-driven development in SAFe

This section will present several tools and techniques that can be used in order to create an agile threat-driven development process.

In order to address the problem of security becoming overhead work (see Section 5.2.2) as much security work as possible will be represented in the backlog. By representing security work as tasks and features in the backlog it becomes visible and traceable just like any other development task, it becomes less of an overhead and less likely to be optimised out of the development process [30].

First, the process of identifying new threats and adding them to the backlog is described and later the process of taking an entire threat analysis and integrating it into the development process.

10.1 Identify new threats and adding them to the backlog during development

There are two major security concerns when new features and requirements are created and put in the backlog. The first is to detect when a new feature affects the security
of the system and the second is how to determine the required security features and writing them as user stories. The product owner and the developers have several tools to help them identify risky changes to the system and to add them to the backlog. These tools include checklists, generic security stories, abuse stories and the security personas mentioned in section 8.1. These are all improved further by the defined threat scenarios from the threat analysis.

When new features are proposed they will go through the funnel and analysing stages of the program kanban board before ending up in the program level backlog. The best time to identify new threats is during the analysing stage.

When a new feature is found to introduce a new risk or affecting some security control a backlog item will be created for that threat. This backlog item will be a smaller threat analysis or threat analysis spike [29] that is connected to the feature in the backlog. During this threat analysis, any existing related security controls or new security controls that are needed should be added as acceptance criteria to the feature in the program level backlog. If the new risks should be comprehensive a new security feature or an abuser story can be created and added to the program backlog as a separate feature.

### 10.1.1 Checklist and generic security templates

If a threat or risk will require more than one security control it might require its own security feature or abuser story in the backlog.

Generic security templates are security user stories that are too generic to be able to be copied directly into the backlog but will help the product owner to formulate the security requirements by example. One example is the security user story in table 5 written by Vähä-Sipilä in the Handbook of The Secure Agile Software Development Life Cycle where he also suggests a number of questions for the checklist [30]. In order to identify when you need this particular security user story a checklist containing multiple questions is used. A question from this list that would highlight the need of the user story in Table 5 would be:

"Do you process information about a user?"

If the answer is yes then a number of generic security user stories linked to that questions will be suggested by the checklist. These user stories will then need to be rewritten into the context of the system in question and added to the backlog. The checklist also needs to be adapted to fit the project and the questions that need to be asked should be designed using the threat analysis and especially using the threat scenarios.
Every threat scenario that has been designed should have one or more security questions connected to them. If the answer to a question is yes then the threat scenario will be added to the program backlog again for validation and verification to make sure security controls still protect against the threat scenario.

Using both threat scenario level checklists and the more generic security templates and checklist is recommended to provide good baseline security when adding new features to a system.
<table>
<thead>
<tr>
<th>As</th>
<th>a user</th>
</tr>
</thead>
<tbody>
<tr>
<td>I want</td>
<td>that access controls effectively prevent all unauthorised actions or operations.</td>
</tr>
<tr>
<td>So that</td>
<td>I am protected.</td>
</tr>
</tbody>
</table>

**Description**

Access controls (whether on user level, or component level) must prevent all unauthorised actions, and a key to this is to have a clear and understandable access control model. Clarity of code and maintainability are critical quality aspects. It is also important to use the “least privilege” principle, meaning that users and components are only given the access they really require in the policy, and not anything “just in case” or “because it did not work otherwise”.

**Acceptance criteria**

- Negative tests that try to do accesses that are prohibited by access control policy (i.e., should not succeed).
- Exploratory (manual) tests by a professional that try to bypass access controls (in ways which have been left undefined by policy, for example).
- If there are “test accounts” or access control has a “test mode”, there must be a test case that verifies that these test accounts and modes have been disabled in released code.
- Tests exist where several users in different roles use the system concurrently, and check for concurrency problems.

**Refinement question**

- Are all access control decisions centralised in one (or few) components instead of being spread all over?
- Do all of your components require appropriate permissions from other components that call them, and do your components only have those permissions that they really require?
- Can you easily explain how your access control works?

**Table 5** Example of generic security user story for access control by Vähä-Sipilä as presented in Handbook of The Secure Agile Software Development Life Cycle [30].
10.1.2 Abuse stories

The second tool that Vähä-Sipilä suggests is to allow the product owner to be able to formulate requirements as abuse stories [17]. Abuse stories are user stories from an attackers perspective and these stories state what should not be possible for an attacker to achieve. An example would be:

As an attacker
I should **not** be able to hide my activities
So that I can repudiate my actions.

This allows the product owner to state what should not be possible without having a technical understanding of the problem. The ability to say what is not supposed to happen becomes useful when a threat is identified to a system but the product owner does not know what would prevent it, to the product owner can then say what should not be able to do and the developers will then need to formulate the backlog items to mitigate the threat. The attacker perspective fits well with threat analysis that also looks at a system from the eyes of an attacker. These stories are also referred to as attacker stories in literature.

10.2 Security in Definition of Done

The Definition of Done (DoD) is a list of activities that verify or demonstrate that a feature is done from a predefined set of requirements that will be the same for all features, this is commonly used in Scrum influenced agile development teams. It is used to make sure everyone agrees on what it actually means when the teams say that a feature is done. The activities in the DoD can be things like unit testing and documentation. When a sprint is ended the team can then ask themselves if they have completed all aspects of the DoD for the latest feature, if not they are not done. Adding security aspects to the DoD makes sense but can be problematic in regards to the problem of security becoming overhead work (see section 5.2.2). From a threat-driven perspective adding security to the DoD could be in form of the question: “Given the found threats, are we done from security p.o.v.?” [29]. For this to be effective the organization adding these questions to their DoD needs to make sure the development teams actually have the ability and an interest in stating that they are not done if that is the case. In an agile development environment, this can be very rare when the focus is on getting software delivered and doing it quickly [31]. If the focus on adding security is shifted to the acceptance criteria of user stories instead it will become a part of the backlog rather than overhead work.
10.3 Tracking the backlog

Documenting the backlog becomes much easier if the backlog is digital rather than a number of post-it notes on a board at the office [3]. Since a digital backlog in most cases automatically logs changes it provides an audit trail to when and by whom changes were made. Work done can also be connected back to a user story and the progress on that user story can then be monitored. It provides searchability for stories and issues which can then be flagged for analysis or review. The user stories might be abuser stories or security features that represents a threat scenario and the progress would then show what security controls that have been implemented against that threat and how far the process of handling the threat have progressed. It will also help in illustrating the residual risks that are security controls not in place.

It was also identified during the interviews that for someone who is working with a development team in a security role, like a security engineer or auditor, and might only see the backlog once or twice a week the possibility to quickly see changes done in the last week and similar features become very useful.

10.4 Integrating the threat analysis with the backlog

The process of using a threat analysis in a development project can be described as a four-stage process (see Figure 8). The first stage is when the threat analysis is done and the threat scenarios are created. The second stage is when controls and requirements against the identified threats are determined and implemented. The third stage is validation and verification of the implemented security controls and that the security requirements are met. The final stage is to determine if any residual risk is acceptable or not.

![Figure 8 Four stage process of using a threat analysis in software development.](image)
When a threat analysis is done the development team should have lists of identified and prioritized threats, assets, security controls and threat scenarios. The first task of integrating the threat analysis into the development process it to add the threat scenarios to the program level backlog as abuser stories or security features (See Table 6). These security features will have the security controls that are needed to mitigate the threat or implement the security feature as acceptance criteria.

<table>
<thead>
<tr>
<th>As</th>
<th>an attacker</th>
</tr>
</thead>
<tbody>
<tr>
<td>I should</td>
<td>not be able to do something bad</td>
</tr>
<tr>
<td>So that</td>
<td>it will hurt the organisation or its customers.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the abuse story and its attack vectors.</td>
</tr>
<tr>
<td>Acceptance criteria</td>
<td>• Security control 1</td>
</tr>
<tr>
<td></td>
<td>• Security control 2</td>
</tr>
<tr>
<td></td>
<td>• Manual testing and verification of security controls</td>
</tr>
<tr>
<td></td>
<td>• Other acceptance criteria</td>
</tr>
</tbody>
</table>

**Table 6** The structure of an abuser story with the suggest controls needed in the acceptance criteria.

Since the list of threats is already prioritized the backlog prioritization should be easy to implement. SAFe uses the Weighted Shortest Job First (WSJF) algorithm for backlog prioritizing and risk reduction and time criticality is already factors in that formula so any serious threats should have a high priority in the backlogs since they naturally have high values in both these factors. It is important to remember that all teams can use the backlogs to represent their work, it is not limited to developers.

\[
WSJF = \frac{User/ Business Value + Time Criticality + Risk Reduction}{Job Size}
\]

At the start of every PI the teams in the ART will take the top prioritized items from the program backlog. In the PI Planning process the items will be split into smaller stories and tasks to be added to the appropriate team backlog. There it will become the specific implementation of the determined security control.
Figure 9 Security features or abuser stories have their mitigating security controls as their acceptance criteria. These security controls are put as user stories in the appropriate teams backlogs during the PI planning.

10.5 Baseline security

Some security controls and features can be considered baseline security and are repetitive tasks that do not make sense to keep in a backlog. An example would be to process of doing daily backups of databases. These kinds of security controls should be included in the list of security controls but not the backlog. By keeping them where they are visible so that the knowledge of them is known and documented they will be validated and verified when a threat scenario connected to the security controls is.

10.6 Validation

Validation is about determining if the right security controls are being implemented. By looking at the most serious threat scenarios and their corresponding abuser stories or security features in the program backlog it is possible to see what tasks are related
to the threat and if and when they were implemented (see the identified problem about traceability in Section 5.2.5).

When security controls are implemented or identified they should be added to the list of existing security controls. The list of security controls provides a second connection between threats and their security controls by the threat categorisation with suggested control categories. The connection between threats and security controls can be improved further by adding another column to the security control list representing what threat it mitigates and adding that information during validation. By allowing it to be sorted by threat or threat scenario it will become easy to determine what security controls are in place against a specific threat scenario as well as why a security control is in place.

10.7 Verification

Verification is to confirm that the security controls work as intended, this is done by conducting code reviews and penetrations tests. When code reviews and penetration tests are being done it is worth considering segregation of duties (see Section 5.2.3) and to use security experts from outside the development team that implemented the security controls. An additional benefit of using people who are not involved in the specific development project that is being tested is that they might have a different perspective and an easier time questioning decisions made during development. This benefit was mentioned by several people during the interviews.

10.8 Residual risk

The review of the latest PI in SAFe, Inspect and Adapt (I&A) is a good place to do residual risk acceptance. Since the only person who can accept features as done is the product owner it is very important that the product owner understands what security measures that are implemented and what the residual risks are.

Residual risk can be identified using the threat scenarios. What security controls that are in place and how well they protect against the threat scenario should be possible to see both from the tables created when threat scenarios are evaluated against the list of security controls and from the digital backlog items representing the threats scenarios. All these items combined should provide enough information to the product owner and stakeholders to make decisions on what risks that are accepted. The accepted risks will also be visible in both the backlog and in the lists where security controls are evaluated against threat scenarios.
11 Development process overview

This section presents a brief overview of the development process suggested in this thesis.

The threat analysis provides a list of threats, a list of existing security controls and a number of threats scenarios with suggested security controls against that threat scenario. These threat scenarios are added to the program level backlog as security features or abuser stories. These features or stories include the suggested security controls in their acceptance criteria. These security controls are assigned as tasks to the appropriate teams during the iteration planning and put in their respective team backlogs. Already existing security controls are identified from the list of security controls or new security controls are implemented. New security controls are added to the list and the use of a digital backlog enables progress on the threat scenarios to be monitored in real-time as tasks are completed. Verification of security controls is done using manual penetration tests and code reviews and verified security controls are used to update the threat scenario evaluations.

New features are checked for new risks and threats using a checklist and with the help of threat personas. Risky features are subject for a small threat analysis where potential security controls are identified and added to the acceptance criteria of the feature, this threat analysis is a separate backlog item. Larger threats that require many security controls from different parts of the organisation are added to the backlog as their own security features using generic security templates.

Any threats that are not part of a threat scenario is still categorised and any gap in mitigations should be possible to see from the security control list. Any residual risk acceptance is done by the product owner who has access to the threat scenarios and security control list.

12 Evaluation results

All the comments in this section are based on feedback from interviews and the evaluation workshop. The feedback received during the interviews and during the workshop differentiated between three different parts of the method proposed in this paper. The first part was the handling of threats in the backlog and integrating the threat analysis into the development process, the second was the use of a security control list, and the third one was the process of identifying new threats during development.
Figure 10: An overview of the agile-threat driven development methods proposed in this thesis.
The process of creating threat scenarios and writing them as abuser stories or security features in the backlog was considered easy to understand and implement for developers that already have a development environment with multiple backlogs and who already work with user stories. The conclusion was that organisations using SAFe should have no problem using it. One problem identified with this part of the method was that some digital task management systems used to handle backlogs might have problems with connecting tasks between different backlogs and may lack a good way of working with stories after they are completed. The ease of use is in a high degree determined by the task management systems used and their functionality. One suggestion of an improvement to the method in regards to threat scenarios was to write them directly into the backlog and skipping the use of a separate threat scenario document of any kind.

The use of a list of security controls was considered to make sense in terms of evaluating security but the connections between threats and security controls and the categorisation were considered difficult to understand and implement. Some organisations already have lists of security controls but with a focus on best practices or compliance with regulations and it would be inefficient to implement another list on top of an already existing one so the connection between threats and their controls would need to be modified to fit existing lists. Concern was raised regarding inefficiency of managing the lists. The list could be created separately from SAFe and would thus fit well into a SAFe environment but was not considered to be very agile by the interviewees.

The process of identifying new threats during development and the tools suggested to aid in this was considered to be good but it was questioned if it was sufficient. The problem with security in agile organisations was however considered to lack other solutions by the developers interviewed. Tools like the ones proposed in this paper were considered to be the best solution currently available. The tools are not considered to fully solve the problem of identifying changes to attack surface or new threats but the tools do fit well into SAFe even if the work with keeping the threat scenarios updated might create some additional workload.
12.1 Solutions to identified problems

How well the method proposed in this thesis helped against the identified problems in Section 5.2 was evaluated by feedback provided during the interviews and during the evaluation workshop.

**Changing attack surface** in Section 5.2.1 is countered by several tools presented in this thesis. Especially the use of a list of security controls and threat scenarios that are being kept up-to-date by identifying new threats and changes to attack scenarios. The identification of new threats is in turn done using threat personas and security checklists.

This was determined to help with the identified problem but it does not fully solve it. The use of threat personas and security checklists was considered good approaches during the evaluation and would help identify some threats. The process of keeping the threat scenarios and a list of security controls updated was identified as a possible problem as it could be much work if it needs to be done manually.

**Security as overhead** in Section 5.2.2 is addressed by including security in the backlog and trying to keep security activities outside the backlog to a minimum.

This was determined to be a very good approach by the developers interviewed during the evaluation. Turning threat scenarios into abuser stories was the most appreciated example of this. How well this was achieved in other parts of the method proposed in this paper was discussed and the use of a list of security controls was considered to fit poorly into this approach as it most definitely will create extra work.

**Separation of duties** in Section 5.2.3 is addressed by the suggested use of external threat analysts and penetration testers.

Separation of duties was not really considered a problem by the people participating in the evaluation. The use of external threat analysts and penetration testers was considered to be an advantage for more practical reasons, mainly to gain another perspective of the security in the system. The current level of separation proposed in this thesis was considered enough.

**Risk owners** in Section 5.2.4 is already helped to some extent by SAFe:s original design to have all involved parties participate in planning sessions for the ARTs. By adding the threats to a common backlog that all these teams have access to it should be
easier for teams to identify threats and tasks that they themselves will need to handle. The problem is further countered by the use of a risk manager role with the job to keep the teams focused on identifying risks and assigning mitigations to appropriate teams.

To make threats visible and to show their threat level in a common backlog was considered a good approach in highlighting the current threats and threat level to the different development teams. To implement someone in the role of risk manager was also considered good as it would help to solve the problem. Participants during the evaluation pointed out that even if SAFe is being used it will often be parts of the security work that will be assigned to parties not involved in the ART, sometimes even to parties outside of the own organisation. This problem is not considered solved but is also in some extent part of the threat analysis itself and the solution is in parts outside of the scope of this thesis.

**Traceability** in Section 5.2.5 is implemented by the use of digital backlogs that connect threats to security controls using acceptance criteria and also by the use of threat scenarios and the list of security controls.

The use of the threat scenarios as security features or abuser stories in the backlog was considered good and logical and it would provide a good amount of traceability. The concerns regarding this are that some digital task management systems might lack good tools for managing the stories that are fully implemented and that these might disappear from the system. The second concern was that threats that were not part of a threat scenario would not be traceable in the same way as the threats that are part of a threat scenario but would require the separate method of the listed security controls.

**Residual risk acceptance** in Section 5.2.6 is aided by the threat scenarios and a list of security controls.

This was determined to be aided by the threat scenarios and especially by the traceability in the digital backlogs. The way to present threats as part of a bigger picture using the threat scenarios was appreciated by participants in the interviews as it would help in communicating risks with the product owner or other stakeholders. The interviewees were not able to fully evaluate how useful it would be and the workshop did not include any real stakeholders that would have to make these decisions but the general opinion was that it would aid in residual risk acceptance.

**Penetration test** in Section 5.2.7 is suggested to be done by external parties with access to source code and with a narrow scope. The problem of how to do this is not
handled in this thesis. But penetrations tests are considered a part of verification. The placement of penetration tests as a final element of the development method was appreciated by penetrations testers as it would be used to confirm that security is implemented rather than as a first tool to find vulnerabilities.

**Funding** in Section 5.2.8. The tools proposed in this thesis make it easier for product owners and stakeholders to see the risks to their organisation and what security work that has been done against what threats. This could make it easier to make decisions on funding for security.

During the interviews, it was concluded that it might help while arguing for better funding but that the people making decisions on funding often does not get to see this information.

### 13 Results and discussion

The most successful part of the method proposed in this paper was the use of threats scenarios and their backlog implementations as abuser stories. The evaluations showed this to be easy to understand and easy to use. To list security controls and to use these lists in development was harder to implement and not as easy to understand.

Many interviewees quickly focused on the part of the method where threats are put in the backlog and some completely dismissed the use of listed security controls with the motivations that it would be work done by someone else. This shows that the use of a list of security controls create extra overhead work and is already optimized out by developers during the evaluation. By only using the part of the method with threat scenarios and their backlog implementations a problem with threats that are not included in any of the threats scenarios is created. The problem is how these are monitored and how they are connected to the security controls against them if there are any. One suggestion from the evaluations was to create a threat scenario that just included all other identified threats and use that to monitor and present these threats. It was also suggested to write threat scenarios directly into the backlog and skip any documentation in between, ideally, the threat analysts should be given direct access to the backlogs so they can add these threat scenarios directly to the backlog.

The results of the evaluation support the ideas that all security work should be included in the backlog as mentioned in some related work [30]. And this could have been given more consideration during the development of the method in this thesis as it was identified as a problem in the first interviews.
Overall the method proposed in this paper reaches the goals to integrate a threat analysis into an agile development method. It does create a threat-driven environment that also in some extent covers the continuous development and threat identification.

14 Conclusions

By creating threat scenarios and adding these to the backlog as either security features or abuser stories it is possible to implement the threat analysis created by Nixu into agile organisations that are using SAFe, and possibly other agile frameworks, as part of their backlogs and task management systems. By using the shared backlogs of the SAFe program level to write the threats as stories the work and progress is visible to all parties involved in the development work. The progress and current level of mitigations against the threats is also visible as security controls are added as acceptance criteria to the threats. The product owner that can use this information to help in decisions about residual risk acceptance.

The use of the backlogs to track and manage threats can be complemented by the use of a list of implemented security controls to map threats that are not part of any threat scenarios. By keeping track of the implemented security controls you would make it much easier to see if newly identified threats are already mitigated or if new controls are needed. However, this will impose a workload that fits poorly into the agile mindset.

The direct implementation of threat scenarios as abuser stories is one of the key concepts and makes the implementation easy to understand and to work with. The importance of making security work fit into the agile environment and the backlogs that are already identified by Antti Vähä-Sipilä [29][30] is supported by the results of this thesis.

15 Future work

Further work could be done to determine how to monitor threats that are not part of the threats scenarios other than the use of a list of security controls. The idea to monitor these threats by an additional threat scenario mentioned during the evaluation could be considered and evaluated. This could be part of work regarding how to further align security work with agile development methods without creating overhead work.

Many organisations are starting to use development methods that go under the name DevOps where development and operations work closer together and where the deployment rate can be several releases per day. How to keep threat analyses up-to-date in
these kinds of projects and how to keep track of threats against your system could be studied further.
References


A Interview questions threat analysts

What are your current work assignments that are related to threat analysis?

What are your previous experience in development and/or threat analysis?

What tools and methods do you use when making your threat analysis?

How do your customers work with security before you have delivered your threat analysis?

How do the customers make use of your threat analysis?

Have you ever working with scaled agile framework?

Do the customers succeed in mitigating the threats you have identified?

Do you feel you lack any tools when you work with threat analysis?

What are the major problems you see when your customers use your threat analysis?

B Interview questions developer

What experience do you have as a developer?

What experience do you have of threat analysis and threat models?

What are your experience with threat analysis from Nixu?

Were you involved in creating the threat analysis?

Do you have experience in scaled agile framework?

What are your current work assignments that are related to threat analysis or threat models?

Can you describe your development environment, is it agile and how?

How does your organization work with security, how is it implemented in the development process?

How do you verify that the threats identified in the threat analysis are mitigated?

Do you feel anything is missing from the threat analysis you received?
C Evaluation questions

Looking at the identified problems how well would you say that each of the problems have been solved?

What problems do you see with the solutions to each problem?

What do you like about the solutions to each problem?

Do you feel that the suggested method and tools are easy to understand?

What problems do you see with the suggested changes to the threat analysis?

What benefits do you see with the suggested changes to the threat analysis?

What problems do you see with integrating the threat analysis into the agile development?

What benefits do you see from integrating the threat analysis into the agile development?