# A Framework for Detecting and Preventing Security Vulnerabilities in Continuous Integration/Continuous Delivery pipelines

---

Michael Koopman

*June 20th, 2019*
Version: Final version

University of Twente

Department of Services, Cybersecurity & Safety

Documentation

# A Framework for Detecting and Preventing Security Vulnerabilities in Continuous Integration/Continuous Delivery pipelines

Michael Koopman

*1. Reviewer* Maya Daneva
Department of Services, Cybersecurity & Safety
University of Twente

*2. Reviewer* Klaas Sikkel
Department of Services, Cybersecurity & Safety
University of Twente

*Supervisors* Maya Daneva, University of Twente
Maikel Ninaber

June 20th, 2019

**Michael Koopman**

*A Framework for Detecting and Preventing Security Vulnerabilities in Continuous Integration/Continuous Delivery pipelines*

Documentation, June 20th, 2019
Reviewers: Maya Daneva and Klaas Sikkel
Supervisors: Maya Daneva, University of Twente
Maikel Ninaber

**University of Twente**

Department of Services, Cybersecurity & Safety
Drienerlolaan 5
7522 NB Enschede

# Abstract

In a modern, agile, software development team, the goal is to get software made in a timely manner. To achieve this, these teams usually rely on tools Continuous Integration and Continuous Delivery to automate a lot of work for them. New code is automatically tested and integrated with code from other systems to check whether no new bugs are introduced, and a deployment of a new build to production can happen with the click of a button or even automatically. Each of these steps has their own tools that work together to achieve the final goal of bringing new features to production. Having so many tools does come with security risks: how do these tools work together? What data is sent from each tool to another? What would happen if an attacker took over a tool? This paper aims at delivering a framework for detecting and preventing security vulnerabilities in Continuous Integration/Continuous Delivery pipelines in the context of a large consultancy company which provides Continuous Integration/Continuous Delivery environments as a service to customers and internal development teams. Some exploratory research is done on how CI/CD is used within the company, and together with experts from the company, the framework is built. The end result is a baseline which the company can use to detect and prevent security vulnerabilities in their platform.

# Acknowledgement

Dear reader, I am proud to present to you my Master Thesis. Behind every Master Thesis is a story. There's more to it than just these words on paper. Before I started, I heard that a Master Thesis is probably the ultimate test of determination, with ups and downs. A quote from a Reddit user on /r/GetMotivated reads: *"Motivation gets you started, but discipline keeps you going.*"

For me, this means two things: The first is that I try to do most of the work as soon as possible. Early in the morning I am most productive and I expected to lose motivation for writing this Thesis over time. The second is that I always need to have something to look forward to, for example a meetup with friends in a pub or attending all different kinds of local events, but I also travel to a different country for a long weekend regularly. I would like to thank and name everyone who gave me the motivation to finish this thesis and was there for me when I needed them:

My thanks goes out to my mom, dad, brother and sister, who would always welcome me back into the home I left before starting my journey to write this Thesis.

My thanks goes out to the rest of my family and my friends from University, who I knew I could count on when I needed them.

My thanks goes out to Maya Daneva, my academic supervisor, whose positivity and support knew no bounds.

My thanks goes out to my company supervisor, who familiarized me with the company and was able to give me advice and support. We had weekly meetings to discuss progress.

My thanks goes out to the Secure Development Coordinator, who supported me with his knowledge during my time at the company, and all other colleagues.

Now some unconventional, but in my opinion needed thanks:

My thanks goes out to my running shoes, which I used periodically to give me the energy I needed to get through each day, no matter how difficult it was.

My thanks goes out to the Ingress Enlightened Achterhoek, Ingress Enlightened Twente, Ingress Enlightened Het Gooi, Ingress Enlightened Utrecht, Ingress Enlightened Amersfoort, Ingress Enlightened Athens, Ingress Enlightened Bristol, Ingress Enlightened Netherlands communities and Niantic for providing an experience that is “more than a game". During my Master Thesis, I met up with these communities through social drinks and events. This allowed me to get my mind off work and stay relaxed throughout the Thesis.

My thanks goes out to my landlord, who has provided me with a positive learning experience on living on my own for the first time and had the patience to deal with whatever I had to improve.

My thanks goes out to Samsung for creating the Galaxy Note 9. Using this phone, I was able to finish the Thesis nearly paperless by taking notes on the phone. I care quite a bit about the environment, and going paperless is one of the goals I have for myself. The battery capacity got me through each day, which would not have been possible with my old phone.

My thanks goes out to the public transport in the Netherlands. Even though the NS and other transport companies regularly had delays, they always got me where I needed to be.

My thanks goes out to members of my graduation committee for taking the time to read this work and raising points of discussion during my defence.

Last, but not least, my thanks goes out to you, the reader, for reading this work. Without you, the knowledge contained in this work would eventually be lost.

Each of these entities contributed to the success of this thesis in some way and I personally feel need to be thanked.

# Contents

# 1 Introduction

In a modern, agile, software development team, the goal is to get software made in a timely manner. For this, they usually use one to two week sprints. These sprints have certain goals to them: implement a specific set of features or fix a specific set of bugs, all while not introducing new bugs.

To aid developers in this process, a modern team uses continuous integration to ensure their code is still up to standards after they push new code to their version control system. Microsoft defines Continuous Integration (CI) as "*the process of automating the build and testing of code every time a team member commits changes to version control*". [24] CI encourages developers to share their code and unit tests by merging their changes into a shared version control repository after every small task completion. Committing code triggers an automated build system to grab the latest code from the shared repository and to build, test, and validate the full master branch (also known as the trunk or main)."

To automate the deployment of a build from development to testing to production, developers use Continuous Delivery (CD). Microsoft defines Continuous Delivery as "*the process to build, test, configure and deploy from a build to a production environment. Multiple testing or staging environments create a Release Pipeline to automate the creation of infrastructure and deployment of a new.*" [23] This release pipeline ensures that the new build is put into the production environment.

The relation between Continuous Integration (CI) and Continuous Delivery (CD) is that Continuous Integration starts the CD process. In case the CI pipeline is successful, CD can be triggered to deliver a build to some test environment. In case the build passes the test environment, CD can be triggered to bring the build to production.

The benefits for using CI/CD are that it improves the quality of software, reduces developer effort and reduces cost. It also gives an overview about build, test and deploy times, and logs of builds. [40] [1] [5] It usually consists of a link between tools (a common example is a repository on Github with Jenkins CI which is pushed to Amazon Web Services (AWS)). There is a lot of freedom in tools and configurations

when using CI/CD pipelines. This results in many different configurations across development teams.

CI/CD has become part of the software supply chain [3] [40]. Securing this supply chain is becoming more and more important. CI is for example used to check whether code is up to standards. In a scenario in which these checks are silently disabled (or made to always pass) and malicious code gets pushed, for example, it is very difficult to detect this malicious code before it is live in production. The last piece of the supply chain is the continuous delivery pipeline. It accesses many portions of a system from different sources, each with security mechanisms introduced by different people at different times. In many cases, an organization might not have a complete knowledge of these mechanisms. In case any of these mechanisms get compromised, the impact could be significant.

As seen from the definitions of CI/CD, their importance and their integration in the supply chain, security in CI/CD is a growing concern. To aid developers in systematically securing their pipelines, a framework could be used. In previous research [20], it was found that there are frameworks which could aid developers in this, but that those frameworks are either too generic or too specific to be applied efficiently. In this master thesis, a framework will be designed and validated which aims at being useful in aiding developers to secure their pipelines.

## 1.1 Motivation and context for this research

Now that an introduction has been given, the motivation for this research project and its context follows. According to a survey by DigitalOcean, 42% of respondents [28] use Continuous Integration/Continuous Deployment in their workflow. This is expected to grow in the future, since 38% of the respondents who didn't use CI/CD said they plan to use it in the future. With more and more teams switching from traditional workflows to CI/CD, security is becoming more and more important.

As seen in the introduction, CI/CD has major benefits. Improving the quality of software, reducing developer effort and reducing cost is something every organisation wants. When the question "What is your greatest concern?" was asked at a conference related to release engineering in 2014, the response was "someone subverting our deployment pipeline". [3] This fear is justified. Once CI/CD is implemented, a disruption in a pipeline could lead to a loss in productivity and delays in releases. We therefore need to ensure that the odds of these disruptions happening are minimal, and if they happen, that we minimize the impact and recovery time.

Implementing security for CI/CD might seem not too difficult at first, but it is very easy to overlook something. Since many systems are linked together to make CI/CD happen, a leak in one system can be used to gain access to another system. To prevent overlooking something, a security framework could be used during integration of CI/CD and while it's running.

During previous research [20], currently existing frameworks were found, but they were either too generic or too specific to be directly applicable to CI/CD pipelines. This is the one of the main reasons on creating a new framework. It is important that this framework is neither too generic nor too specific. To ensure this framework meets those criteria, interviews will be held with experts from the company to determine a scope that works for them. Using this well-defined scope, the experts can apply this framework to CI/CD pipelines to ensure it is secure and compliant.

Another reason is traceable to the context in which this research project is executed. This research takes place at a large consulting company providing managed security services to client organizations worldwide. The company is well aware of the importance of employing a framework for securing CI/CD pipelines. Several people within the organization have been interviewed and have confirmed that the current frameworks are not sufficient for their needs and that a new framework would be beneficial to them. [20]

Now that the motivation behind this paper is clear, the structure of the thesis will follow.

## 1.2 Thesis structure

The structure of this thesis is as follows: Chapter 2 provides related work regarding CI/CD, DevOps, agile development and other topics relevant to this paper. Chapter 3 describes how the research goal was formulated and what research questions this thesis answers. Chapter 4 describes the methods that will be used to answer the research questions. Chapter 5 describes the results from applying the methods. Chapter 6 discusses these results. Lastly, Chapter 7 provides the conclusions of this research.

# 2 Related Work

In this chapter, related work on the topics of Continuous Integration, Continuous Delivery, DevOps, security and working in an agile environment will be discussed. As part of preparing this thesis, we searched Scopus for security-related research related to the topic of the paper. In the Introduction, we provided definitions of CI and CD. Here, we introduce two other terms related to CI/CD: DevOps and agile project delivery methods. For clarity, in this thesis we refer to DevOps as a concept meaning "*a development methodology aimed at bridging the gap between Development and Operations, emphasizing communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilizing a set of development practices*", as defined by Jabbari et al. [16]. We refer to agile software development as a concept that revolves around "The use of light-but-sufficient rules of project behavior and the use of human- and communication-oriented roles.", as cited from Cockburn et al. [6]. In agile software development, the concept of DevOps can be applied as a basis for a list of rules to help achieve light-but-sufficient rules of project behavior and the use of human- and communication-oriented roles. CI/CD are the digital tools that can help ease and enforce that process.

Fowler et al. [9] summarize the technique and the current usage of CI very well. It teaches beginners to CI what it is, what it can do and what its benefits are. It also gives ideas on how to implement CI in an organization. It is a great article to read for people who want to understand what CI is exactly. This thesis took the article of these authors as a starting point.

Hilton et al. [14] present a qualitative study of the barriers and needs developers face when using CI. They use semi-structured interviews with developers from different industries and different development scales. This study is a great read to get insight in how developers experience using CI and what barriers they face while using it.

Humble et al. [15] explain how configuration management, automated testing, continuous integration and deployment, data management, environment management, and release management can be brought together as a whole. This book is a great start for people who want to gain more knowledge about how to successfully apply Continuous Delivery in an organisation. Our work draws on this textbook.

Rodriguez et al. [29] classify and analyse the literature related to continuous deployment in the software domain in order to scope the phenomenon, provide an overview of the state-of-the-art, investigate the scientific evidence in the reported results and identify areas suitable for further research. This study is a great read to get up-to-date on the latest developments in the area of continous deployment.

Shahin at al. [32] systematically review the state of the art of continuous practices to classify approaches and tools, identifies challenges and practices in this regard. It provides a very clear distinction between continous integration, continuous delivery and continous deployment, as seen in Figure 2.1. As seen in this figure, continous integration is defined as the process of building and testing automatically. Continous delivery is the manual process to release a build to Production after a successful Acceptance Test. In Continous deployment, the process of releasing a build to production is automated.

Villamizar et al. [41] characterize the publication landscape of approaches that handle security requirements in agile software projects. They do so by conducting a systematic mapping to outline relevant work and contemporary gaps for future research. They conclude that their analysis suggests that more effort needs to be invested into empirically evaluating the existing approaches and that there is an avenue for future research in the direction of mitigating the identified limitations.

Daneva et al. [7] report on results from a documentary study initiated to understand the agile-ready security practices that organizations use. They conclude that Security RE (requirements engineering) adds up to the documentation in an agile project, as teams introduce new story types, e.g. evil user stories, abuser stories, security stories. Plus, they found that Security RE relies on investments into the security training of the agile project teams and into organizing hack sessions. Last, if companies take security requirements seriously, it seems that they should consider ignoring the gatekeeping role of the agile product owner.

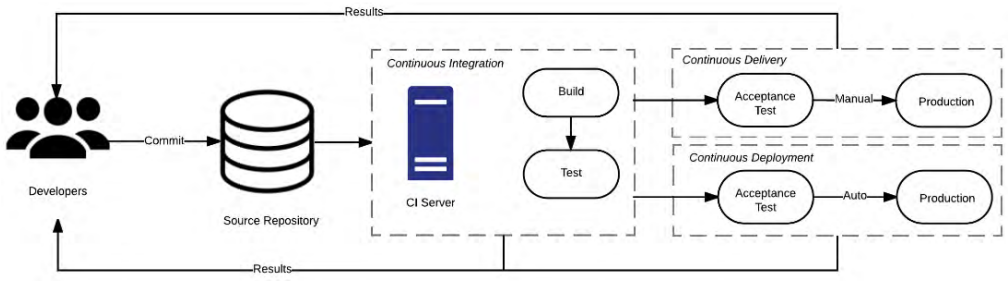


**Fig. 2.1.:** The relationship between continous integation, delivery and deployment.

Bass et al. [3] put the problem of securing a development pipeline into perspective and give concrete solutions for securing a specific pipeline used in their environment.

Security is achieved by having trusted components mediate access to sensitive portions of the pipeline from other components, which can remain untrusted. We us

Ullah et al. [40] analyze the effectiveness of 5 security tactics on CD pipelines. A non-secured pipeline and a secured pipeline are tested both qualitative using assurance cases with goal-structured notations and qualitative by the use of penetration tools. Using these security tactics improves the security of the CD pipeline by controlling access to the components and establishing secure connections.

Jabbari et al. [16] published a literature study focusing on DevOps. Since DevOps is a vague concept, the study attempts to give it a definition, associates practices with the concepts of DevOps and identifies similarities and differences with other development methods. These authors define DevOps as "*a development methodology aimed at bridging the gap between Development and Operations, emphasizing communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilizing a set of development practices*", which the author believes summarizes the concept of DevOps very well.

Cockburn et al. [6] introduce the concept of agile software development. In traditional software development methods, communication between team members is not optimal. The use of light-but-sufficient rules of project behavior and the use of human- and communication-oriented roles lay at the heart of agile software development. Using these rules and roles a development team can become more efficient at developing software than by using traditional methods.

Now that some context has been given about CI/CD and DevOps in the form of related work, the research goal will follow in Chapter 3.

# 3 Research Goal

Since the framework that follows from this research is designed to be used by employees of the company, the research goal was designed together with them to suit their needs.

## 3.1 Research Goal Formulation

In the interview with the Secure Development Coordinator, the goal was proposed by the researcher to be *“The goal of this research is to develop and validate a framework that aims at preventing and detecting security vulnerabilities in Continuous Integration/Continuous Deployment pipelines for software development projects that are made within the company and for external customers who wish to check the security of their own Continuous Integration/Continuous Deployment pipelines.*". The Secure Development Coordinator requested that this was e-mailed to him, and in the reply he stated that it is well-described, but that the term “framework" is vague, but that it can be made clearer some other time.

In the interview with the Project Manager and Agile Coach at the company, the research goal was kept the same, but some interpunction was added to make it more readable. The Project Manager and Agile Coach told that there are three situations in which this framework would be applicable. The company provides a "Production Line" to customers, which is basically a CI/CD environment. The customer can choose that the company manages this pipeline (“managed") or that they manage it themselves (“unmanaged"). It is also seen that it would be useful to apply the framework to CI/CD environments at customers where employees work which use their own CI/CD environment (“external"). The research goal is updated to reflect these environments: *“The goal of this research is to develop and validate a framework, that aims at preventing and detecting security vulnerabilities in Continuous Integration/Continuous Delivery pipelines in managed and unmanaged CI/CD pipelines provided by the company and external CI/CD pipelines created and used by customers of the company.*"

Since developing a framework for three separate environments is very likely significantly more work than developing a framework for one environment, it is decided to

limit the scope to just the managed environments. This ensures that the framework is the most useful, since this is the most predictable environment, and that the framework will be delivered in time. The final research goal is updated to reflect this: *"The goal of this research is to develop and validate a framework, that aims at preventing and detecting security vulnerabilities in Continuous Integration/Continuous Delivery pipelines in managed Production Line environments provided by the company to customers."*

## 3.2 Research Questions

The research questions which support this research goal are:

1. *What types of risks are companies using the managed Production Line exposed to?*
2. *Which practices exist to mitigate each type of risk?*
3. *How should these risks be mitigated for each risk level?*

The answer to the first question is needed in order to understand the problems that companies experience while using managed CI/CD. We would like to know what types of risks organizations are exposed to and what risk levels there are. The second research question will help us understand the solutions in terms of practices that mitigate each type of risks identified in the answer to the first research question. Matching the practices to the risks will help us understand any gaps that are possibly existing, for example it might be the case that there are no practices that match a particular risk type. In turn, understanding these gaps will lead us to formulating goals for the risk mitigation framework that will be developed in this thesis. Our framework is the answer to the third research question.

The methodology to answer each research question will be discussed in the next chapter.

# 4 Methods

## 4.1 Research method

The main method that will be used for this research is Wieringa's Design Science Methodology [42]. We chose this method for the following reasons:

1. It fits in contexts of industry-relevant research and this is what this master thesis includes.

2. It is suitable for situations in which a researcher is embedded in a business organization and can closely observe problems and issues as practitioners experience them.

3. It is suitable for research which can not be replicated in academic environments.

Wieringa's method is summarized in figure 4.1.

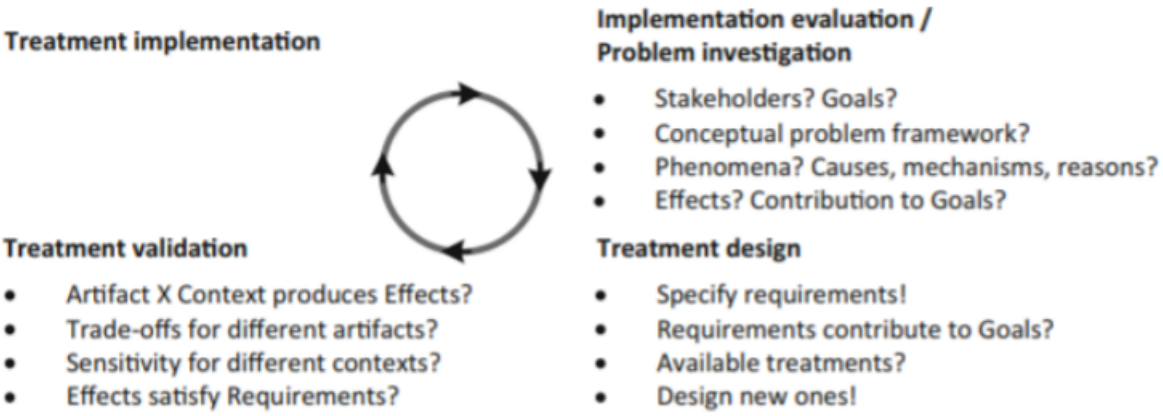


**Fig. 4.1.:** Visual representation of Wieringa's Design Science Methodology

The design cycle starts out with a Problem Investigation. This phase aims at determining what problem needs to be solved. The problem investigation starts with the Stakeholder and Goal Analysis. Wieringa defines a stakeholder as "a person, group of persons, or institution who is affected by treating the problem". These stakeholders have goals in mind that they wish to achieve to treat the problem,

the so called *Stakeholder Goals*. However, the desires of the stakeholders and the Stakeholder Goals may conflict. These conflicts might be solvable by technical means or increasing the budget, but others might not be solvable, due to legal or moral reasons.

After the Stakeholder and Goal Analysis, the Implementation Evaluation and Problem Investigation stage begins. In case there is already an implementation, either from a previous cycle or an existing solution, this implementation is evaluated with respect to the stakeholder goals.

Problem investigation consists of investigating real-world problems to prepare for designing a treatment for the problem. The goal of this is to learn about Stakeholder Goals and to understand the problem that needs to be treated.

After the Implementation Evaluation and Problem Investigation, the Requirements Specification stage begins. Requirements are treatment goals, which are desired by some stakeholder who has committed a budget to realize them. They are specified by the design researcher. Each requirement must, under the assumptions made in the context of the project, contribute to the stakeholder goal.

After the requirements are clear, it is time to implement these in such a way that they can be validated to provide the effect they were designed for. If this turns out to be the case, the artifact could be considered “done". If there is feedback or the artifact behaves different than expected, this is taken into account and the design cycle starts anew.

Now that the general idea of Wieringa’s Methodology for Design Science is clear, it will now be discussed how it is applied in this research.

## 4.2 Application of the Research Method

The first research question is: “*What types of risks are companies using the managed Production Line exposed to?*". To answer this question, the author of this thesis will first have to understand what the Production Line is, what it is used for and how it is used. To achieve this goal, interviews have been done with experts on the topic of the Production Line, and the author has setup a sample project on their own Production Line environment to get a hands-on feel with the Production Line.

Once the author understands what the Production Line is, what it is used for and how it is used, a Risk Analysis is done. For this, first a Threat Model is made based

on several sources, namely literature research, a webinar, lessons learned from past incidents, and another small research about CI/CD security done by my supervisor. Experts are asked to validate this threat model and add to it as they seem fit.

Based on this threat model, risks and controls for these risks are identified. This answers the first and second research question. To repeat, the first research question is "*What types of risks are companies using the managed Production Line exposed to?*". The second research question is "*Which practices exist to mitigate each type of risk?*".

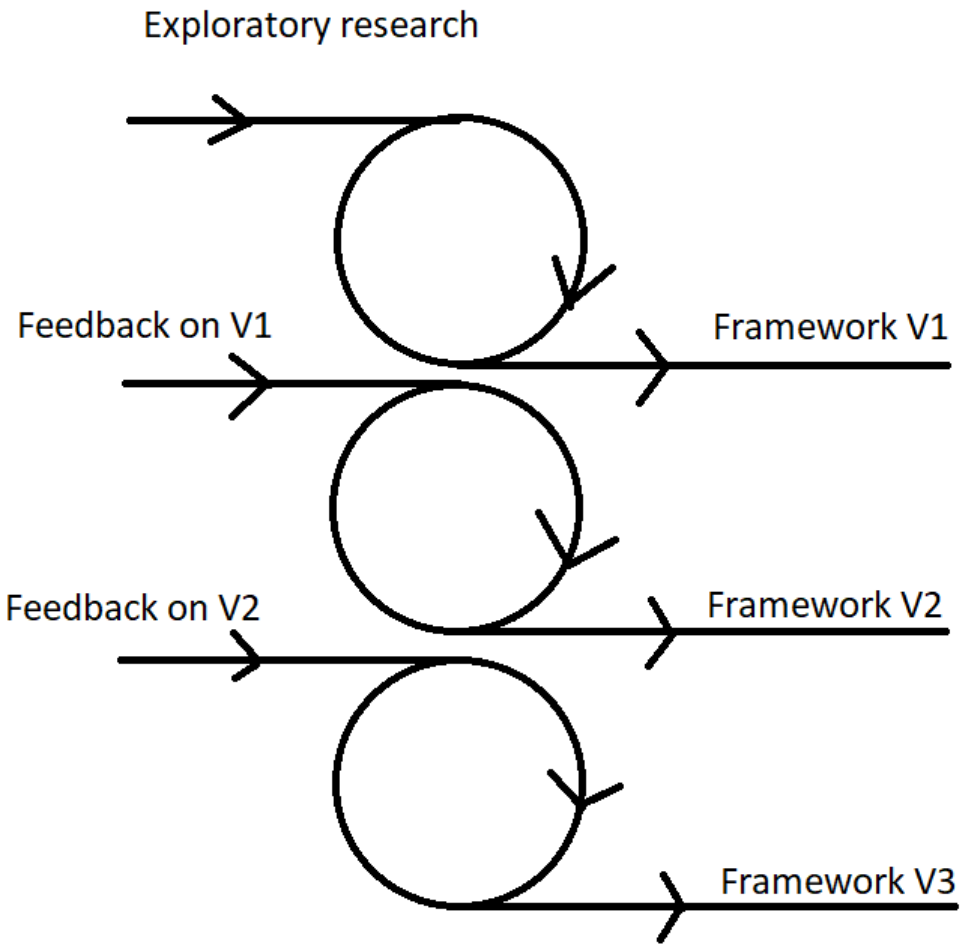


**Fig. 4.2.:** This research performed three iterations of the Design Cycle

Once this is complete, exploratory research on which risk levels are assigned to which kind of threats in the company will be performed. The author will then propose risk levels to the controls for the threats that were identified previously. This is version 1 of the framework. The framework will be sent to the Secure Development Coordinator, an expert on the topic of frameworks, for initial feedback on whether the framework is going in the right direction.

Once their feedback has been taken into account, version 2 of the framework will be delivered. Together with experts, a group evaluation on the framework itself and on which controls have to be applied at which risk level will be performed. Based on this, the framework, the threats, controls and risk levels will be validated or improved.

Version 3 of the framework consists of an extension to the the current framework, consisting of an Excel sheet with a checklist. It will be extended to include controls specifically for projects which use the Production Line and to clarify how to apply existing checks to a Production Line environment. This extension will be evaluated by some experts on frameworks. This answers the third research question: "*How should these risks be mitigated for each risk level?*".

This method will result in a useful framework for preventing and detecting security vulnerabilities in managed Production Line environments.

# 5 Results

In this section, the results from applying the Design Science research method as seen in the previous section will be discussed.

## 5.1 Getting familiar with the Production Line

In the context of this research, the term Production Line plays an important role. We define it as follows: A Production Line is a set of server-side collaboration tools for engagements of the business. It has been developed for supporting project engagements with individual tools like issue tracking, continuous integration, continuous deployment, documentation, binary storage and much more.

To get familiar with the production line, interviews were conducted with four experts and a sample project was built. They have been summarized in the next subsection. In the subsection after that, the process of building the sample project will be described.

The interview process included four people. They were chosen because of their expertise, their organisational roles and their professional interest in this research.

The first interview was held with the Secure Development Coordinator. This person is important in this research, as their knowledge on existing frameworks within the company and feedback on my versions of the new framework are used to improve each version of the framework. This person was also interviewed by me during previous research [20]. On each version of the framework, some feedback is given by this person and taken into consideration. This interview was aimed at determining stakeholders and what this person wants from the new framework.

The second interview was held with a Project Manager and Agile Coach at the company. This person was recommended to interview by the Secure Development Coordinator in the previous interview, because they are an expert on the topic of how the company uses Continuous Integration and Continuous Deployment. The goals of the interview were to understand how and when CI/CD is applied by the company, understand which tools are used for CI/CD by the company, understand which

measures are currently being taken to secure CI/CD environments and potentially determine a scope for the framework to be designed.

The third interview was held with an Engagement Security and Privacy Manager, who is responsible that the teams working on projects use the frameworks and that compliance is checked on a regular basis. This person was also recommended by the Secure Development Coordinator, since the Engagement Security and Privacy Manager told the Secure Development Coordinator that they would like to have some security baselines regarding CI/CD, DevOps streets and development at some point in the past. The goal of this interview was to find out what they do on a daily basis, what their responsibilities are and which frameworks they currently use.

The fourth interview was held with the previously mentioned Secure Development Coordinator and the Security Manager of Delivery in the Netherlands. The Security Manager is responsible for that anything that is delivered adheres to the ISO standards and that every Engagement Manager or Service Delivery Manager, who are responsible for a contract, can deliver the contract according to these specifications, including security. The goal of this interview was to determine the current state of the security of CI/CD pipelines, how the new framework could be useful to them and to identify experts on the area of CI/CD which I could potentially interview in the future.

After the fourth interview, it was decided that enough information was gathered for the purpose of this research, and no further interviews were held until after the delivery of the first version of the framework. The author of this research was provided with an instance of the company's CI/CD solution, and used this to build a sample project to learn how each of the tools in the solution work and how they work together. The sample project consisted of uploading some code to the version control system, making new code build automatically and using a static code analyzer to analyze the quality of the code.

### 5.1.1 Interviews with experts

#### Secure Development Coordinator about stakeholders and what they want from the framework

In the first interview with the Secure Development Coordinator, an expert on the currently existing framework, the goals were to identify what they want from the new framework and to understand which people would be affected by the framework in different use cases. The Secure Development Coordinator was also interviewed

in previous research to get a feel of how CI/CD is used in the company, how CI/CD pipelines are secured within the company, what the limitations are of the method used to secure CI/CD pipelines, and how to overcome these [20].

The interview starts off with a stakeholder analysis. The questions that are asked are in line with the stakeholder analysis method as Wieringa describes in their book [42]. Assessors and the Account Manager or Service Delivery Manager are identified as Normal Operators. Assessors get appointed by e.g. the Service Delivery Manager to check the checklist. The Assessor is responsible that everything on the checklist gets checked off, but the Service Delivery Manager or the Account Manager are accountable if something goes wrong.

Another Normal Operator that is identified is the people from the Security Organization, who will check (audit) that the Assessor has correctly checked off the checklist. Firstly, every contract has an Engagement Security Manager, who validates such assessments. If the Engagement Security Manager has questions about this, a Subject Matter Expert will decide.

When asked which other users could fall under “Normal Operators", the interviewee mentions the person(s) setting up the DevOps street (Infra), and the Chief Security Officer (CSO). Since the framework applies to both development and shaping a DevOps street, Technical Application management and Operators are also interested. They will need to keep the environment up and running.

We then try to identify which stakeholders are considered Maintenance Operators. The Chief Security Officer is ultimately responsible that the framework is updated once it needs to be updated. These frameworks get could get approved by NL, EU and/or Global branches of the company.

When talking about stakeholders in the category of Operational Support, the interviewee mentions that the company is currently setting up training to make the current baselines more clear. The Trainers will be responsible for explaining the framework to employees who don’t know the framework.

When it is unclear whether something is compliant, several people could give clarification, e.g. the Secure Development Coordinator or the Engagement Security Manager. As part of the training program, Champions will be trained, who are experts on a certain topic. A champion could be trained on the topic of DevOps security.

When talking about Functional Beneficiaries, the Engagement Security Manager will benefit from being compliant with the framework. If the guidelines are followed,

the risk is addressed and you have no more unknown vulnerabilities in your system. The customer will also benefit from less incidents.

Other entities affected by the new framework (Interfacing Systems) include NL's branch of the company's own Security Baselines or Information Security Management Systems (ISMS). There is a preference that the framework is an extension to the currently existing Security Baselines.

In terms of the Wider Environment, there are also negative stakeholders. The Assessor might think: "Yet another framework that I have to check...". The Service Delivery Manager might also not like it for the same reason as the Assessor. Furthermore, all stakeholders could hurt the development, introduction and/or use of the framework. If they say "no", the framework simply won't be adopted. It is therefore very important to involve as many stakeholders as possible during the development of the framework.

After the interview with the Secure Development Coordinator, an interview with a Project Manager and Agile Coach follows. It is important to mention that the Research Goal was created together with the stakeholders to ensure that as many stakeholders as possible stand behind the research goal and a "no" as mentioned in the previous paragraph is prevented as much as possible.

### Project Manager and Agile Coach about the application of CI/CD by the company, CI/CD security and stakeholders

The goals of the interview were to understand how and when CI/CD is applied by the company, understand which tools are used for CI/CD by the company, understand which measures are currently being taken to secure CI/CD environments and potentially determine a scope for the framework to be designed.

There are two kinds of teams which use CI/CD within the company. The first are development teams, in case of new development. CI/CD is used very limited there. The second are administration teams, who have taken over applications from customers and are using the Production Line to do their deliveries. CI/CD is used there to optimize the development process.

Several tools are used for CI/CD, about which the author will later go in more detail about.

There is a team responsible for ensuring the Production Line works and for its security. This team is divided over Morocco and India and consists of around 20 people. They administer the environments, create new ones, update them, add new tools to the environment and do everything else related to these environments. There are hundreds of Production Line instances used by the company.

The Production Line is ISO 27001 compliant, which is great, but also comes with some limitations. It cannot be accessible from the outside world. It would be beneficial if it could be accessible from a customer's environment, which is currently not possible for security reasons.

Currently, there is no method specifically designed for the security of CI/CD pipelines as far as the interviewee knows. They feel like ISO is too generic to apply effectively to the Production Line.

The interview continues with determining a scope, which was discussed in 3. Then, a stakeholder analysis follows. Only new stakeholders not identified during the previous interview will be discussed.

The only relevant new stakeholder identified falls under Normal Operators and are the Testers of the application. A penetration tester could use the framework during their tests to find possible security issues.

**Engagement Security and Privacy Manager on what they do on a daily basis, what their responsibilities are and which he frameworks currently uses**

Their responsibilities are to make sure that the teams working on projects fill out their forms related to security. If the teams check a checkbox on the form, they can assume it is done. They also have to make sure that these forms are resubmitted periodically. When new projects are obtained, the forms are filled in for the first time. Sometimes, a customer doesn't want to do one of the things on the form, because it costs too much money, e.g. Threat Analysis. In this case, higher management has to sign off that they accept this risk and that they themselves are responsible for anything that happens because of it. When something does happen though, usually the company is the party who looks bad, because they should have prevented it, even though the customer signed a waiver accepting the risks.

### Security Manager and Secure Development Coordinator on the current state of the security of CI/CD pipelines, how the new framework could be useful to them and experts on the area of CI/CD

The goals of this interview were to get to know the Security Manager, determine the current state of the security of CI/CD pipelines, determine how the new framework could be useful for them, determine some risks for the use of the Production Line and identify experts on the area of CI/CD.

After providing some background, the Security Manager is asked what their responsibilities within the company are. They are responsible for that anything that is delivered adheres to ISO standards. This is implemented based on the baselines. They also ensures that every Engagement Manager or Service Delivery Manager or Service Coordinator, who are responsible for the contract, can deliver the contract according to the specifications, including security. They assist them in this and reports to Delivery Heads and higher management about the progress in this. They are responsible that the Engagement Manager, Service Delivery Manager or Service Coordinator know what they have to do, but in the end, they will have to do it themselves.

Currently, the Security Manager relies on the company France to correctly apply the Secure Development guidelines to the Production Line. They should, since they are certified for this framework. The Secure Development Coordinator mentions that the Production Line can be viewed as an application itself and therefore these Secure Development guidelines can be directly applied to them. Methods like Threat Analysis, Vulnerability Assessment can be applied directly to them.

The Security Manager doesn't really know of any past security incidents on Managed Production Line instances. If the platform is correctly setup, many problems are avoided. When Unmanaged Production Line instances are used, there is more risk. It has happened that the antivirus was not turned on during setup, resulting in ransomware on the machine. The Secure Development Coordinator does mention that it has happened that in the case of Managed Production Lines, source code was leaked to entities of the company that were not supposed to have it, e.g. the Indian branch of the company got hold of code that couldn't be distributed outside of NL or EU.

The Security Manager does see a need for this framework, since the current framework is more aimed at the traditional way of development. More and more technology keeps getting used nowadays. They mention Docker as an example. Docker containers are so diverse that it it's not possible to develop a standard checklist

for them. Of course there could be a high-level checklist, but it would not cover everything.

The new framework will be an addition to the baselines the Security Manager aligns. They look at which baselines are applicable to which projects. The new framework is expected to fit more with activities people are working on.

Both interviewees agree that an addition to the Excel sheet is a desirable method for making this new framework. Making a new method not based on Excel sheets is outside of the scope of this project and not possible in time.

Some advice both interviewees gave was to know your audience, and make the framework understandable to users of all levels, junior, medior and senior. Not everyone is familiar with the subjects these checklists are about.

A risk analysis is attempted, but it is quickly determined that an interview is not the best method of doing a risk analysis. The method will be changed according to this experience.

Both interviewees provided me with some contacts in various topics related to my research. The interview concludes.

### 5.1.2 Sample project

In this section, some information about the sample project that was created in the Production Line will be provided. This sample project was created with the aim of getting an idea of what the Production Line can do and how it is used. It is not aimed at becoming an expert on the Production Line, since there is not enough time for that and is not needed for this research.

The company provided some documentation in their Knowledge Management System on the use of the Production Line. This included, among other things, a Getting Started guide and policies. I downloaded the Pet Clinic sample project [38], a commonly used web development framework for Java [26]. The reason I downloaded this example is that it was easily findable on Google and that it uses Maven [2], which integrates really well with one of the tools used in the Production Line (Jenkins [18]).

The tools that are in the scope of this project and offered within the Production Line are Jenkins [18], SonarQUBE [35], LDAP Account Manager [22], Selenium [31], Graylog [12], Nexus3 [36], Grafana [21] and Gitlab [11]. Each tool that was

not already familiar to the researcher was used to get a basic understanding of the functionality.

Firstly, the source code of the example project was imported into Gitlab. Then, Jenkins was linked to Gitlab and builds were made. SonarQUBE was linked to Jenkins to show code quality for every build, and it was verified that it did that. It was verified that logs showed up in Graylog. During the process, users were made with LDAP Account Manager were necessary. The researcher already had experience with Selenium, Nexus3 and Grafana, so after looking at the tools it was decided that it was understood how these integrate and no further action was taken.

This sample project gave the author enough information on the use of the Production Line to understand it well enough. Together with the interviews, enough information was collected about how the Production Line is used and what it can do. The threat model will now follow.

## 5.2 Risk analysis

In this section, the risk analysis on the Production Line will be discussed. A threat model is made, risks are identified and controls are suggested to keep these risks acceptable for each risk level. The process is visualized in the following picture:

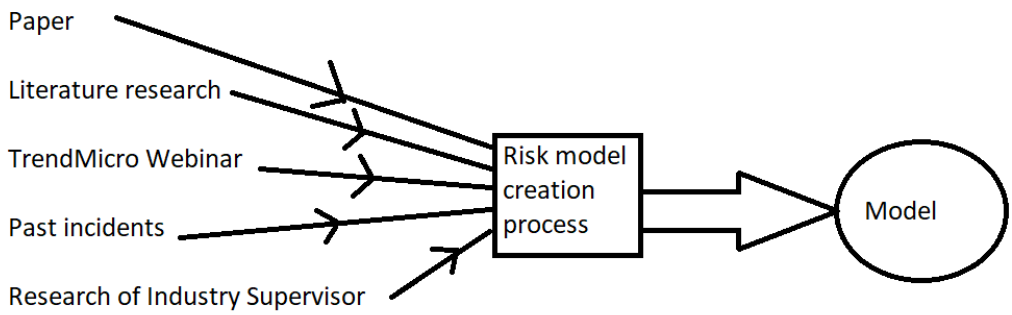


**Fig. 5.1.:** Visualization of the risk analysis process

### 5.2.1 Sources for threat model

The threat model will be created based on five sources. The first is the paper “Securing a Deployment Pipeline" [3], which describes 3 threats to CI/CD pipelines. The second is literature research on how some of the current suppliers of CI/CD pipelines actively help to protect against threats. The third is information shared during the TrendMicro Webinar on the security of CI/CD pipelines that happened on the 27th of March 2019 [39]. Fourthly, lessons learned from past incidents will also

identify threats to the CI/CD pipelines. Finally, a research of my Industry Supervisor at another company will provide more insights in CI/CD security in practice, from which can be learned.

Below, we summarize the ideas in these sources and explain why we included them in the development of our frameworks.

### Source 1: Paper “Securing a Deployment Pipeline"

The paper “Securing a Deployment Pipeline" [3] mentions three threats to the CI/CD pipelines. What follows is a direct citation from this paper:

1. A remote attacker attempting to exploit a component in the build environment that is directly accessible from outside of the environment. If successful, an attacker can gain the privileges of the process. We do not consider further privilege escalation (to administrative rights) - this would trivially compromise all processes on the machine.

2. In the spirit of the authors of [27], a remote attacker attempting to exploit a component indirectly and without direct network access to the build environment. This can be done by a seemingly benign and unnoticeable change in a third-party repository, with code fetched from there as part of the build process. The actual code is not malicious itself for the third-party but does introduce an exploit which can, as above, allow a compromise of the entire process on the machine, with new privileges for the attacker.

3. Attacks on the network links on the public Internet, i.e. on the connections between our machine, third-party repositories, Deployer, storage, and cloud.

Each of these are interesting and relevant to include in the threat model, since they are direct threats to CI/CD pipelines. They will therefore be included.

### Source 2: How do current suppliers of CI/CD pipelines actively help to protect against threats?

**Gitlab**

Gitlab provides CI/CD tooling in their solution. In the Production Line environment, this tooling is disabled in favor of using Jenkins. It is nonetheless interesting to look at how Gitlab protects against current threats.

Gitlab applies DevSecOps to integrate security best practices in the DevOps workflow [10]. Traditionally, it was difficult to balance business velocity with security. DevSecOps solves this problem by building this architecture into the CI/CD pipeline. Gitlab mentions the following advantages for DevSecOps:

- Every piece of code is tested upon commit, without incremental cost.
- The developer can remediate now, while they are still working in that code, or create an issue with one click.
- The dashboard for the security pro is a roll-up of vulnerabilities remaining that the developer did not resolve on their own.
- Vulnerabilities can be efficiently captured as a by-product of software development.
- A single tool also reduces cost over the approach to buy, integrate and maintain point solutions.

The concrete features Gitlab offers for DevSecOps are as follows:

**1. Static Application Security Testing (SAST)**
Prevents vulnerabilities early in the development process, allowing to be fixed before deployment.

**2. Dynamic Application Security Testing (DAST)**
Once code is deployed, prevent exposure to your application from a new set of possible attacks as you are running your web applications.

**3. Dependency Scanning**
Automatically finds security vulnerabilities in your dependencies while you are developing and testing your applications, such as when you are using an external (open source) library with known vulnerabilities.

**4. Container Scanning**
Gitlab can analyze your container images for known vulnerabilities.

**5. Auto Remediation**
Auto remediation aims to automated vulnerability solution flow, and automatically create a fix. The fix is then tested, and if it passes all the tests already defined for the application, it is deployed to production.

**6. Secret Detection, IAST and Fuzzing**
Future features GitLab will be adding to its Security capabilities.

In conclusion, GitLab offers several features to help customers protect their CI/CD pipelines.

**Jenkins**
Jenkins provides a Wiki page with some information on how to secure your Jenkins instance. [19] This Wiki page mentions 2 aspects, namely Access Control and Protecting users of Jenkins from other threats. We will briefly elaborate on these aspects.

**1. Access Control**
To provide the right users the right access at the right time for the right reasons. An example is that every logged-in user has all permissions, or that specific users have specific rights. This is controlled by 2 axes: the Security Realm and the Authorization Strategy.

The Security Realm determines users and their passwords, as well as what groups the users belong to. The Authorization Strategy determines who has access to what.

**2. Protecting users of Jenkins from other threats**
Jenkins also provides some features to protect users from other threats. They are turned off by default, but should be considered to be turned on after reading the documentation. These features are:

- CSRF Protection, which prevents a remote attack against Jenkins running inside of a firewall.
- Building on master. Jenkins master should be protected from malicious builds.
- Slave to Master Access Control, which is used to protect Jenkins master from malicious build agents.
- Securing JENKINS_HOME, which protects Jenkins from users with local access.

There are also some features which are turned *on* by default, which are Content Security Policies, which protects users of Jenkins from malicious builds and Markup formatting, which protects users of Jenkins from malicious users of Jenkins.

### Source 3: TrendMicro webinar on the security of CI/CD pipelines

On Wednesday the 27th of March 2019 TrendMicro held a webinar with the topic "*Securing Containers and your CI/CD pipeline without friction*". The topic was relevant to this paper, so it was decided to attend this webinar. Although the presentation was given by sales people with the intent of providing customers information on how TrendMicro can help protect their CI/CD pipelines, the presentation did include relevant information on threats TrendMicro is protecting customers from. The presentation named 7 ways TrendMicro can help protect customers, and 6 of them were relevant for inclusion:

**1. Build Scanning**
TrendMicro provides customers with a Jenkins plugin which can be used to scan builds for viruses. When a virus is found, it will not be possible to deploy the build. A limitation of this method is that a virus is only stopped if it is within TrendMicro's signature database or TrendMicro's machine learning algorithm picks it up.

This control will prevent a malicious individual (both from within the company and external) to accidentally or purposefully deploy a file containing a virus to any environment.

**2. Registry Scanning**
TrendMicro provides customers with a real-time, continuous registry scanner which scans for vulnerabilities, malware and embedded secrets while the application is running. This also scans open-source libraries which are used in projects for vulnerabilities. When something is found, the user will at least be alerted. If possible, the suspicious process can also be terminated.

This control will detect new vulnerabilities and malware as soon as the virus definitions are updated. It could also prevent secrets from leaking using pattern recognition.

**3. Admission Control**
TrendMicro has a feature planned for a future release which ensures that only signed containers can be deployed and that potentially infected hosts can be excluded from container deployment. After a container is scanned for vulnerabilities and malware, it will be signed. When deploying, the deployment environment will check whether the signature is correct. On an incorrect signature, the deployment fails. When it is seen that a host is potentially infected with malware, the host will be excluded from container deployment and can be investigated further.

Only deploying signed containers will prevent direct deployment to the deployment environment without a build going through the CI/CD environment first, provided that the private key used for signing is not leaked and the deployment host is not compromised already. Excluding potentially infected hosts from deployment will prevent spreading malware to other hosts.

**4. Runtime Protection**
TrendMicro applies Malware scanning and Vulnerability Shielding to network traffic coming from and going to containers. This way, suspicious network traffic can be detected and stopped.

This control will prevent malicious network traffic from going to the host or other containers, thereby preventing exploits.

**5. Container Platform Protection**
TrendMicro provides a service to monitor Docker and Kubernetes. It can detect upgrades, downgrades and removal of these two applications. It also monitors the binaries of these programs for attribute changes and their processes for abnormal behaviour. It monitors changes to critical files, like configuration files, keys and certificates. It monitors for changes to iptables rules to prevent unauthorized port changes and permissions in key directories to prevent tampering with them. Lastly, it monitors key events like errors from forbidden actions.

These controls will protect the integrity of the container platform.

**6. Container Hosts Protection**
TrendMicro provides an anti-virus solution for machines hosting containers. The anti-virus can detect and remove malware from the host. This prevents malware on the host machine from disabling other controls or affecting the host machine or containers in any way.

This control will protect the integrity of the machines hosting containers.

**7. Application Protection (Serverless)**
In the case the application is stand-alone and does not run on a server, TrendMicro can provide a library which can detect malicious libraries and code in the application. This is not very applicable to the CI/CD pipelines, and is just named for the sake of completeness.

In conclusion, TrendMicro provides 7 controls to mitigate risks in CI/CD pipelines, of which 6 are related to the Production Line. These 6 will be included in the threat assessment.

### Source 4: Past incidents

Two incidents were reviewed and served as a source for our framework development. To get information about these incidents, the author was helped by the Design Authority of Shared Services. The company's CI/CD solution falls under Shared Services. Meeting notes can be found in.

The first one is code that belongs in the Production Line environment being uploaded outside of the Production Line. This could result to e.g. employees of the Indian branch of the company to gain access to source code that can't leave Europe.

The second one would be a hypothetical situation in which an unprotected Excel sheet with passwords is uploaded to the Production Line environment. This could result in people getting access to systems they shouldn't have access to.

Threats related to both incidents were included in the threat list for the threat model.

### Source 5: Research at another company

Since most of this is part of a non-disclosure agreement, the conversation with my supervisor I had with this will be summarized to include as few details as possible.

Some threats that were identified, are:

- Services in a Docker container should not run as root, but that is the first thing developers do when setting up the container.
- There can be so many tools in a pipeline that having access to logs and statistics of these tools might need to be done on a tool-per-tool basis. A recommended solution is to have this collected centrally in one tool, e.g. Splunk (monitoring tool) [37].
- No signature checking on deployed containers, which would allow unsigned Docker containers to be deployed.
- Developers having too many permissions. E.g. one developer can build and approve their own code. The separation of duties principle could be applied here. This means that a developer can't do a critical process all by themselves. [30]

## 5.3 Framework version 1.0

Now that some sources are identified for the threat model, we can start to build the framework. To do this, the threat model must be created, then controls for these threats must be identified and risk levels must be assigned to each threat.

A well-known way for threat modeling is the STRIDE method. [13] Each letter in STRIDE stands for a category of threats. For each tool, each currently identified threat will be categorized. The raw threat list that was categorized can be found in an appendix [Appendix A]. The categories are as follows:

- Spoofing - How can I spoof my identity?
- Tampering - How can I modify the product maliciously?
- Repudiation - How can I ensure no traces lead back to me?
- Information Disclosure - How can I make data available to people who shouldn't have access to it?
- Denial of Service - How can I make the service unavailable to users?
- Elevation of Privilege - How can I gain more permissions than I should have?

### 5.3.1 Groups of threats

Since there were common threats for each tool, these common threats were grouped. The decision to group these threats was made in a Skype call with my supervisor. Controls for these groups of threats are valid for most or all of the threats in the group. The entire group falls under one category of the STRIDE model. Please note that these Threat Groups do *not* include all threats. Threats that are not common to each tool are not included in the groups, but are included in the threat model separately. These Threat Groups are:

**Spoofing:**
Gaining access to another person's account:

- Compromise a user's password
- Physical access to a user's machine

- Compromise session cookie (using XSS for example)
- Access to the database to change someone's password, then login as them
- Bruteforce password attack

**Tampering:**
Abuse of privileges:

- Add/Remove projects
- Modify project settings
- Perform admin tasks maliciously

**Repudiation:**
Improper audit log security/redundancy:

- Local access to modify/delete audit logs, disable audit logging
- No audit logging in place?
- Hard drive failure

**Denial of Service:**
Denying access for users:

- Remove permissions for users
- Remove users
- Change password for user

Denying access to service:

- Local access to delete/encrypt files related to Jenkins
- Attacks on the network links on the public internet and attacks from the intranet

**Elevation of Privilege:**
Improper permission distribution:

- Social engineering someone to give permissions
- Give yourself more permissions
- Local access to the privilege database to manually insert permissions

Please note that there are no groups for the Information Disclosure category.

### 5.3.2 Threat model, controls and risk levels V1.0

What follows now is version 1.0 of my framework. This framework consists of three aspects. Each of these aspects will now be explained. A visual representation of the explanation that follows can be found in Figure 5.2.

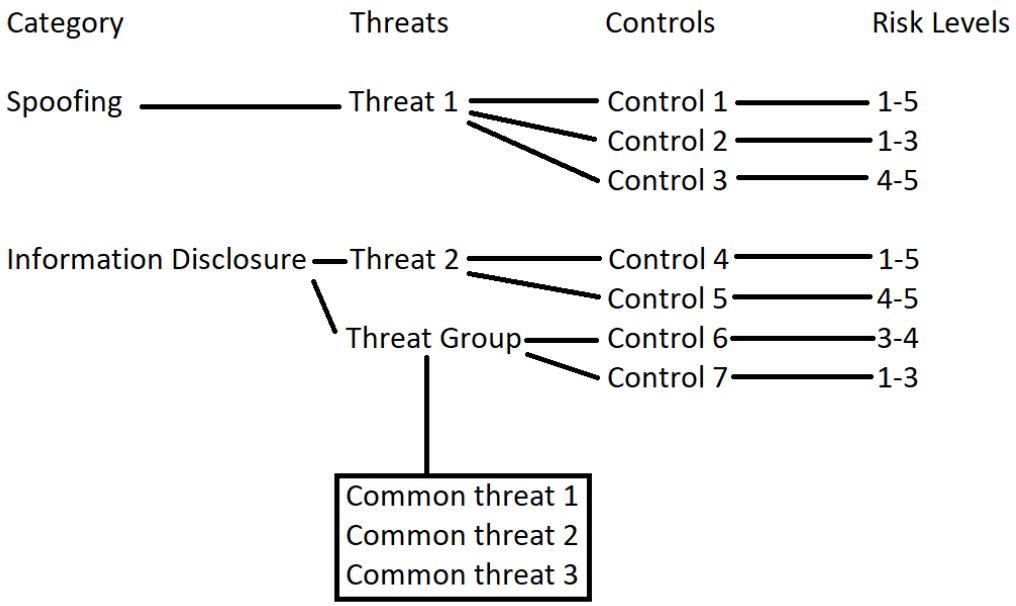


**Fig. 5.2.:** Visual representation of the first version of the framework

Firstly, the framework consists of the **threats** as identified previously and summarized in [Appendix A]. These threats are categorized using the STRIDE methodology as explained previously [13]. Some threats are **Threat Groups**, consisting of multiple threats with the same **controls**. The threats that these Threat Groups consist of, can be found in section 5.3.1. I have e-mailed the threat list to all of my contacts regarding the Production Line and CI/CD in general with request for feedback before making the framework. This resulted in feedback that I should have a look at the MITRE ATT&CK knowledge base [25] for more threats and controls. This resulted in seeing that the framework included most of the threats seen there. The knowledge base was therefore mainly used for identifying controls for threats.

Secondly, the framework consists of controls that can be used to mitigate these threats that are based on existing policies of the company, the MITRE ATT&CK knowledge base [25] and common sense.

Finally, the framework includes proposed **risk levels** for each combination of threat and control. Each project at the company has a certain risk level associated with them. Low-risk application include but are not limited to internal applications which are not connected to the internet. High risk applications include but are not limited to applications running on critical infrastructure. This risk level indicates at which risk level a control must be implemented on a scale of 1-5, with 1 being "low risk" and 5 being "high risk". These risk levels are based on CIAA (Confidentiality, Integrity, Availability and Accessibility) and are already established by the company in the Secure Development Baselines. I will make a proposal of these risk levels for each control to the best of my ability, and experts will be asked to either validate or improve it.

The first version of the framework can be found in [Appendix B].

This goal of this version of the framework was to serve as a basis to mainly get some feedback on the layout and the contents of the threat model, controls and risk levels. I have e-mailed this framework to the Secure Development Coordinator, who is an expert on the topic of frameworks. They provided me with excellent feedback. A meeting followed up on this e-mail. The feedback will be discussed in the next section.

### 5.3.3 Feedback on framework V1.0

As pat of the application of the design science research cycle, the first proposal of the framework was reviewed with the Secure Development Coordinator. They provided me with plenty of feedback on this version of the framework. In this section, the feedback will be summarized, and concrete actions will be proposed for V2.0 for the framework. Text in bold are the previously mentioned concrete actions, text not in bold are the reactions to this.

**Consider whether controls on the usage of the Production Line (the DevOps process) would be beneficial and/or outside the scope of this project. Examples would be gates and their prerequisites for passing.**
This request is considered out of scope for this project, since this framework is for the security of CI/CD pipelines, and not the DevOps process. No action will be taken on this point of feedback.

**Consider categorizing tool types, e.g. backlog and issue tracker, version control management and mapping the controls to these tool types, not to the specific products**

These requests are considered fair. The action that will be taken for V2.0 of the framework is to categorize the tool types according to what they are used for and not by their name. The name of the tool will still be included to show that the controls mostly apply to this tool, but the main categorization will be tool type and not tool name.

**Consider adding Integrated Development Environments (IDEs) like Eclipse to the framework**

This is a difficult one to decide. From the IDEs the sources are created, placed and retrieved in and out of the source code system, from which code scans and unit tests are initiated. The decision has been made to include the communication from the IDE to the version control system in the framework, but the IDEs themselves not. The reason is that IDEs run locally on developer's machines and the rest of the Production Line is run on a server somewhere. There are already baselines regarding the security of developer machines. The action that will be taken for V2.0 of the framework is that threats to the communication between the IDE and the version control system will be added to the threat model.

**Consider adding deployment tools like Ansible to the framework**

This request is considered fair. It is a framework for Continuous Integration *and* Continuous Deployment after all. It does not appear that Ansible is available for the Production Line, but since the threats, controls and risk levels will be categorized by tool type and not tool name, this is not a problem for the framework. However, for several reasons explained later in section 5.4.2, this tool will not be added to the framework.

**Consider that the core threats are not sufficiently visible. The Information Disclosure and Tampering threats are more about insight, stealing or manipulating the developed software. Make this the main focus of the threats**

This request is considered fair. The action that will be taken for V2.0 of the framework is that new threat analysis will take place on these two aspects with the developed software in mind.

**Consider that the DevOps street may contain production(-like) data which requires protection**

This request is considered fair. The action that will be taken is that during the new threat analysis for the Information Disclosure and Tampering threats, production(-like) data will be taken into account.

**Consider clarifying what Lam is**
This request is considered fair. Lam stands for LDAP Account Management. The action that will be taken for V2.0 of the framework is that this will be clarified by indicating that this is the Account Manager.

**Consider visualizing the DevOps street**
This request is considered fair. This is also feedback that I received from my supervisor. The action that will be taken for V2.0 of the framework is that this version will include a more visual variant, probably in the form of PowerPoint slides. An example of how these PowerPoint slides could look was sent to me by my supervisor, but can't be posted in this paper due to an NDA.

**Consider adding "who is responsible" to the controls**
After some discussion, this request is considered fair. Different controls apply to different groups of people. The Secure Development Baseline, which this framework might be merged into, is meant for developers of software. Developers are users of the Production Line. The action that will be taken for V2.0 of the framework is that for each control it will be noted to which group of people it is most applicable to. The visual variant of the framework, as discussed in the previous paragraph, will be mainly focused on threats, controls and risk levels which apply to developers.

**Consider which aspects are interesting for determining risk level**
This request is considered as already implemented. The risk levels are based on the risk levels that are in the Secure Development Baselines, which are based on CIAA (Confidentiality, Integrity, Availability and Accessibility. They are proposed by me and reviewed, validated and/or improved by experts. These experts will do so during after the evaluation of V2.0 of the framework, such that it can be included in V3.0 of the framework. No action will be taken for V2.0 of the framework.

**Consider adding layers of concern: customer, development, APaaS, PLaaS, TAM, Operations, and indicate control per layer**
This request is considered a significant amount of work and not critical for the development of the framework. Furthermore, it would likely be lost in translation to the Secure Development Baselines. It is therefore rated as low-priority and may or may not be included in future versions of the framework. The action that will be taken for V2.0 of the framework is that it will be considered to include this in this version of the framework, but most likely this will not happen.

**Consider adding more information on context and how to check the controls**
This request is considered fair and was planned already to be included in a future version of the framework. Since V2.0 of the framework is mainly visual, this is planned for V3.0 of the framework, which is the final version.

**Consider adding controls to protect data sources (source code, software config, binaries etcetera)**
After some discussion, this is considered to be a fair request. The action that will be taken for V2.0 of the framework is that threats to data sources in each tool will be reanalyzed and added to the framework.

**Consider extending the scope to include proprietary tools like Microsoft MFS**
This request is considered outside of the scope of this thesis, mainly because these tools are not offered as part of the Production Line. Some if not most controls will be applicable to proprietary tools as well, but no specific attention will be given to proprietary tools.

**Consider organizing controls according to architecture, business, information/data etcetera**
Controls will be grouped, but not according to this. I will keep them grouped per category of the STRIDE model.

**Consider that the framework is currently too high-level, and requires more details**
This request is considered fair and was planned already to be included in a future version of the framework. Since V2.0 of the framework is mainly visual, this is planned for V3.0 of the framework, which is the final version.

**Consider expanding threats and controls to authentication**
This request is considered fair. The action that will be taken for V2.0 of the framework is that threats to authentication for each tool will be re-assessed.

**Consider putting more focus on assets (data, source code, binaries)**
This request is considered fair. The action that will be taken for V2.0 of the framework is that threats to assets and controls for this will be re-assessed.

### 5.3.4 Conclusion and reflection on framework V1.0

STRIDE was a good method for threat modeling. Using this, concrete threats to the Production Line were found. It was very useful to group threats, to prevent an endless list of controls. There was not much feedback related to the controls and risk levels, but that was to be expected, since it wasn't asked for and the evaluation for this is planned after the creation of V2.0 of the framework.

Version 1.0 of the framework was a good first attempt at making a useful framework and has proven to be in a good format to receive feedback on. This feedback will

be taken into account for the creation of V2.0 of the framework. The concrete improvements that will be made can be found in the next section.

## 5.4 Framework version 2.0

After receiving plenty of feedback on V1.0 of the framework, it is time to build the second version. In this version of the framework, the following improvements will be made:

- Clarification of what “Lam" stands for (LDAP Account Manager)
- Categorization of tool types
- Explanation of exclusion of deployment tools like Ansible
- Threat assessment, control identification and risk level assessment of:
  - New threats identified in an interview with a student who used CI/CD in the past
  - Communication between the version control system and IDE
  - Core threats related to insight, stealing or manipulating the developed software in the Tampering and Information Disclosure categories
  - Production-like data in tools
  - Data sources in each tool (assets)
  - Threats related to authentication for each tool
- Adding who is responsible to each control
- Grouping of controls
- Visualization of the DevOps street

The second version of the framework will be structured as seen in Figure 5.3. It is very similar to the first version of the framework, but the “Who is responsible?" column has been added.

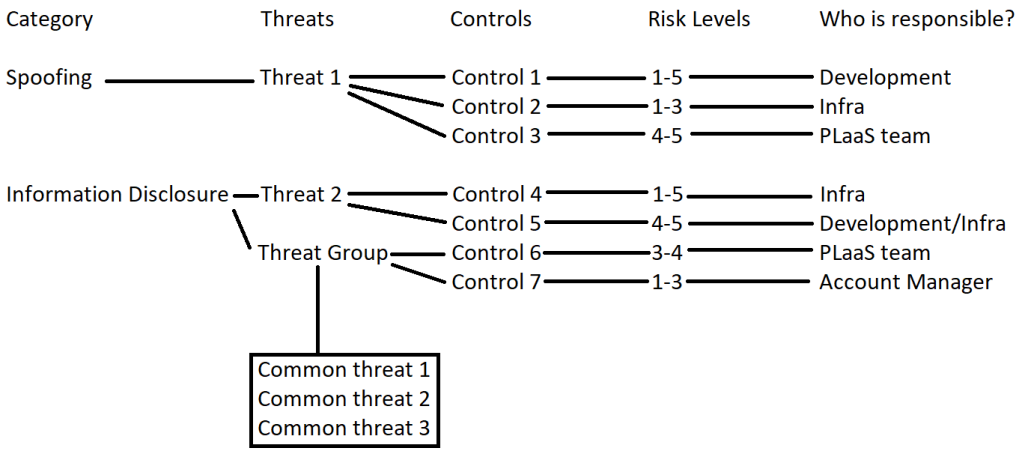


**Fig. 5.3.:** Visual representation of the second version of the framework

### 5.4.1 Clarification of “Lam"

We will start off by clarifying what “Lam" is. Lam stands for LDAP Account Manager. Lam is used for managing LDAP accounts that are used for the Production Line. Accounts can be created, deleted, assigned to groups and removed from groups. Groups have certain permissions assigned to them. An example is that every member of the “admins" group will be able to perform administrative tasks on certain aspects of the Production Line. Users within the “users" group have less permissions and can’t perform administrative tasks. In the rest of the paper, the term LDAP Account Manager will be used instead of Lam.

### 5.4.2 Categorization of tool types

Now that it is clear what “Lam" stands for, we will categorize the tool types. Some tool types were proposed in the feedback on the first version on the framework. Some tool types do not apply to the tools in the scope of this research, and are therefore left out. Some tools (like Selenium) don’t fall in the proposed categories and are therefore categorized differently.

Jenkins is an automation server, as stated by their own website [18] [33]: “The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project." [18]. It will therefore be categorized as such.

SonarQUBE is a static code analyzer, as stated by their About page: “SonarQube is an open source platform to perform automatic reviews with static analysis of code to

detect bugs, code smells and security vulnerabilities on 25+ programming languages [...]". [34] It will therefore be categorized as such.

LDAP Account Manager is, as the name says, an account manager. It will therefore be categorized as such.

Selenium is a bit more difficult to categorize. As per their website: “Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that." [31] Selenium is a tool that automates predefined actions in several browsers at once. It therefore helps developers test their programs faster. It could therefore be categorized as a “browser emulator". One could argue that it is a “test server", but it is not. The application is not launched on the Selenium server, Selenium simply connects to a running application and performs tests. It will therefore be categorized as “browser emulator".

Graylog is a log management tool. To get a citation for this, I had to get a bit creative. The “meta content" HTML tag is used to contain text that search engines will show in their search results. By viewing the page source of their main website, this quote was found: “Graylog is a leading centralized log management solution built to open standards for capturing, storing, and enabling real-time analysis of terabytes of machine data." [12] Graylog will therefore be categorized as a “log management tool".

Nexus3 is a repository manager. The text “Sonatype Nexus Repository Manager" is displayed in the top left corner when opening the tool. It will therefore be categorized as a “repository manager".

Grafana is a platform for visual analytics and monitoring. According to their website, they are “The open platform for beautiful analytics and monitoring" [21]. Since Grafana has multiple features, it will be categorized under multiple categories. The three categories Grafana falls in are: “analytics", “monitoring" and “dashboards".

Gitlab is a product with many features. In the Production Line, most of these features are disabled because there are dedicated tools that implement these features better. According to their website, “GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security." [11] It is only used as a version control system. Since the scope of this research is the tools in the Production Line, it will therefore be categorized as a “version control system".

#### Explanation of exclusion of deployment tools like Ansible

Although deployment tools like Ansible were previously considered to be within the scope of the framework, it was decided to not include these tools in this version of the framework for the following reasons:

- An approaching soft deadline on finishing the Excel version of V2.0 of the framework
- An approaching (but a little further away) hard deadline on finishing a visual version of V2.0 of the framework for the evaluation
- Lack of experience with Ansible and lack of time to get this experience
- No currently available threat models on the internet for Ansible or similar tools
- No time to find and interview experts on threats related to Ansible or comparable tools
- The fact that Ansible has more to do with the Operations side of DevOps, and that the framework will eventually be merged into the Secure Development Baselines, which is meant more for developers. This would result in threats and controls being left out, since they are not applicable to developers.

The value of the addition of Ansible and comparable tools will be re-evaluated during the group evaluation session of V2.0 of the framework. If deemed valuable, Ansible and comparable tools will be included in V3.0 of the framework. If time permits, a V4.0 of the framework could be created, in which feedback on V3.0 of the framework is processed. By making this decision, the framework will be delivered in time and all feedback on the previous version of the framework is processed carefully.

An e-mail was sent to the Secure Development Coordinator and my supervisor to inform them of my decision. The Secure Development Coordinator agreed with this proposal.

### 5.4.3 Threat assessment

Now that it is clear why deployment tools like Ansible weren't included in the framework, an interview with a student who has used CI/CD before in their projects

will follow. This interview identified new threats which will be added to the list of threats.

### Interview with a student who has used CI/CD before

The student studies at the University of Twente and is a good friend of the author. After the author asked around in a group chat who had worked with CI/CD, they said they had worked with CI/CD and that they wouldn't mind talking with me about the topic. A meeting was planned and a talk about CI/CD commenced. New threats identified during the meeting will now be discussed and added to the list of threats.

The first threat that they identified is a malicious insider pushing code. This falls under the "Uploading untested code to the repository" that was already identified in the first version of the framework. The controls identified for that threat are sufficient to cover this threat and a malicious insider pushing code will therefore not be considered further.

A noteworthy comment in the interview is that they say that when you have a small project, most threats are external, e.g. come from outside of the group that is working on the project. When an insider would sabotage the project, they would be sabotaging themselves as well. Therefore, there is no or little incentive to be malicious during a small project.

The second threat that they identify in the category "Spoofing" is having a user with a known username or e-mailaddress but slightly altered (e.g. uppercase i instead of lowercase L). This is a valid threat, and will therefore be added to the threat list.

The third threat that they identify is physical access to the machine of the person you want to spoof. Although this is a valid threat, the security of the machine of the developer is already covered by existing policies, and will therefore not be added to the threat list.

The fourth threat that they identify is that uploading malicious Javascript code to phish users for their login credentials. Uploading malicious code was discussed before and falls under the "Uploading untested code to the repository" category. Measures against phishing are already covered in other policies. The threat will therefore not be added to the threat list.

The fifth threat is that there might be less control on which libraries get added for the purpose of testing than for other purposes, e.g. development. This is a valid threat, and will therefore be added to the threat list.

The sixth threat that was identified is that there was a feature to revert to an older build. If a malicious build were uploaded, and later replaced by a non-malicious build, this feature could be used to go back to the malicious build. This is a valid threat, and will be added to the threat list.

The seventh threat is rewriting history in the master branch in Git. For some reason, an unprotected master branch was not yet considered in the first version of the framework. This is a significant risk, so the threat of abuse of unprotected master branch will be added to the threat list.

The eighth threat is that developers could have permission to delete other people's branches. This is a valid threat and will be added to the threat list.

The ninth threat is shared accounts. Shared accounts would be a threat in the category of repudiation. You can't audit who did something if there are multiple people on the same account. This is a valid threat and will be added to the threat list.

The tenth threat is code that is uploaded to the public domain. Code being re-uploaded by developers to places where it shouldn't be is a valid threat and will be added to the threat list.

The eleventh threat is social engineering to get more permissions. This is an already identified in the category of Tampering in the group Abuse of privileges.

The twelfth threat is that a malicious version of a tool gets added to the pipeline. This is a valid threat and will be added to the threat list.

Now that some new threats have been identified in an interview, threats related to the communication between the version control system and the IDE will be discussed.

### Communication between the version control system and the IDE

In this section, threats to the communication between the version control system (VCS) and the IDE will be identified. These threats and controls to them will be added

to the threat list for version 2.0 of the framework. Some existing controls will be moved to a different category, as explained later. Two aspects of this communication will be analyzed, namely the cloning of the code in the repository (when checking out the project) and pushing new code to the version control system.

Cloning the repository goes as follows: The IDE sends a request for cloning a repository. If needed, the VCS will request authentication. If authentication succeeds, a copy of the repository is sent to the IDE. There are two aspects which could go wrong here, namely the authentication and the connection.

Threats to authentication have already been identified as a global threat to the Production Line under the category “Spoofing" in the group “Gaining access to another person’s account". In terms of threats to the connection, the only threat related to information disclosure would be a Man-In-The-Middle attack. This attack could either sniff credentials or the source code itself.

Although this threat is already covered by the control “Ensure HTTPS is used throughout the Production Line", which is meant to cover the global threat of “Gaining access to another person’s account" in the category “Spoofing", it feels like this control is better fitted under the category “Information Disclosure". Sniffing authentication credentials might result in spoofing, but it is a possible result of information disclosure. Furthermore, obtaining the source code this way would definitely fall under “Information Disclosure". This control will therefore be moved to the “Information Disclosure" category. The threat that this control will be used for is “Man-In-The-Middle attacks".

### Core threats related to insight, stealing or manipulating the developed software in the Tampering and Information Disclosure categories

The feedback was given that the threats in the Tampering and Information Disclosure sections don’t accurately reflect the core threats related to those topics. This means that some core threats in these categories were absent or misplaced and have to be added or moved from other categories.

One thing that was already done to improve this is the addition of Man-In-The-Middle attacks to the Information Disclosure category with the control of using HTTPS in the Production Line. The reason for this addition can be found in section 5.4.3.

Another threat to Information Disclosure was identified in the interview with the student, which can be found in subsection 5.4.3. It is the tenth threat in the list,

“uploading code to the public domain". A similar threat was identified when looking at past incidents in the interview with Benoît, namely code being made uploaded such that other entities of the company had access to them, which shouldn’t have it. An example is the Indian branch of the company getting access to source code which was not supposed to leave Europe. This threat will be grouped as “Source code disclosure" under the category “Information Disclosure", and includes “Uploading code to the public domain" and “Entities within the company getting access to source code they shouldn’t have access to".

One could argue that modifying software that is running in production is a risk to the deployment tools. Deployment tools were decided not to be included in this version of the framework in section 5.4.2 and will therefore not be considered at this time.

To come back to the title of this subsection, insight in the source code of the developed software is managed by proper access control, which is covered under Spoofing, and by the new measures regarding Information Disclosure proposed in the second and third paragraph of this subsection.

Insight in the running software pre-production is related to deployment tools like Ansible, which were excluded from the scope of this framework. Insight in the running software in production is covered in other policies and outside of the scope of this framework.

Stealing the source code or artifacts would either mean improper access control, which is covered under Spoofing, or Information Disclosure, which, as previously said, has had controls added which should help to reduce the risk of this happening.

Manipulation of the source code can be done by rewriting Git history for example, as identified in the interview with the student in subsection 5.4.3. This threat has already been identified. Other ways include improper access control, which is covered under Spoofing.

In conclusion, this point of feedback is now properly addressed.

### Production-like data in tools

First of all, it's usually not a good idea to use production-like data in terms of security, but it does represent the production environment the best, so it can be understood why this is done.

Extreme care must be taken in case personal information is involved. Developers should never have access to personal information of real people. This could be a violation of the GDPR, since the data subject might not have given permission for processing of their data in this way [8]. Production-like configuration files would most likely be fine, provided that they don't contain secrets used in production. To make it clear, data stored in the database of the application that is developed is outside of the scope of this research. Data stored in the database of tools related to the Production Line is within the scope of this research, as long as they can be retrieved using the web interface or APIs. Direct access to the database is covered by other policies and outside of the scope of this research.

Most of the threats related to the databases of tools should be covered by existing policies, but there are some specific threats to certain applications that are not.

For Jenkins, that would be secrets (e.g. API keys) that are meant for production stored in a place where everyone has access to them. For Graylog, that would be logs that come from a Production Environment. For Grafana, that would be statistics and data coming from a Production Environment. These three threats will be added to the threat list.

### Data sources in each tool

The three threats identified in the previous subsection covers this subsection. Furthermore, the control of "Applying the principle of least-privilege to all accounts" ensures that data is exposed to as few people as possible, further reducing the risk.

### Threats related to authentication for each tool

In short, every tool that does not integrate with LDAP is a risk on itself. Every tool must support logging in with LDAP credentials and have their own authentication disabled. This one-time action improves security, reduces setup time and reduces maintenance time and cost. The threat that will be added to the threat list is "Tools

relying on their own authentication methods instead of LDAP", and will be added under the category “Repudiation", since someone could create an account with the name of a user that doesn’t use the tool and perform actions under their name, therefore leading to non-auditability as to who performed these actions.

### Conclusion, threat list V2.0 and summary of differences with the previous version of the framework

Some new threats have been identified using the feedback given on the first version of the framework. They are added to the framework. The full threat list V2.0 can be found in [Appendix C]. In summary, the differences in the second version of the framework in comparison to the first version of the framework are:

- Added the threat of a “*Man-In-The-Middle attack*" in the “*Information Disclosure category*" in the “*Global*" threats.
- Removed the control of “*Ensuring HTTPS is used throughout the Production Line*" from the “*Gaining access to another person’s account*" threat in the “*Spoofing*" category in the “*Global*" threats and added it to the newly created “*Man-In-The-Middle attack*" in the “*Information Disclosure category*" in the “*Global*" threats.
- Added the threat of “*Impersonating a user by creating a user with a slightly altered username or e-mailaddress*" to the “*Spoofing*" category in the “*Global*" threats, with the control of “*Inform developers that this might happen*" with a risk level of “*3,4,5*".
- Added the threat of “*Malicious tool added to pipeline*" to the “*Tampering*" category in the “*Global*" threats, with the control of “*Validate signatures of download of tools added to the pipeline before installing them*" with a risk level of “*4,5*".
- Added the threat of “*Shared accounts*" to the “*Repudiation*" category in the “*Global*" threats, with the control of “*Each developer must have and use their own account*" with a risk level of “*1,2,3,4,5*".
- Added the threat of “*Tools relying on their own authentication methods instead of LDAP*" to the “*Repudiation*" category in the “*Global*" threats, with the control of “*Disable each tools own authentication method and enable LDAP sign-in, e.g. through a plugin.*" with a risk level of “*2,3,4,5*".

- Added the threat of "*Secrets for production stored in a place where developers have access to them*" to the "*Information Disclosure*" category in the "*Automation Server (Jenkins)*" threats, with the control of "*Ensure that secrets for production are stored in a place where developers don't have access to them*" with a risk level of "*1,2,3,4,5*".

- Added the threat of "*Logs from production which developers have access to*" to the "*Information Disclosure*" category in the "*Log Management Tool (Graylog)*" threats, with the control of "*Ensure production logs are only visible on a need-to-know basis*" with a risk level of "*2,3,4,5*".

- Added the threat of "*Malicious builds not being deleted after identifying them as malicious*" to the "*Tampering*" category in the "*Repository Manager (Nexus3)*" threats, with the control of "*Require malicious builds to be (automatically) deleted*" with a risk level of 1,2,3,4,5.

- Added the threat of "*Statistics and data coming from a production environment*" to the "*Information Disclosure*" category in the "*Analytics, Monitoring and Dashboards (Grafana)*" threats, with the control of "*Ensure that statistics and data coming from a production environment is visible on a need-to-know basis*" with a risk level of "*2,3,4,5*".

- Renamed the threat of "*Uploading untested code to the repository*" in the "*Tampering*" category in the "*Version Control System (Gitlab)*" threats to "*Uploading untested code to the repository / master branch*".

- Changed the risk level of the control "*Require code reviews before changes can be pushed to master*" for the threat "*Uploading untested code to the repository / master branch*" in the "*Tampering*" category in the "*Version Control System (Gitlab)*" threats from "*3,4,5*" to "*2,3,4,5*" to reflect on the renaming of the threat of the previous bullet point.

- Added the threat of "*Developers being able to delete other people's branches*" to the "*Tampering*" category in the "*Version Control System (Gitlab)*" threats, with the control of "*Ensure that developers can only delete their own branches*" with a risk level of "*2,3,4,5*".

- Added the threat of "*Source code disclosure*" to the "*Information Disclosure*" category in the "*Version Control System (Gitlab)*" with the control of "*Ensure that every developer knows where code is allowed to be uploaded, e.g. by including this in every source code file*" with a risk level of "*3,4,5*".

### 5.4.4 Control identification, Risk level assessment, Adding who is responsible to each control and Grouping of controls

Controls have been identified similarly to V1.0, which were based on existing policies of the company, the MITRE ATT&CK knowledge base [25] and common sense.

Risk levels have been proposed similarly to V1.0 and will be either validated or improved by experts in the evaluation session.

Responsibilities have been proposed based on knowledge gained previously and by common sense. They will also be validated or improved by experts in the evaluation session.

The grouping of the controls is kept in line with STRIDE for this version of the framework, since that makes evaluation significantly easier. It will most likely be grouped differently in the next version.

The second version of the framework can be found in [Appendix D]. Additions to the framework have a green background, removals have a red background.

### 5.4.5 Visualization of the DevOps street

Feedback was given that this version of the framework was not very visual, and that it would be beneficial to have this, e.g. one PowerPoint slide per tool. Due to the complexity of the framework, as seen in Figure 5.3 and in the full second version of the framework in [Appendix D], and the great number of threats and controls, this is difficult to realize. To greatly simplify the framework and to have something comprehensible for the evaluation of this version, it was required to make some representation of the framework with a maximum of one slide per tool. The only way I saw this happening would be to make a table for each tool with 2 columns: the threat categories and the threats. I sent the PowerPoint to the Secure Development Coordinator and my Industry Supervisor for initial feedback, and the Secure Development Coordinator replied that the visualization was clear and decently. The author later spoke to their academic supervisor who suggested to make some improvements to the PowerPoint. This PowerPoint was used during the evaluation of the second version of the framework.

### 5.4.6 Evaluation of V2.0 of the framework

A meeting was planned with the Secure Development Coordinator and an Ethical Hacker to evaluate the second version of the framework. Although there might have only been two people to evaluate the framework, their skills are complementary. The Secure Development Coordinator is an expert on the topic of frameworks and baselines within the company, and the Ethical Hacker is an expert on threats, controls etcetera. Therefore, the framework can be evaluated and mostly validated by these two people. It is identified in the evaluation meeting that some additional validation is needed, which has to be done by other people. Now, a summary of the evaluation meeting will follow. Although the summary is quite elaborate, it is required to be this long. A list of actions that are taken on the basis of this evaluation meeting can be found in section 5.4.6.

#### Evaluation meeting

The evaluation meeting starts with the interviewer explaining the story until now. Then, the agenda for the evaluation is brought up. After the planning for the evaluation is clear, the global threats are evaluated. These threats are applicable to some, most or all tools of the Production Line. The Secure Development Coordinator mentions that the “Abuse of Privileges" could be placed in multiple threat categories of the STRIDE model [13]. The interviewer confirms this, and say that they believe it belongs in the “Tampering" category, because in the end, the “Abuse of Privileges" is used to tamper with the tool.

The point is brought up by the Ethical Hacker that the interviewer should look more at the side of a Developer every now and then, and not just at the side of a hacker. An example is that in the case external developers get hired who quickly want to develop code, they will try to get around certain checks to ensure their code is delivered faster. The interviewer argues that they have identified controls for multiple groups of people, and shows an example. The Ethical Hacker argues that the research is not at its full potential, due to not considering what this CI/CD pipeline is actually used for, and what the most important things are which you have to look at when you’re going to start using or implementing the CI/CD pipeline. The Secure Development Coordinator throws in some examples: Securing the source code and documentation such that they don’t get stolen and ensuring the environment isn’t taken down. These primary security goals of what the framework protects against are called Security Objectives. The Secure Development Coordinator thinks that naming five Security Objectives are sufficient and that the controls against threats which are in the framework can help achieve these Security Objectives.

The interviewer continues by arguing that an attempt was made to look more from the side of a developer by looking at past incidents and the research of the Ethical Hacker at another company for example. It is sad that the past incidents has only identified a few threats, but that significant effort was put into this. If this has to be improved, a concrete plan has to be made for this. The Secure Development Coordinator proposes to talk to the Project Manager and Agile Coach, who was interviewed previously during the exploratory research. If there were any such incidents, they should know about them. The Ethical Hacker agrees with this approach. It is decided to move forward with this plan.

A small discussion ensues, and ends in the Ethical Hacker identifying a new category of threats: “Configuration mistakes". A concrete threat to the Nexus Repository Manager is identified in this category: “Having debug mode enabled". An example of why this is a threat, is that with debug mode enabled, credentials are logged and everyone can see them. The proposal for a control is that there has to be a Secure Configuration Baseline for each tool, and that this baseline is checked. To implement such a Secure Configuration Baseline is outside of the scope of this research and is seen as potential Future Work.

The Ethical Hacker identifies another threat in the threat group “Configuration mistakes", namely “Use of root accounts". An example is that from the beginning a developer is told not to use a root account, but that they do it anyways, or that they simply rename the root account to e.g. “developer". There could already be existing baselines for secure configuration on the internet which could prevent these configuration mistakes from happening.

The evaluation continues with global threats. The interviewer explains that the Man-In-The-Middle attack was added to the “Information Disclosure" with the control using using HTTPS, as a response to feedback that this category was lacking core threats. The Secure Development Coordinator asks whether this also includes documents, production data, test data, source code, documentation and production-like test data, which are the most important points to them. The interviewer mentions that this point of feedback was addressed in this version of the framework, for example the threat of “Statistics and data coming from a production environment" in the “Analytics, Monitoring and Dashboards" section. Separate controls have been proposed for this threat.

The evaluation continues with threats identified to the Nexus Repository Manager. After discussing the limited list of threats, the comment is made by the interviewer that the threats identified earlier during this evaluation would make a great addition to this list.

Then, Graylog, the logging tool, is discussed. The threat of “Running demanding queries" in the “Denial of Service" category is discussed. The interviewer said that if this is expected to be a problem, a time-out for queries could be implemented. The Secure Development Coordinator mentions that this does not just relate to developers, but that it could also be abused by attackers. The interviewer agrees and confirms that this should definitely be a control.

Selenium is discussed. The interviewer mentions that this was a weird tool, because it only contained configuration. A lot of configuration information was visible. The Secure Development Coordinator asks whether that includes test scripts or that it was just configuration information to perform the tests. The Ethical Hacker says that he thinks it is in fact a script. The interviewer mentions that you could see the configuration, and logs in to their Production Line environment and shows the tool. It is seen that there is a WebDriver, which you could connect to, and that it for the rest is just configuration information. The Secure Development Coordinator then raises the question where the tests are which are executed. The interviewer mentions that they were not able to find these anywhere. The Secure Development Coordinator mentions that this is strange and proposes to ask the Project Manager and Agile Coach about this. The interviewer says that this would indeed be a nice question for them. The Secure Development Coordinator says that these tests could also be used to bring the environment down, e.g. by starting a performance test. The interviewer again shows the configuration information and says “That’s it". The Secure Development Coordinator thinks there is more than that. The interviewer agrees and will ask the Project Manager and Agile Coach about this.

Then, LDAP Account Manager is evaluated. There is a discussion about the purpose of this tool, and it is concluded that this is not clear. It is not clear whether the CORP account can be used to login to the Production Line, and for which tools the LDAP Account Manager provides accounts. The decision is made, on advice of the Ethical Hacker, to contact the person who gave the interviewer access to the server to gain access to this information. The importance of obtaining this information is emphasized by the Secure Development Coordinator, who says that the LDAP Account Manager is an important tool and if it were breached, and the persons who breached it could access the accounts of persons, they could do anything. It is also seen in the LDAP Account Manager that First names and Last names are visible, amongst other information. This could be a form of Information Disclosure, and has to be investigated further. Lastly, the Secure Development Coordinator comments that it was initially unclear and slightly confusing that the threat of a compromised account falls into the “Gaining access to another person’s account" threat in the “Global" threats, and that if a threat is significant to a certain tool, it could be named in that tool again. The Secure Development Coordinator understands why the decision was made to not name this threat again in the threats for this specific tool.

It has to be reconsidered whether to put significant threats to individual tools, which are already under the “Global" threats, in the threats to that individual tool again.

Then, Jenkins is considered. After naming the threats, the Secure Development coordinator asks whether specific controls for Docker and Kubernetes are within the scope. The interviewer says that there are currently two controls, namely “Monitor for changes in files related to Docker/Kubernetes" and “Ensure that the running Docker/Kubernetes is called from the correct path". The Secure Development Coordinator mentions that this does not cover, for example, configuration such that attackers cannot break out of Docker containers. The interviewer agrees that this should be included, but that they will make a control of “Ensure there is a baseline for Docker container security and ensure it is followed" and will not create this baseline itself. The interviewees confirm that this is fine and not expected.

Lastly, SonarQube is evaluated. The threat “Exploitable vulnerabilities visible to malicious user" was identified previously, but the Secure Development Coordinator argues that there are so many possible vulnerabilities and tests possible, a developer can make many mistakes, and that this is just one of the mistakes that there are and all mistakes that are made can be abused in the production environment, some mistakes might even be used consciously. When asked whether they see this as a threat or not, the Secure Development Coordinator argues that this is a threat, but not a very special one, and that it can be left in. The control for this, “Give permissions to show details of exploitable vulnerabilities only on a need-to-know basis" is quickly mentioned.

After this, it was quickly checked whether anything was missed. A quick discussion follows, and the control “Ensure that statistics and data coming from a production environment is visible on a need-to-know basis" for Grafana comes up. Some explanation is given on how this can be achieved, by making teams for example. The interviewees indicate that this is clear to them.

Now that the threats have been shown, the Secure Development Coordinator asks whether Ansible was included in this version of the framework. The interviewer mentions that Ansible was skipped for this version of the framework, and that it might be included in the next version of the framework, depending on how much time there is. The Secure Development Coordinator stresses the importance of this tool by saying that you could bring a Production Environment down, which could have major consequences. It is repeated that it is on the To Do list, and will be included if time allows.

The evaluation moves on with the validation of the visualization of the framework. Figure 5.3 is shown. The visualization is explained and it is asked whether this

is an accurate respresentation of the framework. Both the Secure Development Coordinator and the Ethical Hacker confirm this.

Then, an attempt to validate the identified threats is made. The sources for the threats are shown and explained. The interviewer mentions that the threats that will be identified in the future conversation about past incidents with the Project Manager and Agile Coach and threats identified during this evaluation will be included in the third version of the framework. When asked whether the threat list would be complete enough for the purpose of this framework, the Secure Development Coordinator says that they currently cannot judge this, and the interviewer proposes to send an e-mail to the interviewees with the request to take a look at this, which they agree to. After the conversation with the person who setup the Production Line environment, the conversation with the Project Manager and Agile Coach and the processing of the threats that were gathered during this evaluation, the threat list will be e-mailed to the Secure Development Coordinator and Ethical Hacker for validation.

After a discussion about the format of the third version of the framework, an attempt is made to validate the controls. It is decided to validate these over e-mail as well, in the same way the threats are validated.

Once this was decided, how to validate the risk levels is discussed. Someone will have to check and improve them. Normally, 4-5 man do this in a similar way to planning poker. The control is told to everyone, each person secretly writes down which risk level the control should apply to, and then it is thrown on the table. If there are minor differences, a majority vote decides. If there are major differences, a discussion happens to figure out why people have different opinions. The interviewer agrees that this is a good idea.

Now that a method for risk level validation is decided, it is time to find a method for validating the responsibilities. Since this is not very complex, this process can be done by one person. The Ethical Hacker volunteers to do this job. They will be e-mailed the framework and asked to either validate or improve who is responsible for a control.

The big questions follow: Is the framework useful to the interviewees? And how would it be used? The Secure Development Coordinator answers positively to the first question with “Yes." To the second question they answer: “There is currently already a demand for security baselines regarding CI/CD pipelines. For that, we just want to use it." The Ethical Hacker further validates the usefulness of the framework by saying that “In my view in the direction of customers, we can now

give consultancy with the advice you have put forward". These two answers validate the usefulness of the framework.

The evaluation concludes with the topic of confidentiality. It is asked whether the entire thesis is confidential, or that they can censor certain things and publish it. To summarize, the thesis has to be censored such that it meets Security Level 0. This means that it is public information. This Security Level can be reached by censoring certain parts of the thesis, and will be determined in consultation with the Ethical Hacker.

The Secure Development Coordinator mentions that it could be useful to create dataflow diagrams of threats, which would show how data is moved between tools. The interviewer mentions that something like that has already been created in the form of a whiteboard drawing, but that it could definitely be included in the thesis. This whiteboard drawing can be seen in Figure 5.4.

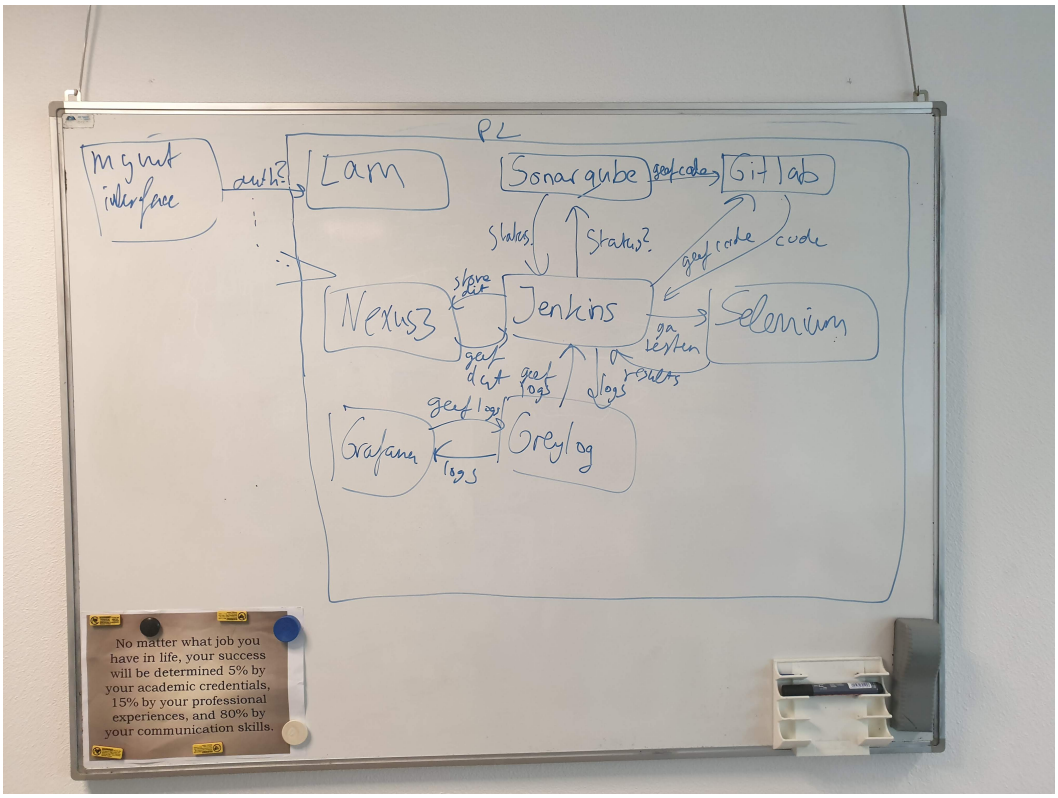


**Fig. 5.4.:** Whiteboard drawing of the visualization of the Production Line

The Secure Development Coordinator mentions that this does include the processes and the data flows, but not the data stores and the description of the tools. There is, for example, no flow to version control, and which tools use version control. The actors are also missing, both the good and malicious ones. It will be considered whether dataflow diagrams will be included in the thesis. After a quick rundown of the planning and a conclusion, the evaluation ends.

### Concrete points of action found from the evaluation

Now that the evaluation is summarized, the concrete points of action that were found during the evaluation will be presented. These are:

1. Name approximately five Security Objectives, which clarify what the framework aims to protect against
2. Add the threat of “Configuration Mistakes" to the framework and propose controls for this threat
3. Talk to the Project manager and Agile Coach about past incidents
4. Modify the control for the threat “Running demanding queries" for the Log Management Tool (Graylog) to stress its importance
5. Talk to the Project Manager and Agile coach about Selenium
6. Ask the person who gave the author access to the Production Line how LDAP is integrated in the Production Line
7. Reconsider explicitly naming significant threats to individual tools in the threat list for the individual tool, and not just in the global list
8. Add the control of “Ensure there is a baseline for Docker container security and ensure it is followed" to the framework
9. Reconsider adding Ansible to the framework
10. Evaluate risk levels with experts
11. Send an e-mail to the Secure Development Coordinator and Ethical hacker asking to validate that the threats and controls are accurate and sufficient
12. Send an e-mail to the Ethical Hacker to validate the responsibilities and process the reply
13. Censor the paper such that it reaches security level 0
14. Consider adding dataflow diagrams

Now that version 2 of the framework has been fully analyzed, version 3 of the framework can be built. Section 5.5.1 described how these actions are implemented.

## 5.5 Version 3 of the framework

In this version of the framework, feedback that was given during the evaluation of the second version of the framework will be implemented and the framework will be converted to match the format of current baselines of the company. First, each point of feedback will be addressed, and then, the framework will be converted.

### 5.5.1 Addressing feedback

In the following subsections, the feedback that was given during the evaluation meeting for the second version of the framework will be addressed.

**1. Name approximately five Security Objectives, which clarify what the framework aims to protect against**

In the evaluation meeting for the second version of the framework, the feedback was given that it was necessary to identify approximately five Security Objectives, which would describe what the framework aims to protect against. Three of such Security Objectives were given as an example: Securing the source code such that it doesn't get stolen, securing the documentation such that it doesn't get stolen and ensuring the environment isn't taken down. In this section, we will propose a couple more.

The first additional Security Objective is that the framework ensures that every action in the environment is auditable: it is clear who did what at which time. This is achieved by the controls in the "Repudiation" category.

The second additional Security Objective is that the framework prevents some human errors from happening. It was seen from the past incidents meeting with the Design Authority of Shared Services and in the evaluation meeting for the second version of the framework that human errors are a significant threat to the security of the Production Line. By giving the users a baseline to follow, some human errors can be prevented.

The complete list of Security Objectives will therefore be:

- Ensuring the source code doesn’t get stolen
- Ensuring the documenation doesn’t get stolen
- Ensuring the environment isn’t taken down
- Ensuring every action in the environment is auditable
- Ensuring that the number of human errors is minimal

### 2. Add the threat of “Configuration Mistakes" to the framework and propose controls for this threat

In the evaluation meeting for the second version of the framework, it was found that configuration mistakes are a threat to several tools in the Production Line. Leaving debug mode on in Jenkins could leave production credentials exposed in the logs, for example.

The control that was identified for this threat during the meeting, is that the configuration for each tool has to adhere to a Secure Configuration Baseline. It is not within the scope of this research do develop such a Secure Configuration Baseline for each tool, but adding a control for having and adhering to such a baseline is within the scope. The threat of Configuration Mistakes is placed the best in the category "Tampering", and therefore it will be added in the "Tampering" category in the "Global" threats, with the previously named control.

### 3. Talk to the Project manager and Agile Coach about past incidents

This point of feedback was addressed by sending an e-mail to the Project Manager and Agile Coach asking them whether they had any information regarding past incidents. To summarize, this didn’t result in any useful information. After informing my supervisor and the Secure Development Coordinator about this, the Secure Development Coordinator suggested to contact an Agile Evangelist, and so I did. They said that they have limited experience due to that they are only involved with one contract, but that they might be able to help. They also recommended to contact two other persons with more experience in the field. The phone call which was had later was expected to be recorded. The author of this paper had two call recorders running on their smartphone. Unfortunately, due to a policy change in the Google Play Store, some call recorders seized to function, amongst which the two the author

was running. Therefore, the information about the call in the next paragraph comes completely from memory. Normally, this would be put in an Appendix, but the summary of the phone call that would be provided normally is equal to what would be in the Appendix, so there is no point in doing so.

The Agile Evangelist mainly works with Microsoft products which are hosted on the Azure cloud. Some threats were identified regarding CI/CD pipelines. The first was not enforcing that passwords have to be changed on the first login, even though this was required by policy. Another threat was that users used their own personal e-mailaddresses (@hotmail.com) to gain access to the CI/CD environment. Later, this was fixed to include corporate e-mailadresses. Even later, a cleanup happened, and some people lost access to both accounts, preventing them from working. Another threat is that you might say that you're compliant, but you're actually not. This is very difficult to spot. Securing a CI/CD environment is very complex, since many, many tools are added to the CI/CD pipeline, which each have to be secured in their own way. This is especially the case with Java projects, for which nearly each build step has their own tool. Another threat the Agile Evangelist mentioned is that at some point they were able to push to production without the customer's approval.

Although the phone call with the Agile Evangelist seemed like it might have resulted in new threats, these threats were already present in the framework as part of a threat group or outside of the scope of the framework. That passwords have to be changed on the first login, falls under the threat of "Gaining access to another person's account". Personal e-mailaddresses are outside of the scope of this framework, since the Managed Production Line only works with corporate accounts, and LDAP Account Manager is synced to the Active Directory. The cleanup is not included for the same reasons. Saying that you are compliant, but are actually not is something that can be prevented by regularly checking the framework against the environment. The fact that many tools can be added to the pipeline is true, but the scope was limited to the tools in the Managed Production Line. The fact that they were able to push to production falls under the threat group "Configuration mistakes".

The Agile Evangelist also suggested having a phone call with two people, Engagement Manager 2 and Engagement Manager 3. Both people were e-mailed. Engagement Manager 2 replied saying that they haven't had incidents on their CI/CD environment, which is the Managed Production Line. They also say that the Production Line was made ISO27001 compliant. Since there was no response from Engagement Manager 3, a reminder was sent to them. After waiting several days, no response was received and no further attempts to communicate were made.

Significant effort has been put into gaining information regarding past incidents, which has not resulted in concrete new threats. We therefore believe we have learned as much as we can from these past incidents and will not explore this further.

#### 4. Modify the control for the threat “Running demanding queries" for the Log Management Tool (Graylog) to stress its importance

Initially, the idea was that the threat “Running demanding queries" could only be abused by developers. During the evaluation meeting, it became clear that attackers could also abuse demanding queries to create a Denial of Service attack. Therefore, the control for this threat was changed from “If this is expected to be a problem, limit the time a query can run and how many queries a user can run at the same time" to “Limit the time a query can run and how many queries a user can run at the same time". This stresses the importance of the control more than the old version.

#### 5. Talk to the Project Manager and Agile coach about Selenium

Although it was planned to talk to the Project Manager and Agile Coach about Selenium, the topic came up in a meeting with the person who gave me access to the Production Line. They were able to provide me with information that said that there are remote WebDrivers available in the Production Line. These remote WebDrivers can be connected to in the test code to run specific tests on. Example code to show how this would work can be seen in figure 5.5. A new threat assessment for this tool is required.

Configuration of Remote Webdriver to connect to Production Line Selenium Grid

```
@BeforeTest(alwaysRun = true)
public void setUp() throws Exception {
       DesiredCapabilities capability = DesiredCapabilities.firefox();
       capability.setBrowserName("firefox");
       capability.setPlatform(Platform.LINUX);
       this.driver = new RemoteWebDriver(
       new URL("http://selenium-hub-core:4444/wd/hub"), capability);
       this.baseUrl = "baseUrl Path";
       this.driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
}
```

**Fig. 5.5.:** Example code showing how to use the Selenium WebDrivers

Literature research is performed to see which threats exist against Selenium Grid. It is explicitly chosen to use Selenium Grid, since that is running in the Production Line.

The first identified threat is that files on the server could be accessible from the Selenium Webdriver [17]. By using file:\\URLs, it is possible to access OS files. When the researcher reported this, the developers said that: "*ChromeDriver is not designed to be robust against attacks, and users shouldn't rely on it as a security boundary. Instead, the system should be configured so that malicious code can't connect to ChromeDriver. ChromeDriver helps this by only allowing local connections by default, though when Selenium server is in use, non-local app can connect to ChromeDriver through Selenium server. Users should limit connections to the Selenium server, e.g., by running the who system behind a firewall. ChromeDriver should never be run with admin permissions, and should generally be run with a test account that only has the necessary permissions.*" So the control to the threat of "A malicious user can open an arbitrary file on the filesystem by abusing the tool" would be "Ensure the tool is ran under a non-root account, with only the permissions that it needs". This threat and the control is added to the framework under the category "Tampering".

More threats were identified in a Red Team exercise by Marcos Carro and their team [4]. It was possible for them to obtain test parameters by subscribing a new node to the Selenium Grid. It is also found that Remote Code Execution is possible on grids running Google Chrome. This is part of Google Chrome and not Selenium. The command line argument *–renderer-cmd-prefix* can be used for Google Chrome to run a command before opening the browser. It was not possible to check whether the Production Line was vulnerable to the Remote Code Execution in time, due to that there was no direct connection possible between the Selenium WebDrivers and the laptop of the author. It would be possible to test this by uploading a test script to the Production Line and then running it, but it would take significant effort to get this to work. In any case, the control for this threat remains the same as the control for the previously identified threat, namely that "Ensure the tool is ran under a non-root account, with only the permissions that it needs".

More literature research shows that Selenium was not built for security, and that it should not be trusted in any case. The control "Ensure the tool is ran under a non-root account, with only the permissions that it needs" stands. Therefore, all threats related to this control will be put under the threat group of "Selenium was not built for security". This will be adjusted in the framework. This will apply to all risk levels, and Infra will be responsible for this.

### 6. Ask the person who gave the author access to the Production Line how LDAP is integrated in the Production Line

This was done by sending this person a Skype message. They followed up with an e-mail asking for a time to discuss this. A Skype meeting was planned to discuss this. This person confirmed that LDAP Account Manager was synced to the Active Directory and that the CORP identities were used throughout the production line.

In the second evaluation meeting, the topic of Information Disclosure in LDAP Account Manager was discussed. Although the first name and last name are visible, this information is available to employees of the company by other means, e.g. through the Outlook Address Book. It is therefore not considered a threat, and no controls will be proposed for this.

### 7. Reconsider explicitly naming significant threats to individual tools in the threat list for the individual tool, and not just in the global list

Although it is understood why this is confusing, it is decided to not do this at this time. The framework in its current version will be reformatted to look like existing baselines of the company. Therefore, implementing this feedback would be a waste of time.

### 8. Add the control of “Ensure there is a baseline for Docker container security and ensure it is followed" to the framework

Since each tool of the Production Line is running in a Docker container, this control should be placed in the Global sheet. A threat against which this control could work is “Improper Docker container security". It is considered to be best fitting in the “Tampering" category.

### 9. Reconsider adding Ansible to the framework

Because of an approaching deadline, and this tool not being critical, it was decided to not further consider this tool, and to add it to Future work.

### 10. Evaluate risk levels with experts

A session was held with two experts to evaluate risk levels. We will call these experts Ethical Hacker and Ethical Hacker 2. The purpose of this session was to have these experts propose the risk levels of projects for which each control would have to be applied. As explained previously, each project has a certain risk level based on CIAA (Confidentiality, Integrity, Availability, Accessibility). An standalone non-sensitive application would for example have risk level 1, and a military grade application would have risk level 5. Risk levels are given on a scale of 1-5. The experts are asked to determine for which risk levels a control must be applied. A control is likely to be applicable to multiple risk levels. The experts were given a copy of the third version of the framework without the risk levels filled in.

The initial idea was to sit together, and do this in a similar way to planning poker. Everyone would have paper cards with all possible risk levels, but it was quickly determined that this would not be feasible, as this would result in many cards. Cards with the text 1, 2, 3, 1-3, 1-4, 1-5, 3-5 etcetera would all have to be printed. Therefore, the solution was implemented to physically sit together, but to make a Skype conversation. I would say the control, the experts would type out to which risk level(s) they believed this control was applicable to, and then hit enter at the same time. This was initially attempted, but after two controls it was found out that this took too much time. The solution was that they would fill out the risk levels in the spreadsheet separately from each other, and then e-mail them to me. If there was a significant difference between the risk levels they determined (more than 2 risk levels difference), it would be discussed and adjustments would be made to reflect this discussion. The average risk levels as found by the experts can be found in the third version of the framework, as seen in [Appendix E].

Also, a control for the threat “Malicious update of third-party library" was found during this meeting. The Ethical Hackers both agreed that the control of “Verify hashes outside of the Production Line and only allow verified libraries to be added to the environment" would be effective, with a risk level of 3, 4 and 5. This was added to the framework.

### 11. Send an e-mail to the Secure Development Coordinator and Ethical hacker asking to validate that the threats and controls are accurate and sufficient

Both the Secure Development Coordinator and Ethical Hacker were e-mailed the version of the framework as it stood before the risk level evaluation. They were asked to give final feedback and answer these two questions:

- Do you believe the threat list is complete and accurate enough? Did I miss anything essential in this list? Why/why not?

- Do you believe the controls against the threats that are current, accurate and effective enough? Did I miss anything essential in this list? Why/why not?

The Secure Development Coordinator asked what the definition of *complete and accurate enough/effective enough* was. The author answered by saying that this meant that the Secure Development Coordinator would not be able to come up with major things that the author forgot and that what is in the framework, is correct, and that the controls are effective against the threats.

Based on this definition, the Secure Development coordinator came up with some points of feedback. The first point was that the threat of “Malicious development has direct access to repositories with source codes, binaries, testscripts, documentation, images or business data circumventing the CI/CD products access control, stealing or manipulating the content" has to be added to the “Information Disclosure" category in the GLOBAL threats. This request is considered fair, and it was added. The control that is proposed for this is “Ensure that direct access is restricted on a need-to-have basis". The risk level that is proposed for this is 1, 2, 3, 4, 5, since this is applicable to all projects. It was not possible for the author to determine who would be responsible for this, so an e-mail was sent to the Ethical Hacker to ask this question. The Ethical Hacker and Ethical Hacker 2 (from the Risk Level Analysis were also asked to validate that this control would be applicable to projects with all risk levels. They responded by saying that this was indeed the case.

The second point of feedback was that a control was proposed for the threat of “Malicious update of third-party library" in the category “Tampering" for the Version Control System (Gitlab), which previously had none. The Secure Development Coordinator proposed the control of “Running a Software Composition Analysis tool, like Black Duck. If needed, only install signed or certified libraries". In the risk level evaluation, a similar control was proposed, namely “Verify hashes outside of the Production Line and only allow verified libraries to be added to the

environment". These controls are merged to form the control “Verify hashes and run a Software Composition Analysis tool against libraries outside of the Production Line environment. Only install libraries verified by this tool.". The Ethical Hacker and Ethical Hacker 2 were asked to propose risk levels for this control. They responded by saying that risk levels 3, 4 and 5 were appropriate.

The third point of feedback was that for the threat “Developers being able to delete other people’s branches" in the “Tampering" category for the Version Control System (Gitlab) should be improved by taking into account that branches can also be non-person specific, like feature and release branches, and that the master branch could also be deleted. The control for this threat was originally to “Ensure that developers can only delete their own branches". Following this feedback, it is proposed that the threat is changed to “Developers being able to delete branches they are not supposed to" and the control is changed to “Ensure that developers cannot delete any branches they shouldn’t be able to delete". The risk level and responsibilities are left unchanged.

The final point of feedback was that in the “Information Disclosure" category for the “Version Control System (Gitlab)" the threat of “If Git is in the cloud, then Git admins can get access to code" should be added. This point of feedback is captured in a previously identified threat, namely “Projects visible to people who don’t need access to them" in the “Information Disclosure" category for the Version Control System (Gitlab) and will therefore not be considered.

The Ethical Hacker also came up with some points of feedback. The first point of feedback is that for the control “Monthly checks that passwords are not logged" for the threat “Gaining access to another person’s account" in the “Spoofing" category in the “GLOBAL" threats, this should happen more often if sensitive data is involved. The control is therefore changed to “Check that passwords are not logged at least monthly. If sensitive data is involved, check this more often". The responsibilities and risk levels remain unchanged.

The second point of feedback is that for the control “Use strong passphrases for private keys" for the threat “Gaining access to another person’s account" in the “Spoofing" category in the “GLOBAL" threats, this should include username/passwords as well. The control is therefore changed to “Use strong passphrases for private keys and usernames/passwords". The responsibilities and risk levels remain unchanged.

The third point of feedback is that for the control “Monitor for malicious network traffic containing exploits" for the threat “Performing an exploit on a vulnerability to gain access" in the “Spoofing" category in the “GLOBAL" threats, anomalies in network traffic should also be monitored. Since an anomaly in network traffic could

for example be a bruteforce password attack, it is justified the control is changed to “Monitor for malicious network traffic containing exploits and anomalies in network traffic". The responsibilities and risk levels remain unchanged.

The fourth point of feedback is that for the control “Audit logging must be sufficient enough such that in the case of an incident, the cause of the incident can be determined" for the threat “Improper audit log security/redundancy" in the “Repudiation" category in the “GLOBAL" threats, the baseline for audit logging should be adhered to. The author was not aware that such a baseline existed, and the control is changed to “Ensure that the baseline for audit logging is followed". Since this baseline has its own risk levels, this control applies to projects with all risk levels, e.g. all projects with risk level 2 have to follow all controls in that baseline for projects with risk level 2. It is therefore justified to change the risk level for this control to 1, 2, 3, 4 and 5, and validation of this is not required. The responsibilities remain unchanged.

The fifth point of feedback is that for the control “Scan each file in each build with an anti-virus" for the threat “Upload a build with viruses" in the “Tampering" category for the “Automation Server (Jenkins)", it should be changed to “Follow the anti-virus baseline". This is considered fair and the control is thus changed. The risk level for this control was already 1, 2, 3, 4 and 5, so it can remain unchanged. The responsibilities remain unchanged as well.

The sixth point of feedback is not specific for a control, but more globally. It considers the question of how all of this will be checked, and that it maybe should be monitored that the baseline is enforced. The difficulty in this is that this baseline is a tool to prevent and detect security vulnerabilities in CI/CD pipelines. How the tool is used, is up to the user of the tool. It is therefore not considered to be in the scope of this research to enforce a certain use of the tool. This point of feedback is dismissed.

The final point of feedback is that for the control “Ensure that permissions for projects are given out on a need-to-have basis" for the threat “Projects visible to people who don’t need access to them" in the “Information disclosure" category for the “Static Code Analyzer (SonarQube), that the control should also include a need-to-see basis, and not just need-to-have. The control is therefore changed to “Ensure that permissions for projects are given out on a need-to-have and need-to-see basis". The risk levels and responsibilities remain unchanged.

The updated versions were e-mailed to the Secure Development Coordinator and the Ethical Hacker for their final points of feedback. The Secure Development Coordinator replied that it was good enough to be converted to a baseline, and the Ethical Hacker said that it was good enough as well. This validates the threats and controls.

#### 12. Send an e-mail to the Ethical Hacker to validate the responsibilities and process the reply

This point of action was taken exactly as stated. Most responsibilities were validated, but some were improved. Based on feedback received in the second research meeting regarding configuration mistakes, improper Docker container security and the newly identified threat against Selenium, new threats and controls were added. The Ethical Hacker was asked to validate the responsibilities for the controls against these threats separately, and they did so.

#### 13. Censor the paper such that it can be published

This point of action was taken and the paper was censored accordingly.

#### 14. Consider adding dataflow diagrams

It is not seen how adding dataflow diagrams adds a significant amount of value to this research. It would also take significant amount of work to make these. This point of feedback will not be implemented.

### 5.5.2 Version 3.0 alpha of the framework

Taking all the above points of feedback into consideration, the alpha version of the third version of the framework was delivered. The original plan was the third version of the framework would be in the form of a new baseline, but due to technical limitations this was not possible in time for a set deadline and required further effort. The third, alpha, version of the framework can be found in [Appendix E].

### 5.5.3 Converting the framework to match current baselines

The alpha version of the third version of the framework lays the foundation of the actual third version of the framework. The third version of the framework consists of a baseline. While converting the alpha version into the baseline, two problems were encountered. Each problem and their solution will be discussed in this section. After the problems and solutions are clear, the process of converting the baseline will be discussed.

### Sheet and macro protection

The template which I wanted to as a foundation for my baseline was protected. I was not able to edit any cells or macros without knowing a password. I made an attempt at removing this password using a Hex Editor, but that failed. I asked the Secure Development Corodinator whether they could help me out with this. They suggested that I planned a meeting with a Cybersecurity Consultant who would help me out with this. The meeting happened after the delivery deadline for the baseline, which is why it was not possible to deliver the baseline in time. The outcome of this meeting was that the Cybersecurity Consultant sent me an unprotected version of the sheet, which I was able to edit without problems.

### Re-appearing protection

The Excel sheet has two buttons, one to fill the baseline, and one to empty the baseline. Pressing any of these two buttons would re-enable the protection, therefore defeating the purpose of the unprotected sheet. After more e-mails with the Cybersecurity Consultant, I was able to obtain the password to undo this protection. With this password, I removed the re-appearing protection by editing the macros responsible for this. Finally, I was able to create and deliver the baseline, albeit later than planned.

### Creating the baseline

After the problems were solved, the baseline was ready to be made. The format of the baseline was similar to the format the framework was already in, but required some modifications. It was also more focused on controls rather than threats.

The first modification consisted of an extra field. This was a “Control ID", which is the unique identifier for the control. I decided it would be best to keep this in line with the already existing baseline and give them names like “GLOBAL-8" and “LOG-MANAGEMENT-TOOL-2", meaning the eighth control in the global category of threats and the second control of the log management tool category of threats respectively.

The second extra field was named “Control Group", which determined in which group a certain control would belong. Since I already grouped threats according to

the STRIDE methodology previously, I decided it would be best to fill this field with the STRIDE group that the control was meant for.

The third extra field was named “Name". This is the name of the control. In a couple of words, this would describe what the control was all about. I decided to take a look at existing names and base my names on that.

Fourthly, there was a “Description" field. I copied this pretty much one-to-one from the existing framework. This worked in most cases, with some minor edits necessary to make things more clear.

The fifth difference was an extra field named “Context". This would give some more context about the meaning of the control. I filled this field with some information to provide context to the reader.

The sixth difference was an extra field named “How to check". This is a field meant for the auditor, which indicates how to check that a control is properly implemented. I filled this field with exactly that information.

The seventh difference was that there were extra fields named “Compliant", “Applicability", “Mandatory", “Same control" and “General". These were presumably used by the macros to do some behind-the-scenes things. They were left unaltered, with the value for “Compliant" always being “Compliant", the value for “Applicability" always being “Always", the value for “Mandatory" always being “X", the value for “Same Control" always being empty and the value for “General" always being “H".

The eighth difference was a field named “Prio", which indicated risk levels. This was directly copied from the framework.

The baseline marks the end of three iterations of the Design Cycle. In the next section, we will discuss these results.

# 6 Discussion of Results

## 6.1 Implication

The objective of this research was to develop and validate a framework, that aims at preventing and detecting security vulnerabilities in Continuous Integration/Continous Delivery pipelines. To make this happen, we firstly have determined the scope of the research to be “Managed Production Line environments provided by the company to customers" together with experts from the company. Then, we proposed an initial version of the framework with threats and controls based on several sources. Using feedback we gathered from an expert within the company, the second version was delivered. In an evaluation meeting with two experts, more feedback was collected. Together with the results of the risk level evaluation meeting and validation on who is responsible for implementing each control, this resulted in the third and final version of the framework. This version was validated by experts, who said that the framework was ready to be adopted into the baselines. The new framework has significant implications on the use of the managed Production Line for the company, provided that it is used effectively.

### 6.1.1 Validity and reliability of the research

Throughout the research, it was continously kept in mind that each step of the creation of the framework had to be validated. In this section, validity of each aspect of the research will be discussed.

#### Thesis topic

The topic of the thesis is very industry-relevant. The author of this thesis judges this based on the fact that the Ethical Hacker came to them with this topic before they even signed a contract. Furthermore, the lack of published work also shows that more research into this area was needed. Lastly, it was continuously stressed in evaluations that the framework was important to the company, and that it is definitely useful to them.

### Research goal

The research goal was developed together with experts from the company over the course of three interviews, which can be seen in section 3.1. This ensured that the research was useful for the company.

### Research method

The research method that was used for this research is Wieringa's Design Science Methodology [42]. This is a well-known research method which currently has 375 citations on Google Scholar. This Research Method was recommended to the author by the Academic Supervisor and was applied successfully. Three cycles of the design cycles were executed, each resulting in a new version of the framework. For each version, feedback was requested and given, and in the final version, the threats, controls, risk levels and responsibilities were all validated.

### Author's knowledge of tools and the company

Since the scope was limited to the Managed Production Line environments provided by the company to customers, it was necessary to gain knowledge about this platform. Since there was no time to become an expert on the platform, some time was invested into building a sample project to get basic knowledge of how the Production Line works. In case there was a lack of knowledge, experts on the particular subject within the company would be asked to provide information. Extensive documentation was also available.

The author was not very familiar with big companies like the company. At the time of writing, the company has a significant number of employees. One of the goals of performing this research at the company was to get a feel of what working for a big company would be like. The author read through policies and other documents to get relevant information for this research. In case of a lack of knowledge, the author e-mailed, informally spoke or interviewed employees of the company to get information. The author was positively surprised by how much these people are willing to help and to make time for you if you need them to.

### Usefulness outside of the company

We believe that the framework can be used in similar CI/CD environments in other organizations. It can be used as a checklist to check compliancy or as a check to see whether basic security controls in CI/CD environments are implemented. It is very unlikely that the framework will directly map to another CI/CD environment, since there might be other tools in such an environment. In such case, an extension to the framework might be needed to fulfill the organisation's needs. It might also be possible that the organization does not use a tool that is in the framework, in which case the controls for that tool might have to be left out. In any case, it could serve as a basis for a checklist for other environments.

### Exploratory research

As I was told by my supervisor, not many research papers do exploratory research. I had to find out what the company wanted from the framework and how CI/CD is used within the company. Significant time was invested in having interviews with experts and transcribing and summarizing them. This resulted in all knowledge necessary to start building the framework in a way that is useful for the company.

### Completeness of the framework

The framework is classified as a living document. This means that additions can be and probably will be made at any time in the future. However, because the author did three cycles of Wieringa's Design Cycle [42] with expert reviews, we think that the level of completeness is more than satisfactory. The subsections below explain how this review process worked.

### Threats

The threats were initially based on five sources: a paper, how some of the current suppliers of CI/CD actively protect against threats, a webinar, past incidents and research from my supervisor at another country. This resulted in feedback from the Secure Development Coordinator that several major categories of threats were left unidentified, namely:

- New threats identified in an interview with a student who used CI/CD in the past
- Communication between the version control system and IDE
- Core threats related to insight, stealing or manipulating the developed software in the Tampering and Information Disclosure categories
- Production-like data in tools
- Data sources in each tool (assets)
- Threats related to authentication for each tool

These threats were analyzed and added to the framework. An interview with a student who used CI/CD in the past revealed some more threats as well. The second version of the framework was delivered and evaluated. This resulted in more missing threats to be added, namely those related to:

- The threat of “Configuration Mistakes" to the framework and propose controls for this threat
- Talk to the Project manager and Agile Coach about past incidents
- Talk to the Project Manager and Agile coach about Selenium
- Ask the person who gave the author access to the Production Line how LDAP is integrated in the Production Line
- Add the control of “Ensure there is a baseline for Docker container security and ensure it is followed" to the framework

After all threats were e-mailed to the Secure Development Coordinator and Ethical Hacker, they suggested some more changes, which can be found in section 5.5.1. After these small adjustments were applied, they both stated that these threats would be sufficient for further adoption in the baselines. This validates the threats.

### Controls

Controls were validated similar to threats. Each evaluation, feedback on controls would also be implemented. The difference is that the sources on which the controls are based, are different. These sources include policies of the company, a website with a knowledge base on attack vectors and common sense.

The Secure Development Coordinator and Ethical Hacker were asked to validate the controls at the same time as the threats, and they both confirmed that these threats would be sufficient for further adoption in the baselines. This validates the threats.

### Risk levels

Risk levels for each control were initially proposed by the author. The Ethical Hacker mentioned in the evaluation meeting for the second version of the framework that this was not standard practice at the company and that this should be done in a meeting with multiple experts. This was acknowledged by the researcher and such it was done. Full details on this meeting can be found in section 5.5.1. This means that the risk levels were evaluated in a way the company normally does this, and by experts, thus validating the risk levels in the framework.

### Responsibilities

Responsibilities of who must implement each control were initially proposed by the researcher. The Ethical Hacker mentioned in the evaluation meeting for the second version of the framework that they themselves could go over it and validate or improve these responsibilities. This plan was followed and the responsibilities were updated. This validates the responsibilities.

### Limited number of experts

For validation purposes, one to three experts were involved. This could be a possible threat to validity. However, we think this threat is minimal, because they were seasoned professionals in their fields and knew how to reason critically about the framework. The Ethical Hacker is an expert on the topic of threats, control etcetera. The Secure Development Coordinator is an expert on the topic of frameworks. These

knowledge bases can be used perfectly together for an accurate validation of the framework. Therefore, we see the opinion of these experts on validity of certain aspects of the framework as valid.

### 6.1.2 Conclusion of validation

Each part of the framework and development thereof was validated by experts. Therefore, we believe the framework is valid for use within the company within the provided scope. No claims about validity of the framework can be made when applied outside of the scope, although it might be applicable partly or entirely. If it is desired to use the framework in a different scope, a new risk analysis must be done. If new risks are found, controls must be identified for these risks, risk levels must be assigned and it must be determined who is responsible.

## 6.2 Limitations of the Research

### 6.2.1 Scope

Early on, it was determined to limit the scope to Managed Production Line environments delivered by the company. This was done since they are predictable and to ensure a high quality framework is delivered. This framework might be partially applicable to Unmanaged Production Line environments or at other CI/CD environments, but its effectiveness in those cases is currently unknown.

### 6.2.2 IDEs

After the feedback received on V1.0 of the framework, it was decided to exclude IDEs from the framework. It was decided to include the communication between the IDEs and the version control system, though. This framework is therefore not completely applicable to IDEs.

# 7 Conclusion

## 7.1 Answers to research questions

The main research goal of this study was:

*The goal of this research is to develop and validate a framework, that aims at preventing and detecting security vulnerabilities in Continuous Integration/Continuous Delivery pipelines in managed Production Line environments provided by the company to customers.*

In order to achieve this goal, Wieringa's Design Science and Methodology was applied. Various versions of the framework were delivered and feedback was received and processed on each of these versions. With the final version of the framework, we can answer the three research questions, which are the following:

1. *What types of risks are companies using the managed Production Line exposed to?* Companies using the managed Production Line are exposed to risks. Several threats to the Managed Production Line were identified. Using the STRIDE model [13], the threats were categorized. Threats can belong in the category of Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privileges. Although these threats are not immediate risks, they could be exploited to do damage in some form to the system or to the company. This means that there is a risk of a threat happening. The full list of identified threats can be found in [Appendix E]. These threats were validated by experts.

2. *Which practices exist to mitigate each type of risk?* For each of these threats, controls were identified to mitigate each threat. The full list of controls per threats can again be found in [Appendix E]. These controls were also validated by experts.

3. *How should these risks be mitigated for each risk level?* For each of the controls, risk levels were assigned in a risk level evaluation meeting with two experts. Large differences in proposed risk levels were discussed, small differences were averaged. The full list of risk levels can again be found in [Appendix E].

Based on these research questions, the research goal was achieved by combining these into a spreadsheet and validating each aspect of it. The result of this research is a developed and validated framework that aims at preventing and detecting security vulnerabilities in Continuous Integration/Continuous Delivery pipelines in managed Production Line environments provided by the company to customers, which is useful for the company.

## 7.2 Future work

There are multiple ways to extend this work. Below we describe them.

### 7.2.1 Future threats

The threat model as described in the paper was made in early-mid 2019. After the publication of this paper, new threats might have emerged for which this threat model has to be updated. It is impossible to predict these new threats, so it is very important to make your own threat model and add new threats and controls.

### 7.2.2 IDEs

IDEs integrate closely with CI/CD pipelines. Code is uploaded to and updated from version control systems for example. There might also be other features in IDEs which interact with CI/CD pipelines. It might be interesting to extend this framework or to create a new one which includes IDEs.

### 7.2.3 Secure Configuration Baseline

In the evaluation meeting for the second version of the framework, the group of threats of “Configuration mistakes" was found. A concrete example of a threat would be to leave Debug mode on in the Nexus Repository Manager, which would mean that all credentials are logged and visible to everyone. A control for this threat would be to create a Secure Configuration Baseline for each tool, containing controls for each tool on ensuring that the configuration of the tool is secure.

### 7.2.4 Deployment tools like Ansible

It was stressed in both the first and second evaluation that securing Ansible is a very important tool and that it has serious threats against it. Due to time and knowledge

constraints, it was not possible to find threats and controls for this tool. Future research could focus on identifying threats and controls regarding this tool.

## 7.3 Recommendations for the company

In this section, recommendations for the company are discussed. We will discuss what the company can do with this framework that they couldn't do before, discuss in which situations the framework is applicable and not, and which expertise is required to use the framework.

### 7.3.1 What can the company do with the new framework?

Previously, the company had no baseline that was applicable directly to Managed Production Line instances. Their baselines were too generic, and it was too difficult to see how to check each control in these instances. With the new framework, each control is made and intended for these instances, making it trivial to check them. On top of that, the new framework covers more threats and controls than their existing baseline, which reduces risk and improves security.

### 7.3.2 Situations in which the framework is applicable

The scope of this framework was limited specifically to Managed Production Line instances provided by the company. It is applicable in any circumstance where a Managed Production Line environment is provided by the company, both by internal development teams and when the Managed Production Line is offered as a service. It is not directly applicable to other CI/CD environments, e.g. with tools other than the ones that are in the Production Line by default. The framework could form a basis for this, but a new threat model would have to be made, it must be checked that the existing controls mitigate the risks associated with the threats in this model sufficiently and if not, controls should be modified or added.

### 7.3.3 Required expertise

It is recommended that the person who uses this framework has some experience with the Production Line and CI/CD in general. A cybersecurity background would be preferred. If the person who uses this framework might not have either of these, it is recommended to have a Subject Matter Expert at hand who can help answer any questions the auditor might have.

# Bibliography


[1]Keith H Anderson, John L Kenyon, Benjamin R Hollis, Jill Edwards, and Brad Reid. *Continuous deployment system for software development*. US Patent 8,677,315. 2014 (cit. on p. 1).

[2]Apache. *Maven*. `https://maven.apache.org/`. 2019 (cit. on p. 21).

[3]Len Bass, Ralph Holz, Paul Rimba, An Binh Tran, and Liming Zhu. „Securing a deployment pipeline“. In: *Proceedings of the Third International Workshop on Release Engineering*. IEEE Press. 2015, pp. 4–7 (cit. on pp. 2, 6, 22, 23).

[4]Marcos Carro. *Attacking QA platforms: Selenium Grid*. `https://www.tarlogic.com/en/blog/attacking-selenium-grid/`. 2019 (cit. on p. 60).

[5]Lianping Chen. „Continuous delivery: Huge benefits, but challenges too“. In: *IEEE Software* 32.2 (2015), pp. 50–54 (cit. on p. 1).

[6]Alistair Cockburn. *Agile software development*. Vol. 177. Addison-Wesley Boston, 2002 (cit. on pp. 5, 7).

[7]Maya Daneva and Chong Wang. „Security requirements engineering in the agile era: How does it work in practice?“ In: *2018 IEEE 1st International Workshop on Quality Requirements in Agile Projects (QuaRAP)*. IEEE. 2018, pp. 10–13 (cit. on p. 6).

[8]EU. *Document 32016R0679 (GDPR)*. `https://eur-lex.europa.eu/eli/reg/2016/679/oj`. 2016 (cit. on p. 45).

[9]Martin Fowler and Matthew Foemmel. „Continuous integration“. In: *Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf* 122 (2006), p. 14 (cit. on p. 5).

[10]Gitlab. *DevSecOps with GitLab*. `https://about.gitlab.com/solutions/dev-sec-ops/`. 2019 (cit. on p. 24).

[11]Gitlab. *Gitlab*. `https://gitlab.com`. 2019 (cit. on pp. 21, 39).

[12]Graylog. *Graylog*. `https://www.graylog.org/`. 2019 (cit. on pp. 21, 39).

[13]Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack. „Threat modeling-uncover security design flaws using the stride approach“. In: *MSDN Magazine-Louisville* (2006), pp. 68–75 (cit. on pp. 29, 31, 49, 75).

[14]Michael Hilton, Nicholas Nelson, Timothy Tunnell, Darko Marinov, and Danny Dig. „Trade-Offs in Continuous Integration: Assurance, Security, and Flexibility“. In: (2017) (cit. on p. 5).

[15]Jez Humble and David Farley. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010 (cit. on p. 5).

[16]Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer. „What is devops?: A systematic mapping study on definitions and practices“. In: *Proceedings of the Scientific Workshop Proceedings of XP2016*. ACM. 2016, p. 12 (cit. on pp. 5, 7).

[17]Peter Jaric. *Guest Blog: Don't Leave your Grid Wide Open*. `https://labs.detectify.com/2017/10/06/guest-blog-dont-leave-your-grid-wide-open/`. 2017 (cit. on p. 60).

[18]Jenkins. *Jenkins*. `https://jenkins.io/`. 2019 (cit. on pp. 21, 38).

[19]Jenkins. *Securing Jenkins*. `https://wiki.jenkins.io/display/JENKINS/Securing+Jenkins`. 2019 (cit. on p. 25).

[20]Michael Koopman and Maya Daneva. *Frameworks for Detecting and Preventing Security Vulnerabilities in Continuous Integration/Continuous Delivery Pipelines and Their Limitations: State of the Art*. 2019 (cit. on pp. 2, 3, 15, 17).

[21]Grafana Labs. *Grafana*. `https://grafana.com/`. 2019 (cit. on pp. 21, 39).

[22]LDAPAccountManager. *LDAP Account Manager*. `https://www.ldap-account-manager.org/`. 2019 (cit. on p. 21).

[23]Microsoft. *What is Continuous Delivery?* `https://docs.microsoft.com/en-us/azure/devops/learn/what-is-continuous-delivery`. Accessed: 2018-11-22. 2017 (cit. on p. 1).

[24]Microsoft. *What is Continuous Integration?* `https://docs.microsoft.com/en-us/azure/devops/learn/what-is-continuous-integration`. Accessed: 2018-11-22. 2017 (cit. on p. 1).

[25]MITRE. *MITRE ATT&CK™*. `https://attack.mitre.org/`. 2019 (cit. on pp. 31, 32, 48).

[26]Oracle. *Java*. `https://java.com/`. 2019 (cit. on p. 21).

[27]M. Perry, S. Schoen, and H. Steiner. *Reproducible Builds - YouTube*. `https://www.youtube.com/watch?v=ilu6yMBGS6I`. 2019 (cit. on p. 23).

[28]Ryan Quinn. *DigitalOcean Currents: December 2017*. `https://blog.digitalocean.com/currents-dec-2017/`. (Accessed on 12/07/2018). 2017 (cit. on p. 2).

[29]Pilar Rodríguez, Alireza Haghighatkhah, Lucy Ellen Lwakatare, et al. „Continuous deployment of software intensive products and services: A systematic mapping study“. In: *Journal of Systems and Software* 123 (2017), pp. 263–291 (cit. on p. 6).

[30]Ravi Sandhu. „Transaction control expressions for separation of duties“. In: *[Proceedings 1988] Fourth Aerospace Computer Security Applications*. IEEE. 1988, pp. 282–286 (cit. on p. 28).

[31]SeleniumHQ. *Selenium*. `https://www.seleniumhq.org/`. 2019 (cit. on pp. 21, 39).

[32]Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. „Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices“. In: *IEEE Access* 5 (2017), pp. 3909–3943 (cit. on p. 6).

[33]John Ferguson Smart. *Jenkins: The Definitive Guide: Continuous Integration for the Masses*. " O'Reilly Media, Inc.", 2011 (cit. on p. 38).

[34]SonarQUBE. *About SonarQUBE*. `https://www.sonarqube.org/about/`. 2019 (cit. on p. 39).

[35]SonarQUBE. *SonarQUBE*. `https://www.sonarqube.org/`. 2019 (cit. on p. 21).

[36]Sonatype. *Nexus Repository Manager 3*. `https://help.sonatype.com/repomanager3`. 2019 (cit. on p. 21).

[37]Splunk. *Splunk*. `https://www.splunk.com`. 2019 (cit. on p. 28).

[38]Spring. *Spring*. `https://spring.io/`. 2019 (cit. on p. 21).

[39]TrendMicro. *Webinar: Securing Containers and your CI/CD pipeline without friction*. `https://resources.trendmicro.com/2019-Q1-Europe-BeNeLux-BNX-WBN-27-03-SecuringContainersandyourCICDpipelinewithoutfriction_02TY-page.html`. 2019 (cit. on p. 22).

[40]Faheem Ullah, Adam Johannes Raft, Mojtaba Shahin, Mansooreh Zahedi, and Muhammad Ali Babar. „Security Support in Continuous Deployment Pipeline". In: *arXiv preprint arXiv:1703.04277* (2017) (cit. on pp. 1, 2, 7).

[41]Hugo Villamizar, Marcos Kalinowski, Marx Viana, and Daniel Méndez Fernández. „A Systematic Mapping Study on Security in Agile Requirements Engineering". In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE. 2018, pp. 454–461 (cit. on p. 6).

[42]Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014 (cit. on pp. 11, 17, 70, 71).

# List of Figures

# Raw threat list for framework V1.0

# A

Current threats to CI/CD pipelines include:

1. A remote exploit on one of the components
2. An indirect exploit using a third-party repository
3. Attacks on the network links between components of the CI/CD pipelines
4. Untested code
5. Known security issues not getting resolved
6. No insight in which security issues are present
7. Container images with known vulnerabilities
8. Secrets in plainsight
9. Improper access control to the CI/CD environment
10. CSRF attacks on Jenkins
11. Unprotected master branch
12. Too many permissions given to tools interacting with Jenkins
13. Users with local access modifying JENKINS_HOME and other folders related to the Production Line
14. Improperly configured Content-Security-Policy header
15. (Users exploiting Markup formatting)
16. Files containing viruses being uploaded to the repository

17. Unsigned container deployment
18. Compromised container host
19. Malicious network traffic
20. Compromised container platform (Docker, Kubernetes)
21. Vulnerability Management System (Central Monitoring of Vulnerabilities)
22. Developers best effort, not enough, different role as Product Owner
23. Docker container privileges
24. "Developer" account with root privileges
25. Key manager
26. Docker images run master process as root
27. Nexus artifacts need to be checked that they are signed
28. Compromised Jenkins/Slaves (monitoring)
29. Build privileges for build agents
30. Endpoint protection for container hosts
31. No insight in logs and statistics due to no centralized logging and statistics collection tool

# Framework v1.0

# B

Due to technical reasons, the first page of the framework will be on the next page.

GLOBAL

| TYPE | THREAT | CONTROL | RISK LEVELS |
|---|---|---|---|
| Spoofing | Gaining access to another person’s account | Require strong passwords | 1,2,3,4 |
| | | Require random passwords of significant length which have to be stored in a password vault | 5 |
| | | Require multifactor authentication using a digital token (e.g. MobilePass, Google Authenticator) | 3,4,5 |
| | | Require strong authentication using a digital token and a physical token (e.g. smartcard) | 5 |
| | | Implement rate limiting on login attempts | 1,2,3,4,5 |
| | | Set account lockout policies after a certain number of failed login attempts | 4,5 |
| | | Apply the principle of least-privilege to account managers | 3,4,5 |
| | | Automatically monitor for passwords and secrets in config files | 3,4,5 |
| | | Monthly checks that passwords are not logged | 2,3,4,5 |
| | | Install anti-virus on machine of developers | 1,2,3,4,5 |
| | | Ensure HTTPS is used throughout the Production Line | 1,2,3,4,5 |
| | | Monitor the password policy for change | 4,5 |
| | | Use strong passphrases for private keys | 1,2,3,4,5 |
| | | Ensure passphrases for private keys are stored securely | 1,2,3,4,5 |
| | | Use one account for everything in the Production Line and turn off account services in individual tools (e.g. Jenkins) | 1,2,3,4,5 |
| | Performing an exploit on a vulnerability to gain access | Install security patches in the next release of the Production Line | 1,2 |
| | | Install security patches within a month of them becoming available | 3 |
| | | Install security patches within a week of them becoming available | 4,5 |
| | | Monitor for malicious network traffic containing exploits | 3,4,5 |
| | | Monitor the patch level of tools | 4,5 |
| Tampering | Abuse of privileges | Give users limited permissions by default and have admins with more privileges | 1,2 |
| | | Apply the principle of least-privilege to all accounts, on a role-by-role basis | 3,4 |
| | | Apply the principle of least-privilege to all account, on an account-by-account basis | 5 |
| Repudiation | Improper audit log security/redundancy | Audit logging must be enabled | 1,2,3,4,5 |
| | | It must be monitored that audit logging is enabled | 3,4,5 |
| | | Audit logs should be saved to multiple places and backed up, such that a hard drive failure or ransomware cannot make all audit logs inaccessible | 3,4,5 |
| | | Audit logging must be sufficient enough such that in the case of an incident, the cause of the incident can be determined | 1,2,3,4,5 |
| | | Individual tools have to write to individual logfiles and not have permissions to write to other log files | 3,4,5 |
| Information disclosure | Version number visible | Version number of tools must be hidden | 3,4,5 |
| Denial of Service | Denying access for users | Provide the right to edit permissions of persons to as little people as possible | 2,3,4,5 |
| | | Apply the "separation of duties" principle for editing permissions | 4,5 |
| | Denying access to service | Install anti-virus on host and guest machines | 1,2,3,4,5 |
| | | Do not run Docker containers with elevated privileges | 1,2,3,4,5 |
| | | Do not run the process of the tool as root | 1,2,3,4,5 |
| | | Limit the files and folders the tool has access to to the minimum needed | 3,4,5 |
| | | Ensure DDOS protection/mitigation is present | 4,5 |
| Elevation of Privilege | Improper permission distribution | Only give permissions out on a need-to-have basis | 1,2,3,4,5 |
| | | Ensure users can't give themselves more permissions | 1,2,3,4,5 |
| | | Ensure it is not possible to access the tool's database to give more permissions to a user | 1,2,3,4,5 |

Jenkins

| TYPE | THREAT | CONTROL | RISK LEVELS |
|---|---|---|---|
| Tampering | Upload a build with viruses | Scan each file in each build with an anti-virus | 2,3,4,5 |
| | | Require an anti-virus scan before the container is signed and only deploy signed containers | 4,5 |
| | Having vulnerabilities only visible at run-time | Run OWASP ZAP or a similar vulnerability scanning tool periodically against pre-prod and prod environments | 3,4,5 |
| | Deploying unsigned containers | Only sign a container after it has passed all tests and checks, and only deploy signed containers | 2,3,4,5 |
| | Using malicious network traffic (e.g. to exploit things) | Scan for malicious network traffic on the host machine, the containers and between host and containers | 3,4,5 |
| | Having a compromised container host (Docker, Kubernetes) | Monitor for changes in files related to Docker/Kubernetes | 2,3,4,5 |
| | | Ensure that the running Docker/Kubernetes is called from the correct path | 4,5 |
| Information disclosure | Projects visible to people who don't need access to them | Ensure that when creating a new account, no projects are visible | 1,2,3,4,5 |
| | | Ensure that permissions for projects are given out on a need-to-have basis | 2,3,4,5 |

SonarQUBE

| TYPE | THREAT | CONTROL | RISK LEVEL |
|---|---|---|---|
| Tampering | Add/Modify/Remove checks performed on the code | Only give out permissions to do this on a need-to-have basis | 2,3,4,5 |
| Information disclosure | Exploitable vulnerabilities visible to malicious user | Give permissions to show details of exploitable vulnerabilities only on a need-to-know basis | 2,3,4,5 |
| | Projects visible to people who don't need access to them | Ensure that when creating a new account, no projects are visible | 1,2,3,4,5 |
| | | Ensure that permissions for projects are given out on a need-to-have basis | 2,3,4,5 |

Lam **GLOBAL captures all threats on Lam**

| TYPE | THREAT | CONTROL | RISK LEVEL |
|---|---|---|---|

Selenium

| TYPE | THREAT | CONTROL | RISK LEVEL |
|---|---|---|---|
| Information disclosure | Lot of configuration information visible | Minimize the information shown | 2,3,4,5 |
| | | Only provide access to Selenium on a need-to-know basis | 2,3,4,5 |

Graylog

| TYPE | THREAT | CONTROL | RISK LEVEL |
|---|---|---|---|
| Tampering | Add/Modify/Remove streams, alerts, dashboards and/or sources | Only give permissions out for Greylog features on a need-to-have basis | 3,4,5 |
| Denial of Service | Running demanding queries | If this is expected to be a problem, limit the time a query can run and how many queries a user can run at the same time | 1,2,3,4,5 |

Nexus3

| TYPE | THREAT | CONTROL | RISK LEVEL |
|---|---|---|---|
| Tampering | Uploading container images with known vulnerabilities | Only accept uploads made by Jenkins | 2,3,4,5 |
| | | Sign container images after they have been scanned for viruses and only accept signed container images | 3,4,5 |

Grafana

| TYPE | THREAT | CONTROL | RISK LEVEL |
|---|---|---|---|
| Information disclosure | Metrics visible to people who don't need access to them | Ensure that when creating a new account, no metrics are visible | 1,2,3,4,5 |
| | | Ensure that permissions for metrics are given out on a need-to-know basis | 2,3,4,5 |

Gitlab

| TYPE | THREAT | CONTROL | RISK LEVEL |
|---|---|---|---|
| Spoofing | Upload unsigned commit under someone else's name | Users must authorize to push a commit | 1,2,3,4,5 |
| | | Commits have to be cryptographically signed by the author with GPG | 4,5 |
| Tampering | Malicious update of third-party library | ??? - What can you even do? | ??? |
| | Uploading untested code to the repository | Require that code is tested by developers | 2,3,4,5 |
| | | Require code reviews before changes can be pushed to master | 3,4,5 |
| | | Require that tests have ???% code coverage | ??? |
| | Having dependencies with security vulnerabilities in your repository | Require that newer versions of dependencies with security updates are installed within 1 month of it releasing | 1,2 |
| | | Require that newer versions of dependencies with security updates are installed within 1 week of it releasing | 3,4 |
| | | Require that newer versions of dependencies with security updates are installed in the next release of the application | 5 |
| | Storing secrets in plaintext | There must be a policy on storing secrets | 1,2,3,4,5 |
| | | It must be audited that this policy is followed on a monthly basis | 3,4,5 |
| Information disclosure | Projects visible to people who don't need access to them | Ensure that when creating a new account, no projects are visible | 1,2,3,4,5 |
| | | Ensure that permissions for projects are given out on a need-to-have basis | 2,3,4,5 |

# Raw threat list for framework V2.0

C

Version 2.0 of the threat list adds the following threats to the existing threats:

- Spoofing user identity with a known username or e-mailaddress, but slightly altered
- Less control on which libraries get added for the purpose of testing than for other purposes, e.g. development
- Malicious builds not deleted after identifying them as malicious
- Unprotected master branch in version control
- Developers deleting other person's branches in version control
- Shared accounts
- Source code disclosure
- Malicious version of tool getting added to the pipeline
- Man-In-The-Middle attack
- Jenkins: Secrets for production stored in a place where developers have access to them
- Graylog: Logs from a production environment which developers have access to
- Grafana: Statistics and data coming from a production environment
- Tools relying on their own authentication methods instead of LDAP

# Framework v2.0

# D

Due to technical reasons, the first page of the framework will be on the next page.

GLOBAL

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Spoofing | Gaining access to another person's account | Require strong passwords | 1,2,3,4 | Infra |
| | | Require random passwords of significant length which have to be stored in a password vault | 5 | Infra |
| | | Require multifactor authentication using a digital token (e.g. MobilePass, Google Authenticator) | 3,4,5 | Infra |
| | | Require strong authentication using a digital token and a physical token (e.g. smartcard) | 5 | Infra |
| | | Implement rate limiting on login attempts | 1,2,3,4,5 | Infra |
| | | Set account lockout policies after a certain number of failed login attempts | 4,5 | Infra |
| | | Apply the principle of least-privilege to account managers | 3,4,5 | Street owner |
| | | Automatically monitor for passwords and secrets in config files | 3,4,5 | Infra |
| | | Monthly checks that passwords are not logged | 2,3,4,5 | Developers |
| | | Install anti-virus on machine of developers | 1,2,3,4,5 | Infra |
| | | Ensure HTTPS is used throughout the Production Line | 1,2,3,4,5 | |
| | | Monitor the password policy for change | 4,5 | Infra |
| | | Use strong passphrases for private keys | 1,2,3,4,5 | Developers |
| | | Ensure passphrases for private keys are stored securely | 1,2,3,4,5 | Infra/Developers |
| | | Use one account for everything in the Production Line and turn off account services in individual tools (e.g. Jenkins) | 1,2,3,4,5 | PLaaS dev team |
| | Performing an exploit on a vulnerability to gain access | Install security patches in the next release of the Production Line | 1,2 | PLaaS dev team |
| | | Install security patches within a month of them becoming available | 3 | PLaaS dev team |
| | | Install security patches within a week of them becoming available | 4,5 | PLaaS dev team |
| | | Monitor for malicious network traffic containing exploits | 3,4,5 | Infra |
| | | Monitor the patch level of tools | 4,5 | PLaaS dev team |
| | Impersonating a user by creating a user with a slightly altered username or e-mailaddress | Inform developers that this might happen | 3,4,5 | PLaaS dev team |
| Tampering | Abuse of privileges | Give users limited permissions by default and have admins with more privileges | 1,2 | PLaaS dev team |
| | | Apply the principle of least-privilege to all accounts, on a role-by-role basis | 3,4 | Account manager |
| | | Apply the principle of least-privilege to all account, on an account-by-account basis | 5 | Account manager |
| | Malicious tool added to pipeline | Validate signatures of downloads of tools added to the pipeline before installing them | 4,5 | Infra |
| Repudiation | Improper audit log security/redundancy | Audit logging must be enabled | 1,2,3,4,5 | Infra/Developers |
| | | It must be monitored that audit logging is enabled | 3,4,5 | Infra |
| | | Audit logs should be saved to multiple places and backed up, such that a hard drive failure or ransomware cannot make all audit logs inaccessible | 3,4,5 | Infra |
| | | Audit logging must be sufficient enough such that in the case of an incident, the cause of the incident can be determined | 1,2,3,4,5 | Infra/Developers |
| | | Individual tools have to write to individual logfiles and not have permissions to write to other log files | 3,4,5 | Infra |
| | Shared accounts | Each developer must have and use their own account | 1,2,3,4,5 | Account manager |
| | Tools relying on their own authentication methods instead of LDAP | Disable each tools own authentication method and enable LDAP sign-in, e.g. through a plugin. | 2,3,4,5 | PLaaS dev team |
| Information disclosure | Version number visible | Version number of tools must be hidden | 3,4,5 | PLaaS dev team |
| | Man-In-The-Middle attack | Ensure HTTPS is used throughout the Production Line | 1,2,3,4,5 | PLaaS dev team |
| Denial of Service | Denying access for users | Provide the right to edit permissions of persons to as little people as possible | 2,3,4,5 | Account manager |
| | | Apply the "separation of duties" principle for editing permissions | 4,5 | Account manager |
| | Denying access to service | Install anti-virus on host and guest machines | 1,2,3,4,5 | Infra/Developers |
| | | Do not run Docker containers with elevated privileges | 1,2,3,4,5 | Infra/Developers |
| | | Do not run the process of the tool as root | 1,2,3,4,5 | Infra/Developers |
| | | Limit the files and folders the tool has access to to the minimum needed | 3,4,5 | Infra/Developers |
| | | Ensure DDOS protection/mitigation is present | 4,5 | Infra |
| Elevation of Privilege | Improper permission distribution | Only give permissions out on a need-to-have basis | 1,2,3,4,5 | Account manager |
| | | Ensure users can't give themselves more permissions | 1,2,3,4,5 | Infra |
| | | Ensure it is not possible to access the tool's database to give more permissions to a user | 1,2,3,4,5 | Infra |

Jenkins

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Tampering | Upload a build with viruses | Scan each file in each build with an anti-virus | 2,3,4,5 | Infra |
| | | Require an anti-virus scan before the container is signed and only deploy signed containers | 4,5 | Infra |
| | Having vulnerabilities only visible at run-time | Run OWASP ZAP or a similar vulnerability scanning tool periodically against pre-prod and prod environments | 3,4,5 | Infra |
| | Deploying unsigned containers | Only sign a container after it has passed all tests and checks, and only deploy signed containers | 2,3,4,5 | Infra |
| | Using malicious network traffic (e.g. to exploit things) | Scan for malicious network traffic on the host machine, the containers and between host and containers | 3,4,5 | Infra |
| | Having a compromised container host (Docker, Kubernetes) | Monitor for changes in files related to Docker/Kubernetes | 2,3,4,5 | Infra |
| | | Ensure that the running Docker/Kubernetes is called from the correct path | 4,5 | Infra |
| Information disclosure | Projects visible to people who don't need access to them | Ensure that when creating a new account, no projects are visible | 1,2,3,4,5 | Account manager |
| | | Ensure that permissions for projects are given out on a need-to-have basis | 2,3,4,5 | Account manager |
| | Secrets for production stored in a place where developers have access to them | Ensure that secrets for production are stored in a place where developers don't have access to them | 1,2,3,4,5 | Developers/Infra |

SonarQUBE

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Tampering | Add/Modify/Remove checks performed on the code | Only give out permissions to do this on a need-to-have basis | 2,3,4,5 | Account manager |
| Information disclosure | Exploitable vulnerabilities visible to malicious user | Give permissions to show details of exploitable vulnerabilities only on a need-to-know basis | 2,3,4,5 | Account manager |
| | Projects visible to people who don't need access to them | Ensure that when creating a new account, no projects are visible | 1,2,3,4,5 | Infra |
| | | Ensure that permissions for projects are given out on a need-to-have basis | 2,3,4,5 | Account manager |

Lam **GLOBAL captures all threats on Lam**

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|

Selenium

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Information disclosure | Lot of configuration information visible | Minimize the information shown | 2,3,4,5 | Infra |
| | | Only provide access to Selenium on a need-to-know basis | 2,3,4,5 | Account manager |

Graylog

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Tampering | Add/Modify/Remove streams, alerts, dashboards and/or sources | Only give permissions out for Greylog features on a need-to-have basis | 3,4,5 | Account manager |
| Information Disclosure | Logs from a production environment which developers have access to | Ensure production logs are only visible on a need-to-know basis | 2,3,4,5 | Developers/Infra |
| Denial of Service | Running demanding queries | If this is expected to be a problem, limit the time a query can run and how many queries a user can run at the same time | 1,2,3,4,5 | Infra |

Nexus3

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Tampering | Uploading container images with known vulnerabilities | Only accept uploads made by Jenkins | 2,3,4,5 | Infra |
| | | Sign container images after they have been scanned for viruses and only accept signed container images | 3,4,5 | Infra |
| | Malicious builds not being deleted after identifying them as malicious | Require malicious builds to be (automatically) deleted | 1,2,3,4,5 | Infra |

Grafana

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Information disclosure | Metrics visible to people who don't need access to them | Ensure that when creating a new account, no metrics are visible | 1,2,3,4,5 | Account manager |
| | | Ensure that permissions for metrics are given out on a need-to-know basis | 2,3,4,5 | Account manager |
| | Statistics and data coming from a production environment | Ensure that statistics and data coming from a production environment is visible on a need-to-know basis | 2,3,4,5 | Account manager |

Gitlab

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Spoofing | Upload unsigned commit under someone else's name | Users must authorize to push a commit | 1,2,3,4,5 | Infra |
| | | Commits have to be cryptographically signed by the author with GPG | 4,5 | Developers/Infra |
| Tampering | Malicious update of third-party library | ??? - What can you even do? | ??? | Developers/Infra |
| | Uploading untested code to the repository / master branch | Require that code is tested by developers | 2,3,4,5 | Developers |
| | | Require code reviews before changes can be pushed to master | 2,3,4,5 | Developers/Infra |
| | | Require that tests have ???% code coverage | ??? | Developers/Infra |
| | Having dependencies with security vulnerabilities in your repository | Require that newer versions of dependencies with security updates are installed within 1 month of it releasing | 1,2 | Developers |
| | | Require that newer versions of dependencies with security updates are installed within 1 week of it releasing | 3,4 | Developers |
| | | Require that newer versions of dependencies with security updates are installed in the next release of the application | 5 | Developers |
| | Storing secrets in plaintext | There must be a policy on storing secrets | 1,2,3,4,5 | PLaaS team/Developers/Infra |
| | | It must be audited that this policy is followed on a monthly basis | 3,4,5 | PLaaS team/Developers/Infra |
| | Developers being able to delete other people's branches | Ensure that developers can only delete their own branches | 2,3,4,5 | Infra |
| Information disclosure | Projects visible to people who don't need access to them | Ensure that when creating a new account, no projects are visible | 1,2,3,4,5 | Account manager |
| | | Ensure that permissions for projects are given out on a need-to-have basis | 2,3,4,5 | Account manager |
| | Source code disclosure | Ensure that every developer knows where code is allowed to be uploaded, e.g. by including this in every source code file | 3,4,5 | Developers/Infra |

# Framework v3.alpha.1

E

Due to technical reasons, the first page of the framework will be on the next page.

| GLOBAL TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Spoofing | Gaining access to another person's account | Require strong passwords | 12345 | Infra |
| | | Require random passwords of significant length which have to be stored in a password vault | 2345 | Infra |
| | | Require multifactor authentication using a digital token (e.g. MobilePass, Google Authenticator) | 345 | Infra |
| | | Require strong authentication using a digital token and a physical token (e.g. smartcard) | 345 | Infra |
| | | Implement rate limiting on login attempts | 2345 | Infra |
| | | Set account lockout policies after a certain number of failed login attempts | 2345 | Infra |
| | | Apply the principle of least-privilege to account managers | 2345 | Street owner |
| | | Automatically monitor for passwords and secrets in config files | 345 | Infra |
| | | Check that passwords are not logged at least monthly. If sensitive data is involved, check this more often | 2345 | Developers |
| | | Install anti-virus on machine of developers | 12345 | Infra/Application Owner |
| | | Monitor the password policy for change | 12345 | Infra |
| | | Use strong passphrases for private keys and usernames/passwords | 12345 | Developers |
| | | Ensure passphrases for private keys are stored securely | 12345 | Infra/Developers |
| | | Use one account for everything in the Production Line and turn off account services in individual tools (e.g. Jenkins) | 345 | PLaaS dev team |
| | Performing an exploit on a vulnerability to gain access | Install security patches in the next release of the Production Line | 34 | PLaaS dev team |
| | | Install security patches within a month of them becoming available | 12 | PLaaS dev team |
| | | Install security patches within a week of them becoming available | 5 | PLaaS dev team |
| | | Monitor for malicious network traffic containing exploits and anomalies in network traffic | 12345 | Infra |
| | | Monitor the patch level of tools | 12345 | PLaaS dev team |
| | Impersonating a user by creating a user with a slightly altered username or e-mailaddress | Inform developers that this might happen | 12345 | PLaaS dev team |
| Tampering | Abuse of privileges | Give users limited permissions by default and have admins with more privileges | 12345 | PLaaS dev team |
| | | Apply the principle of least-privilege to all accounts, on a role-by-role basis | 12345 | Account manager |
| | | Apply the principle of least-privilege to all account, on an account-by-account basis | 12345 | Account manager |
| | Malicious tool added to pipeline | Validate signatures of downloads of tools added to the pipeline before installing them | 12345 | Infra |
| | Configuration mistakes | Have and adhere to a Secure Configuration Baseline | 12345 | Infra/Developers |
| | Improper Docker container security | Ensure there is a baseline for Docker container security and ensure it is followed | 12345 | Infra/Developers |
| Repudiation | Improper audit log security/redundancy | Audit logging must be enabled | 2345 | Infra/Developers |
| | | It must be monitored that audit logging is enabled | 2345 | Infra |
| | | Audit logs should be saved to multiple places and backed up, such that a hard drive failure or ransomware cannot make all audit logs inaccessible | 2345 | Infra |
| | | Ensure that the baseline for audit logging is followed | 12345 | Infra/Developers |
| | | Individual tools have to write to individual logfiles and not have permissions to write to other log files | 345 | Infra |
| | Shared accounts | Each developer must have and use their own account | 12345 | Account manager |
| | Tools relying on their own authentication methods instead of LDAP | Disable each tools own authentication method and enable LDAP sign-in, e.g. through a plugin. | 45 | PLaaS dev team |
| Information disclosure | Version number visible | Version number of tools must be hidden | 345 | PLaaS dev team |
| | Man-In-The-Middle attack | Ensure HTTPS is used throughout the Production Line | 12345 | PLaaS dev team |
| | Malicious development has direct access to repositories with source codes, binaries, testscripts, documentation, images or business data circumventing the CI/CD products access control, stealing or manipulating the content. | Ensure that direct access is restricted on a need-to-have basis | | |
| Denial of Service | Denying access for users | Provide the right to edit permissions of persons to as little people as possible | 345 | Account manager |
| | | Apply the "separation of duties" principle for editing permissions | 345 | Account manager |
| | Denying access to service | Install anti-virus on host and guest machines | 12345 | Infra/Developers |
| | | Do not run Docker containers with elevated privileges | 12345 | Infra/Developers |
| | | Do not run the process of the tool as root | 12345 | Infra/Developers |
| | | Limit the files and folders the tool has access to to the minimum needed | 345 | Infra/Developers |
| | | Ensure DDOS protection/mitigation is present | 345 | Infra |
| Elevation of Privilege | Improper permission distribution | Only give permissions out on a need-to-have basis | 12345 | Account manager |
| | | Ensure users can't give themselves more permissions | 2345 | Infra |
| | | Ensure it is not possible to access the tool's database to give more permissions to a user | 2345 | Infra |

Newly added
Validated
Changed and validated

Jenkins

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Tampering | Upload a build with viruses | Follow the anti-virus baseline | 12345 | Infra |
| | | Require an anti-virus scan before the container is signed and only deploy signed containers | 12345 | Infra |
| | Having vulnerabilities only visible at run-time | Run OWASP ZAP or a similar vulnerability scanning tool periodically against pre-prod and prod environments | 12345 | Infra |
| | Deploying unsigned containers | Only sign a container after it has passed all tests and checks, and only deploy signed containers | 2345 | Infra |
| | Using malicious network traffic (e.g. to exploit things) | Scan for malicious network traffic on the host machine, the containers and between host and containers | 2345 | Infra |
| | Having a compromised container host (Docker, Kubernetes) | Monitor for changes in files related to Docker/Kubernetes | 12345 | Infra |
| | | Ensure that the running Docker/Kubernetes is called from the correct path | 12345 | Infra |
| Information disclosure | Projects visible to people who don't need access to them | Ensure that when creating a new account, no projects are visible | 12345 | Account manager |
| | | Ensure that permissions for projects are given out on a need-to-have basis | 12345 | Account manager |
| | Secrets for production stored in a place where developers have access to them | Ensure that secrets for production are stored in a place where developers don't have access to them | 12345 | Developers/Infra |

SonarQUBE

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Tampering | Add/Modify/Remove checks performed on the code | Only give out permissions to do this on a need-to-have basis | 345 | Account manager |
| Information disclosure | Exploitable vulnerabilities visible to malicious user | Give permissions to show details of exploitable vulnerabilities only on a need-to-know basis | 345 | Account manager |
| | Projects visible to people who don't need access to them | Ensure that when creating a new account, no projects are visible | 2345 | Infra |
| | | Ensure that permissions for projects are given out on a need-to-have and need-to-see basis | 2345 | Account manager |

Lam **GLOBAL captures all threats on Lam**

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|

Selenium

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Tampering | Selenium was not built for security | Ensure the tool is ran under a non-root account, with only the permissions that it needs | 12345 | Infra |
| Information disclosure | Lot of configuration information visible | Minimize the information shown | 345 | Infra |
| | | Only provide access to Selenium on a need-to-know basis | 12345 | Account manager |

Graylog

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Tampering | Add/Modify/Remove streams, alerts, dashboards and/or sources | Only give permissions out for Greylog features on a need-to-have basis | 12345 | Account manager |
| Information Disclosure | Logs from a production environment which developers have access to | Ensure production logs are only visible on a need-to-know basis | 12345 | Developers/Infra |
| Denial of Service | Running demanding queries | Limit the time a query can run and how many queries a user can run at the same time | 2345 | Infra |

Nexus3

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Tampering | Uploading container images with known vulnerabilities | Only accept uploads made by Jenkins | 2345 | Infra |
| | | Sign container images after they have been scanned for viruses and only accept signed container images | 2345 | Infra |
| | Malicious builds not being deleted after identifying them as malicious | Require malicious builds to be (automatically) deleted | 12345 | Infra |

Grafana

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Information disclosure | Metrics visible to people who don't need access to them | Ensure that when creating a new account, no metrics are visible | 12345 | Account manager |
| | | Ensure that permissions for metrics are given out on a need-to-know basis | 12345 | Account manager |
| | Statistics and data coming from a production environment | Ensure that statistics and data coming from a production environment is visible on a need-to-know basis | 12345 | Account manager |

Gitlab

| TYPE | THREAT | CONTROL | RISK LEVELS | WHO IS RESPONSIBLE |
|---|---|---|---|---|
| Spoofing | Upload unsigned commit under someone else's name | Users must authorize to push a commit | 12345 | Developers/Infra |
| | | Commits have to be cryptographically signed by the author with GPG | 345 | Developers/Infra |
| Tampering | Malicious update of third-party library | Verify hashes outside of the Production Line and only allow verified libraries to be added to the environment | | |
| | Uploading untested code to the repository / master branch | Require that code is tested by developers | 345 | Developers/Testers |
| | | Require code reviews before changes can be pushed to master | 345 | Developers/Infra |
| | | Require that tests have ???% code coverage | 345 | Developers/Infra |
| | Having dependencies with security vulnerabilities in your repository | Require that newer versions of dependencies with security updates are installed within 1 month of it releasing | 12345 | Developers/AO |
| | | Require that newer versions of dependencies with security updates are installed within 1 week of it releasing | 345 | Developers/AO |
| | | Require that newer versions of dependencies with security updates are installed in the next release of the application | 345 | Developers/AO |
| | Storing secrets in plaintext | There must be a policy on storing secrets | 2345 | PLaaS team/Developers/Infra |
| | | It must be audited that this policy is followed on a monthly basis | 2345 | PLaaS team/Developers/Infra |
| | Developers being able to delete branches they are not supposed to | Ensure that developers cannot delete any branches they shouldn't be able to delete | 12345 | Infra/Developers |
| Information disclosure | Projects visible to people who don't need access to them | Ensure that when creating a new account, no projects are visible | 12345 | Account manager |
| | | Ensure that permissions for projects are given out on a need-to-have basis | 12345 | Account manager |
| | Source code disclosure | Ensure that every developer knows where code is allowed to be uploaded, e.g. by including this in every source code file | 12345 | Developers/Infra |