**Case study:  
Security**

**PREPARED FOR**

Bartosz Paszkowski

Frank Coenen

**PREPARED BY**

Maarten Blömer

Nick Krijgsman

Faruk Aydin

Jursley Gonzalez

Vincent Andersen

# **Table of Contents**

[**Table of Contents**](#_pykfpgqrlej) **2**

[**Case Study**](#_3s39dys3cx66) **3**

[Case goal:](#_qwqn8mh5s18) 3

[Case questions:](#_4frtw1206ej) 3

[Questions to be mindful of:](#_rhg0mb87rhxw) 3

[Case purposes:](#_2bhbl6st83h1) 3

[**Brainstorm**](#_whir32t8ue2v) **4**

[**Security Related tasks**](#_wisaxg2eeii0) **5**

[Jira tasks related to security](#_bnr4vo3zzw77) 5

[**Security by Design plan**](#_j4m0m4i7c598) **6**

[Security requirements](#_ifu62ibmd0as) 6

[Security requirements table](#_p3jsvz62h0j7) 6

[Threat analysis and monitoring](#_yeupf02cg53g) 8

[Secure design and implementation](#_iejkj8pgzky6) 9

[Misuse case diagram](#_y5u6p7khv65h) 9

[OWASP top 10](#_uh7nvci21gnm) 10

[Security verification and testing](#_aquda1oscm8v) 12

[**Technologies Selection**](#_ortrbpg268ws) **13**

# Case Study

## **Case goal:**

The goal of this case study is to get more familiar Security within the software domain. Knowing that we won’t be an expert on security, we can however, prepare for e.g. the most popular attacks as per OWASP top 10.

Our task is to come up with a 'Security by Design Plan', in which is defined what security related activities should be done in the different phases of the software development lifecycle.

Some of the questions that we will ask ourselves will be answered and visible in the brainstorm.

## Case questions**:**

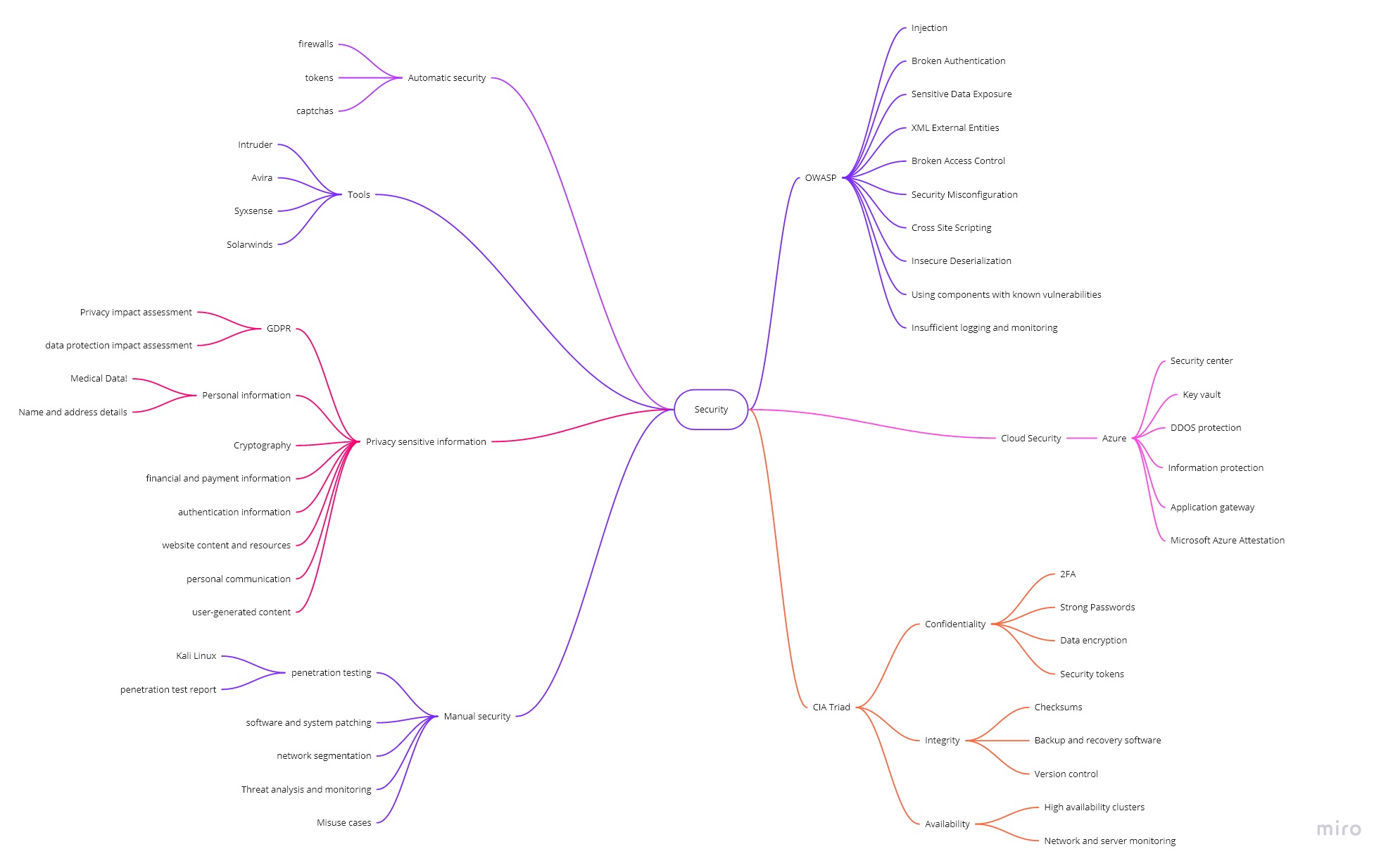
### Questions to be mindful of**:**

* What are the OWASP top 10 maturity levels?
* Which types of Security tools are there?
* How does cloud security differ from local security?
* How much regarding security can be done automatically?
* How is a security plan set up?
* How do you test security? (E.g. passwords strength, and token usage)
* What is privacy sensitive information?
* How do we protect privacy sensitive information in the database?

## Case purposes:

* Write the plan in such a way that it can be directly used in the group project
* Add security related tasks to the product backlog (make sure your PO is involved in this process)
* Decide what supporting technology can also be used for your group project, and motivate why.

# **Brainstorm**



# Security Related tasks

## Jira tasks related to security

|  |  |
| --- | --- |
| **ID** | **Security task** |
| **SEC** | Users can Authenticate using a 3rd party authentication such as Google. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 

## 

# Security by Design p**lan**

## Security requirements

### Security requirements table

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Requirement description** | **Comments** | **Priority** |
| Authentication | The system shall have authentication measures at all entry points, front-end or inbound network connection | To avoid unauthorized access | 3 |
| For a more secure communication JWT (JSONWebTokens) should be used | These tokens help to securely transmit who is asking for the information and what information is required. In addition these tokens allow you to verify that the content has not been tampered with. | 2 |
| Authorization | Users have different roles based on their needs and demands from the product. Buyers/sellers | Make sure the wrong user doesn’t get access to things he/she is not supposed to. (Grid maintainer doesn’t get access to seller/buyer) | 1 |
| The system shall support multi-level system access | Security issues as “bypassing permissions” to be mitigated | 1 |
| Availability | The backup system shall store the recover data in a network system | To help in case of failure or intruder action | 1 |
| Use network or server monitoring system | For detection of devices and other elements that comprise or touch the network | 1 |
| Application | Use the latest available external or third-party components | These should have detailed description of their current status (tested/vulnerabilities/etc) | 3 |
| Integrity | Validate application input | Prevents improperly formed data from entering the system | 1 |
| Confidentiality | Encrypt authentication and authorization mechanisms | Preventing attacker from gaining useful information that the attacker does not already possess | 2 |
| Encrypt data at rest in databases | Preventing attacker from gaining useful information | 1 |
| Encrypt data at transit in requests | Preventing attacker from gaining useful information | 1 |
| Keep access control lists and other file permission up to date | Preventing attacker from altering files remotely | 1 |
| Auditing | The system shall implement system hardware logging | Define more specific security loggings to allow recreating a clear picture of security events | 1 |
|  |  |  |

### 

## Threat analysis and monitoring

* *Who might want to attack or abuse the system, and with what motivation and goal? These attackers are also referred to as threat actors.*

Cyber ​​criminals may find it attractive to get details of large energy networks through our system. With this they can / want to track down bank details from our system and / or track down data in linked databases. This attack can expose data such as: transaction details, sensitive personal data and bank details.

* *What is the risk and possible impact of these attacks and abuse?*

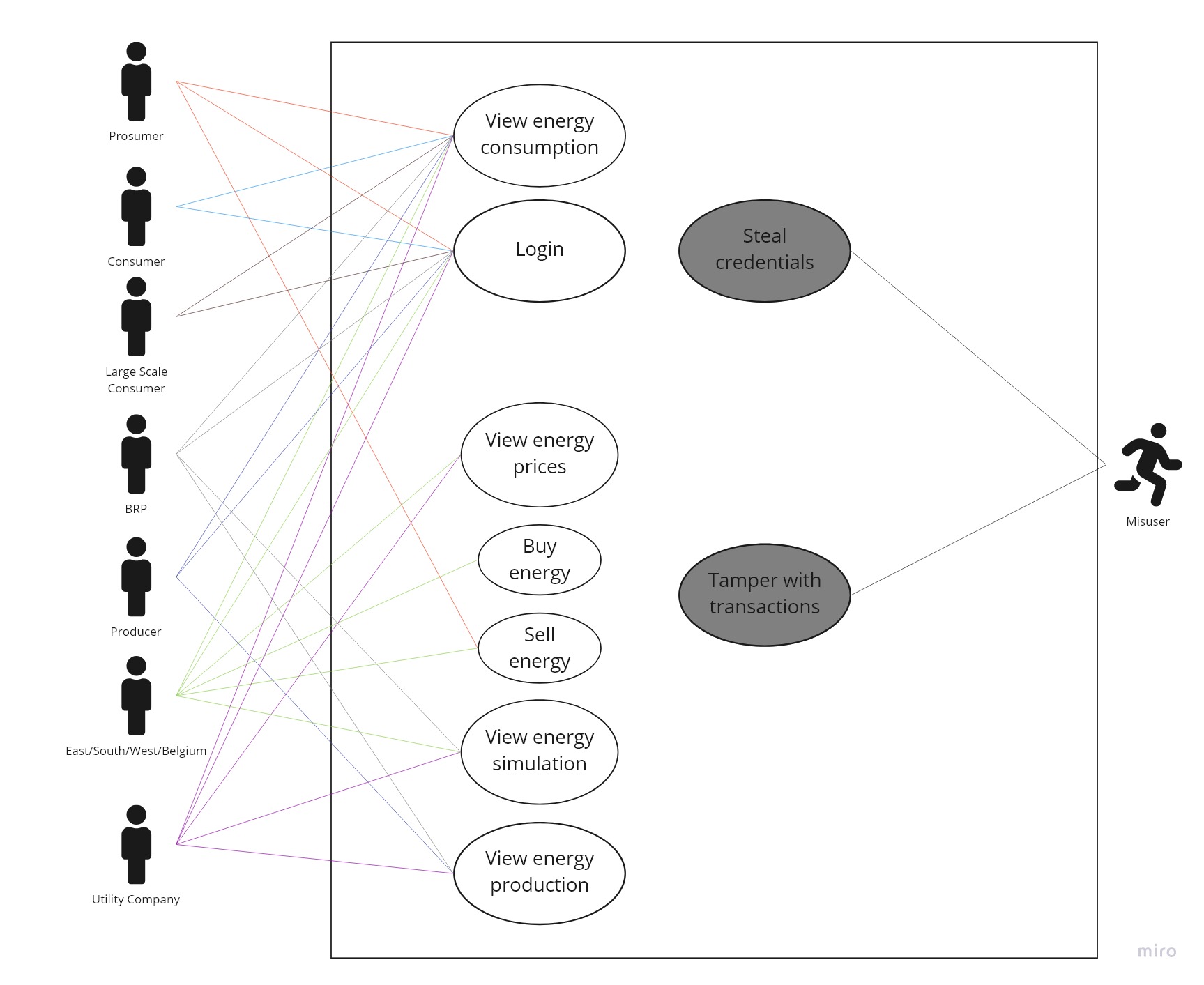
We are fully liable if sensitive data is leaked through our system. We are also responsible for a secure connection between external systems. It may be that an attack point arises in those connections.

* *What attack scenario's and hacking techniques can be used for these attacks and abuse? Work out the attack scenario's per threat actor in an attack tree.*

## 

## Secure design and implementation

### Misuse case diagram

**

Within our application we have to be wary of at least 2 user interactions that handle sensitive data. These are the login credentials and transaction page. The login page deals with different kinds of users that hold different types of data, one more sensitive than the other as the user roles differ between them. The energy transactions between companies or regions may hold sensitive information regarding company bank details which we should also hold in high regard.

### 

### OWASP top 10

The top 10 security risks according to OWASP will be taken into account while developing our application:

### 

|  |  |
| --- | --- |
| **Risk** | **Handling** |
| **A1:2017 Injection** | ReacJS handles embedded values in JSX by converting input to a string before it’s being rendered. It’s only a security risk when the following setting is being used: : ‘dangerouslySetInnerHTML’. React also doesn’t handle this option when you give the user control over an *href* or *a-link,* as an injection point can be opened from there. During development we will be able to keep note of these options.  In terms of browser parameters, they won’t be able to be altered as you have to be authorized to get the different types of data you want. |
|
| **A2:2017 Broken Authentication** | We will be using industry standards authentication options such as 2FA and authenticated requests. We can also explore the usage of 3rd party authentication applications for normal users which also conform to certain authentication standards. |
| **A3:2017 Sensitive Data Exposure** | Sensitive data will be held in different places according to our microservice architecture. The data will also encrypted which won’t allow an attacker to have the data in plain text. |
| **A4:2017 XML External Entities (XXE)** | Within our application, communication will almost certainly all be held in JSON format if it’s about data. |
| **A5:2017 Broken Access Control** | Data will only be available if ownership over that data is in accordance to the user’s availability. With use of JTW tokens, they will e.g. be invalidated on the server after logout. |
| **A6:2017 Security Misconfiguration** | Different credentials for each microservices deployment, any and all excess frameworks will not be used as the configuration will be given from the docker files with the correct installation features needed. In terms of React deployment we will disable the ’GENERATE\_SOURCEMAP |
| **A7:2017 Cross-Site Scripting (XSS)** | ReactJS handles XSS attacks by design. |
| **A8:2017 Insecure Deserialization** | Using JWT tokens we can check where data came from and if they’re allowed to pass through our system. We can also monitor if a user is repeatedly deserializing which allows us to check in. |
| **A9:2017 Using Components with Known Vulnerabilities** | Only use the latest safe versions of frameworks/libraries. By keeping these up to date with the latest protocols we won’t be bothered with 3rd party ways into our system. |
| **A10:2017 Insufficient Logging Monitoring** | Our application will feature a logging/monitoring system that will keep an eye on all of our services. These logs should also be easily digestible for future reference in order to determine if a user is acting maliciously or not. |

## Security verification and testing

*Define and execute testcases (e.g. unit tests) for all your security functionalities (as defined in your secure design) and for all your threats (as defined in your threat analysis). Side note: Consider the use of fuzzing (Links to an external site.) in your tests. Other security verification that can be performed are:*

* *Code review (also by external reviewers)*
* *Security testing by external testers (e.g. penetration testing)*
* *Use automated security scanner tools (e.g. OWASP Zed Attack Proxy (ZAP) (Links to an external site.))*

It is important to check your software for security risks. This can be done in different ways and using more methods usually is better than doing only one method. When developing the software it is important for developers to follow best practices and follow security checklists to make sure you are doing everything as secure as needed. Tools that help developers accomplish this can be very different from one another, this can be manual code reviews or some integrated system tests to hiring an outsider to try and compromise your software.

The next stage after implementation is to test the implemented software. The purpose of this stage is to discover and correct application errors. This includes running automatic and manual tests, identifying issues, and fixing them. Some recommended practises for this stage are:

* **Dynamic scanning**

Dynamic application scanner tools (DAST) expose vulnerabilities by simulating hacker attacks at runtime. To reduce false positives, you can use a combined approach (IAST). This approach complements runtime scanning with monitoring of executed code and application data flow. Dynamic scanning also pinpoints configuration errors that impact security.

* **Fuzzing**

This is a method to test the response of the application to malformed requests. This can improve security against SQL injection.

* **Penetration tests**

It is a good idea to invite a third-party team of security professionals to simulate possible attacks. External experts rely on their knowledge and intuition to reproduce attack scenarios that might be overlooked by your team.

# Technologies Selection

## 

|  |  |
| --- | --- |
| **Tool/Framework** | **Motivation** |
| SonarQube | Sonarqube offers security analysis and warnings where there could be security risks in our code. |
| Spring Security | Fits with our use of Spring Boot and allows for a wide variety of configuration usage regarding authentication. |
| LGTM | LGTM is a free and open source code reviewer that allows for security checking on commits, and can also perform code quality checks. Works together with the Semmle Security Research Team. |
| Code review | Having multiple developers look at the code is good. |
| GitHub Security | Reports vulnerabilities for the repository, disclose security advisories for the repository, and notifies when one of your dependencies has a vulnerability. |
| Docker Vulnerability Scanning | Allows developers and development teams to review the security state of the container images and take actions to fix issues identified during the scan, |
| JWT tokens | Tokens provide secure data transferring between the users and the application. This will also help with authenticated requests so sensitive data is only given to the correct user. |
| Auth0/3rd party Authorization | Possible usage for 3rd party authorization can be used for normal users so that they won’t have to create another account for logging into our application. |

# 