Security Report Shotmaniacs Team 1

This report is about the security analysis of the project, as well as the methods that will still be added later. Not all threats are yet being handled in the current implementation. For these vulnerabilities, possibilities are given how the web application can be protected with the idea that these will be implemented in the upcoming weeks.

A picture containing sketch, drawing, origami, design

Description automatically generated

# Table of Contents

[**Table of Contents 2**](#_heading=h.owbyn51l3b09)

[**Security Threats Prevention/Mitigation 3**](#_heading=h.rdlz5u3t7gf)

[XSS (Reflected) 3](#_heading=h.7x2sklk9v6vi)

[XSS (Stored) 3](#_heading=h.rhv5z0m3oudr)

[XSS 3](#_heading=h.n1123zsvhkkw)

[SQLi 3](#_heading=h.7g9pz3p08msw)

[Path Traversal 3](#_heading=h.whxuff2izhlf)

[Authentication 4](#_heading=h.954vyen2dstu)

[Salting 4](#_heading=h.91jleehsybz7)

[Authorisation 4](#_heading=h.974w7gexqhdt)

# Security Threats Prevention/Mitigation

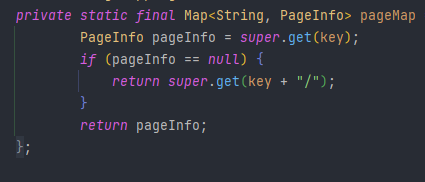
## XSS

Currently the web application is not fully protected against reflected nor stored XSS attacks. The crew’s calendar page is the only page that is protected now. For the other page, these are the plans that can be implemented in the next weeks to negate these attacks.   
For stored XSS, each field that a user can write something in will be checked **recursively** for the two strings ‘<script>’ and ‘</script>’.   
The same method can be used for reflected XSS with the URL of the webpage as a parameter, which checks whether code is written in the parameter.

## SQLi

For **all** SQL statements, the code uses PreparedStatements. The database does not currently use any stored procedures. These don’t necessarily have security benefits as prepared statements are sufficient when talking about security, but it will be more efficient to add stored procedures for efficiency reasons.

## Path Traversal

Path traversal is prevented by putting all the html files within the WEB-INF folder. This way, the code can use a page mapper java class to securely map all the pages to the right files. 

# Authentication

## Salting

The authentication is being done by a secure password checking. The password should be stored with a salt in the database. When the username or email address is queried, it will be checked by salting it again. This way, the passwords are secured better. The code is hashed using Script, which is known to be secure and not outdated.

## Authorisation

Throughout the project, there are three different groups. Clients, crew members and the admin. Each group has different permissions. This is done by a ContainerRequestFilter class and annotations. It acts as middleware. As a result, each time that the @RolesAllows annotation is there. It will check whether the cookie in the request is part of the role. If it does not work, it will return a HTTP 401 (not allowed). 