**OWASP Report**

Lyutfi Ismail

Date: 13/12/2021

# Broken Access Control

Is the Application Vulnerable?

Access control enforces policy such that users cannot act outside of their intended permissions. In the application, when implementing the authorisation, we made the user not able access the actions that require a role other that what they are assigned. There is a check that happens before a user can reach an endpoint, which states that a certain functionality is only permitted to given Authority level.

Also, the information of the user can be changed either by the admin or the user, and the password can be changed only by the user themselves

We also use Cross Origin check, where we state that only a given URL can receive information or send requests to the API. Accessing API with missing access controls for POST, PUT and DELETE is not applied, which means there is no vulnerability.

# Cryptographic failures

Is the Application Vulnerable?

The data of the website will be transferred using HTTPS protocol, which ensures that the information sent information is encrypted and only accessible to the client for which it is intended for. One thing is that password of the user uses simple hashes to store everyone’s passwords. The password should be encoded once, for which I use Encoder implemented in Spring boot, and a random token is generated each time the password is created or changed.

# Injection

Is the Application Vulnerable?

In the application the user-data is validated, so it makes it resistant to attacks. Hibernate Query Language (HQL) is used in the code and the data is retrieved by using mapping of a whole object or path variable. Outside data is not directly used or concatenated by using SQL or command that may contain both structure and hostile data in dynamic queries, commands, or stored procedures.

The frontend framework used in the project is React.js and according to the research made, it prevents from any embedded value in JSX by escaping anything that is not explicitly written in the application. That will prevent any potential unsafe areas when using the built-in components of the framework. However, caution must be raised when using any outside data to be displayed directly on the website. Currently there seems to be no area for malicious use of XSS.

Hence, the app is not that vulnerable and cannot be broken into by attacking the queries or the front-end code directly.

# Insecure design

Is the Application Vulnerable?

This category considers the use of secure design, when building an application, because even with perfect implementation, a design flaw can still be a point of vulnerably.

In our case, the application uses frameworks that have been tested and used by many developers. This provides, together with research into the frameworks themselves, a belief that the underlying technology has a safe design. With that safe design and using methods of implementation approved by the industry, we can provide a secure application, at least by design.

# Security Misconfiguration

Is the Application Vulnerable?

To protect the application, we use the built-in functionality of SpringBoot for security. The usage of different roles to enable only certain functionality to any role. Moreover, I prevent the security misconfiguration is prevented as authentication endpoints by default.

XML External Entities attack shouldn’t be an option with the current application as the frameworks used are not using xml as their mark up. Although more research can be done to prevent any potential threat in this category.

# Vulnerable and outdated components

Is the Application Vulnerable?

The application uses libraries, components, modules, APIs, dependencies etc., that are the most used in their domain and has been recommended by Fontys. These components all get updates through the year to ensure the latest security features are implemented. On the application side, to keep up with the newest version of the components, frequent check-ups must be done to ensure the usage of the latest and safest libraries, components, modules, APIs, dependencies etc.,

# Identification and Authentication Failures

Is the Application Vulnerable?

The software implements a secure way of identifying a user’s identity(authentication). The application stores any passwords values after they have been strongly salted and are not access in any form other than their hashed version.

The session and authentication token can be managed better since most of the handling of it happens in internal built-in functionality of SpringBoot. However, what can be changed to provide more security is the use of Cookie stored session tokens. The current implementation uses local storage on the client side, which in recent years has been proven to be not the safest wat of doing so.

# Software and data integrity failures

Is the Application Vulnerable?

The usage of trusted frameworks and dependency managers is needed to guarantee the absence of unverified libraries and any other dependencies. Both Gradle and NPM are trusted to provide verified secure software. However, when any new outside libraries are introduced, the job becomes much more difficult to ensure software integrity failures.

Together with that, this category encompasses the use of insecure deserialisation of data. To ensure the data integrity, the sent and received data must be properly serialised by trusted source and check for tampering by checking integrity before the usage of it. In the current context of the application, more work can be done to guarantee the data integrity used in the software.

# Security logging and monitoring failures

Is the Application Vulnerable?

This category concerns the use of logging of important events that can help navigate when a potential breach of security happens. In the use case of the current application, the small-scale deployment will not be implementation proper logging of errors and events. This can be improved when thinking of larger scale uses for the application.

# Server-side request forgery

Is the Application Vulnerable?

In the software application, we don’t access any outside service that can be manipulated with SSRF. However, the use of outside service is not out of the question and has to be carefully regarded if a potential need for one is required.