Security requirements

# Contexts

The security requirements have been inspired and based on the [ASVS requirements](C://Users/victo/Downloads/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf). Not only do I look into which security requirements best fit the Care project but also create user stories and misuse diagrams based on these requirements and how attackers can attack the system.

|  |  |  |
| --- | --- | --- |
| # | Description | Applying |
| 1.1.1 | Verify the use of a secure software development lifecycle that addresses security in all stages of development. | By making a checklist for adding security in each development lifecycle (analysis, design, implementation, testing) and applying this can secure software development lifecycle be addressed at each development stage. |
| 1.1.2 | Verify the use of threat modeling for every design change or sprint planning to identify threats, plan for countermeasures, facilitate appropriate risk responses, and guide security testing. | Creating for example misuse diagrams of attackers. |
| 1.1.7 | Verify availability of a secure coding checklist, security requirements, guideline, or policy to all developers and testers. | Creating a secure coding checklist, security requirements, and guidelines to all developers and testers. |
| 1.2.2 | Verify that communications between application components, including APIs, middleware and data layers, are authenticated. Components should have the least necessary privileges needed. | Applying authentication in communication between components (example; APIs) and not components of the application. |
| 1.2.3 | Verify that the application uses a single vetted authentication mechanism that is known to be secure, can be extended to include strong authentication, and has sufficient logging and monitoring to detect account abuse or breaches. | Researching and verifying which authentication mechanisms have secure authentication and can be extended to stronger authentications. |
| 1.4.1 | Verify that trusted enforcement points, such as access control gateways, servers, and serverless functions, enforce access controls. Never enforce access controls on the client. | Access controls are enforced in servers or gateways. |
| 1.11.1 | Verify the definition and documentation of all application components in terms of the business or security functions they provide. | Using terms for all application components based on their security or business functionality. |
| 1.14.4 | Verify that the build pipeline contains a build step to automatically build and verify the secure deployment of the application, particularly if the application infrastructure is software defined, such as cloud environment build scripts. | Using git and integrating automated security tests for integration and building. |
| 2.1.1 | Verify that user set passwords are at least 12 characters in length (after multiple spaces are combined). | Implementing the logging side that passwords are at least 12 characters in length. |
| 2.1.2 | Verify that passwords of at least 64 characters are permitted, and that passwords of more than 128 characters are denied. | Implementing the logging side that passwords are at most 64 characters in length. |
| 2.1.5 | Verify users can change their password | Implementing the possibility that a user can change their password. |
| 2.1.6 | Verify that password change functionality requires the user's current and new password. | Implementing the functionality that password change requires the user’s current and new password. |
| 5.1.4 | Verify that structured data is strongly typed and validated against a defined schema including allowed characters, length and pattern (e.g. credit card numbers, e-mail addresses, telephone numbers, or validating that two related fields are reasonable, such as checking that suburb and zip/postcode match). | Implementing checks for structed input such as e-mail addresses and telephone numbers. |
| 6.3.2 | Verify that random GUIDs are created using the GUID v4 algorithm, and a Cryptographically secure Pseudo-random Number Generator (CSPRNG). GUIDs created using other pseudo-random number generators may be predictable. | Implementing that when GUIDs are made are they made with using GUID v4 algorithm. |
| 7.4.2 | Verify that exception handling (or a functional equivalent) is used across the codebase to account for expected and unexpected error conditions. | Implementing exception handlers when needed across the codebase to handle expected and unexpected error conditions. |
| 8.3.2 | Verify that users have a method to remove or export their data on demand. | Implementing methods where users can remove their data on demand. |
| 8.3.8 | Verify that sensitive personal information is subject to data retention classification, such that old or out of date data is deleted automatically, on a schedule, or as the situation requires. | Implementing automation of deletion of old or out of date data on schedule and as the situation requires. |
| 14.1.1 | Verify that the application build and deployment processes are performed in a secure and repeatable way, such as CI / CD automation, automated configuration management, and automated deployment scripts. | Implementing secure and repeated build and deployment process in the CI /CD automation pipeline. |

# User stories & Misuse cases

|  |  |  |
| --- | --- | --- |
| # | Cases | User Story / Misuse Case |
| 1.1.1 | As a developer, I have added security practices in each stage of the software development lifecycle to be able to build a secure system. | User Story |
| 1.1.1 | As an attacker, I can find a non-secure development of the system where I can misuse the system. | Misuse Case |
| 1.1.2 | As a developer, I create and model designs of threats to be able to identify them and plan how to prevent these threats. | User Story |
| 1.1.2 | As an attacker, I can try to attack the system at every new release to see if there are not any secure implementations. | Misuse Case |
| 1.1.7 | As a developer, I have created a secure coding checklist, security requirements and guideline to all developers and testers to be able to apply the same security checks and reassure consistency of security across development. | User Story |
| 1.2.2 | As a developer, I have implemented authentication in APIs and not the components of the application to be to make sure the correct access is given to each user. | User Story |
| 1.2.2 | As an attacker, I am attacking the application components for vulnerabilities to see what access I need to get information for personal misuse. | Misuse Case |
| 1.2.3 | As a developer, I have chosen an authentication mechanism that is secure and can be extended to strong authentication to prevent easily breaches by attackers. | User Story |
| 1.2.3 | As an attacker, I have found out that the system does not use a strong authentication system where I can abuse and breach user accounts and the system. | Misuse Case |
| 1.4.1 | As a developer, I have enforced access control in the gateway and servers to be able to avoid the client from enforcing access controls. | User Story |
| 1.4.1 | As a user, I have the correct credentials and roles to certain access of the system. |  |
| 1.11. | As a developer, I have used business and security terms for all application components for consistency of documentation and definition. | User Story |
| 1.14.4 | As a developer, I have created security tests and implemented in CI/CD pipeline whenever a project is built to reassure security of the code. | User Story |
| 2.1.5 | As a user, I am able to change my password. | User Story |
| 5.1.4 | As a user, I to type in my phone number and email with knowing that my information is valid for the system. |  |
| 5.1.4 | As an attacker, I can create false data to make an account in the system. | Misuse Case |
| 6.3.2 | As a developer, I have used the GUID v4 algorithm to generate random GUIDs for more secure and not predictable GUIDs | User Story |
| 7.4.2 | As a developer, I have implemented exception handling across the code base to for come errors occurring that can crash or go wrong in the system. | User Story |
| 8.3.2 | As a developer, I have implemented methods where users can remove their own data. | User Story |
| 8.3.2 | As a user, I am able to remove private data I do not want to show or change my personal data. | User Story |
| 8.3.8 | As a developer, I have implemented automation of deletion of out of date and required situation of user’s data. | User Story |
| 14.1.1 | As a developer, I have implemented in the CI / CD deployment pipeline secure and repeated procedures for building and deploying the application so that it can be build and deployed in a secure manner. | User Story |

# Source

*OWASP\_Application\_Security\_Verification\_Standard\_4.0*. (n.d.). Retrieved March 1, 2019, from https://owasp.org/www-pdf-archive/OWASP\_Application\_Security\_Verification\_Standard\_4.0-en.pdf