**Hiking world**



Nikolaj Ørberg Hansen, Matti Andreas Nielsen, Victor-Emil Rossil Andersen & Vilius Janulis

Web security 07/04/16

**Brief description**

Our web application is a forum for hikers. Our users should be able to upload hiking routes from Google maps, to a Facebook like wall. The users can write comments to the hiking routes. The application will have a real time chat for the users.

**User stories**

· As a user I want to be able to register to the application, so that I can upload hiking routes and comments.

· As a user I want to be able to upload a profile picture, so that I can personalize my profile.

· As a user I want to be able to upload hiking routes, so that I can share it with other users.

· As a user I want to be able to comment hiking routes, so that I can give my opinion.

· As a user I want to be able to chat with other users, so that I can have private conversations.

**Web security principles**

**Grant least privileges**

The reason why we chose this as one of the security principles is its wide amount of touchpoints and application use across the entire platform, from application to database to operating system, when you talk about privileges it is also a volatile thing in the sense that it can stop people from doing their job, and it can stop hackers from doing hurtful things which we think is an interesting balance.

On the application level when we make our restful endpoints, we have to evaluate which tables and columns of the mysql database that specific endpoint needs access to. That means that if a specific endpoint is compromised we’re able to limit the contamination to a small amount of tables, to implement this requires us to create a mysql user for every endpoint that differs in access from another.

On OS level, the processes that hosts the application should have the correct file access rights, e.g. it shouldn’t have access to delete a configuration file, basically it should just have read rights for most of the files.

When the developers of the project ssh to our remote server, they should have limited rights, to implement this we have to create a user for every team member. Each user should have its own private and public key, in contrary to using password authentication on the server. Which also means that if a key gets compromised and we close the user down, everybody else will still be able to ssh to the server. The users should also have limited rights to the OS itself, e.g. no one should have permission to su to root.

**Fail securely**

Handling errors securely is a key aspect of secure coding. We are planning on handling all possible exceptions in our applications. By doing that we will prevent system crashes and exposing technical or possibly harmful information to end users.

Our system will include error handling and proper error message display for end users for all or most of the possible errors, which in our current stage of the project includes error handling for user registration, user login and similar member based use cases.

For upcoming stages we also planing to use same security principles for error handling on other areas of the application, like authentications, restfull web-server requests and etc.

Along standard error handling for specific methods and functions we will incorporate solution wide error handling and automatic application restart functions on fail. This is done by using additional tools like foreverjs or core implementation of similar functionality, that will allow us to have most stable and reliable solution along the web security principals for error handling. This tool or functionality will allow us to cache all previously in code un-cached exceptions, log them and restart application for future use.

Apart handling errors and preventing unintended crashes of the application we are planning on introducing error log functionality that will allow us to identify and track possible error better and fix them faster for future use of the system.

We will code our application in the way that it would prevent unauthorized access to specific functions and actions in the system, by using these security principals.

The main idea behind this ‘fail securely’ principal in our applications is to handle exceptions, prevent crashes, fail securely, and log errors.

We chose this web security principal to be used in our application, because it prevents potentially very harmful and insecure actions that would disable our application and compromise data.

**Mediate completely**

We will focus on the Mediate completely design principle, because we want to make sure that every access and every object is checked every time. We will do this to always make sure that a user are authenticated and have the required permissions. We will validate the users with a token that is sent every time. we will also check every object every time to make sure that a user is not trying to make a script injection.