Caren (Nedap) - Team 2
Security Analysis
01/ July 2020

Protecting the customer

Security is important when developing a digital solution. That is also the case for the Caren project where it was implemented in three main areas of the project and they are input fields, prepared statements and stored procedures and finally storage of passwords.

Input fields

As we started designing, from the get go, we realised we would have many input fields, due to the nature of the project, and this can make us more vulnerable to attacks, specifically *SQL-injections* since most input fields go directly to the statements used for queries. What we did to prevent that was use *sanitisation* as a means to ensuring that whatever is inputted in the field does not contain symbols that can potentially be used for SQL Injection. Additionally, sanitisation also protects against cross-site scripting since, again the input fields do not allow specific symbols to be used.

However, we realised that is not enough to prevent SQL-Injection, so we also used *prepared statements* for any query used in our code.

Prepared statements and stored procedures

Prepared statements are an elementary step we took in enforcing our database against *SQL-injections*. All database queries made through JDBC are made using prepared statements which help protect the database in case any user input gets past our *sanitization* process.

Moreover, stored procedures prevent the delivery of raw SQL to the database. This prevents any tempering of queries that can occur during the networking phase of delivering them across as well as protect the database in case the source code is leaked.

Password storage

Amongst the most prominent and frequently used types of attacks that are carried out against digital platforms are database breaches and the leaking of data found within. To protect Nedap against such attacks, we opted to use SHA-512 as the hashing algorithm to encode passwords stored in the database. This strategy allows users with uncommon passwords -passwords that are not found in a dictionary- to remain protected despite any kind of database breach as hashing algorithms are one way and thus cannot be reversed.

However, in case the user uses common passwords such as “hello” or “password”, the hash string can always be compared to a precomputed table of hash strings to decrypt it.