**Password hashing and protection**

Without any sort of hashing or protection if unwanted users got access to the database they would have all users username and password in plaintext ready to hack into their account. Which is where hashing comes in and turns a plaintext password into a long string of characters that looks random in which the unwanted user can’t decipher or reverse engineer the password by applying a mathematical algorithm to the plaintext.Which should work great however when it comes to protection protocols they should follow Kerckhoff's principle "one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them". This essentially means that a system shouldn’t be secure through obscurity, i.e even if the unwanted users know the algorithms/protocols that encrypt the data, they still shouldn’t be able to break the security. Hence if the unwanted user knows the hashing algorithm being used, if the only protection is the hashing then the unwanted user can brute force attack and/or create a table of words and letters and their hash output which they can then check with the hash in the database if they got access to it. This where salts come in, which is a randomly generated string prepended or appended to the password before hashing when the password is created, then the salt is stored in the database. This helps prevent against these pre-calculated hash tables because the hacker has to then recalculate these tables for each user they wish to hack into as they will have their own unique salt with them which makes these attacks more costly. And then for brute force attacks can never be fully stopped but with complex hashing algorithm the time taken to brute force through becomes more and more expensive and so with modern hashing algorithm instead of making them fast, algorithms like argon2 have parameters such as memory size, iteration count, parallelism and salt length which can be adjusted to make the hash more (or less) complex to make it more expensive to brute force as it’s taking a longer time, using more memory and threads to compute the thread which when going after many users helps slow them down and deter hackers since it becomes less production efficient. This where argon2, which won a hashing competition in 2018, is best to use nowadays instead of old algorithms such as SHA\_1 or 2 or even bcrypt, it also creates its own salts and so don’t need to rely on another library to be secure.

1. En.wikipedia.org. 2020. *Kerckhoffs's Principle*. [online] Available at: <https://en.wikipedia.org/wiki/Kerckhoffs%27s\_principle> [Accessed 1 March 2020].
2. Information Security Stack Exchange. 2020. *Is My Developer's Home-Brew Password Security Right Or Wrong, And Why?*. [online] Available at: <https://security.stackexchange.com/questions/25585/is-my-developers-home-brew-password-security-right-or-wrong-and-why> [Accessed 1 March 2020].
3. Information Security Stack Exchange. 2020. *Is My Developer's Home-Brew Password Security Right Or Wrong, And Why?*. [online] Available at: <https://security.stackexchange.com/questions/25585/is-my-developers-home-brew-password-security-right-or-wrong-and-why> [Accessed 1 March 2020].