Security analysis for grade manager project. Group 19.

Over the last two weeks of the project we have been looking at the security of the web application.

Security issues which are considered are the following: Cross side scripting, SQL injection, encryption of passwords of users. Outside of our scope are issues which are related to the communication between the user and the server, such as whether we encrypt information which is sent from the client to the server (like a password).

Not all security issues are relevant in every page a user can access. This is because a user isn't necessarily able to access a page (i.e. a student cannot visit the manager page), and not every page has the same vulnerabilities.

With the login page, it is important to sanitize the user input or use a prepared (parameterized) query, because we make use of an SQL query to compare the submitted login information with our person database, where it would be possible inject SQL in order to login with incorrect credentials.

With the student page, we don't accept textual user input at all. So we don't need to sanitize input or use parameterized statements.

For teachers, we also don't use textual user input. So the teacher has to click in order to navigate. This means that just for the student page, we don't have to be worried about a teacher or a student that they will try cross side scripting.

The manager page does however pose some risks, as we make use of textual user input. However we assume that if someone was able to become a manager, he must be trustworthy and responsible, and whenever a manager is malicious, XSS would not be the primary concern, as he may as well go ahead and delete every user from the database at will anyways. Finally we wouldn't really see a reason for a manager to go ahead and try to hijack a session, making use of XSS, as he already has the most powerful role in the system.

As for session management, we make use of the standard implementation of request.getSession(), to create a session as well as getting the session id, by calling session.getId(). We trust that the session id's generated by this method meet the required conditions for a good session id, namely that it is meaningless as well as unpredictable. We query the database for this session id to get the logged in user id, if it exists.

For password encryption, we decided to hash a password + 16 random bytes as a salt, which are stored in another column. We of course had to choose a hash for this. To elaborate on the choice of the hashing algorithm, we consider the following:

-Using MD5 is a bad idea, not necessarily because of its cryptographic weakness but rather because it is really fast, so that it is quite easy to try a lot of passwords in a short period of time, even on a slow computer.

-We don't necessarily need the highest level of security, or the most advanced hashing algorithm.

The SHA-512 algorithm is not the safest choice, as there are algorithms which provide more advanced security such as PBKDF2, scrypt or bcrypt. Nevertheless, using SHA-512, we can provide quite secure encryption of passwords, given that we generate a random hash for every password aswell.

Unfortunately we have to keep some things out of our scope. This is because we don't have enough time to think about security, or do research about security, let aside implement it. But we consider our system moderately protected, to our knowledge. Additionally, some of the subjects we decided to keep out of our scope, were not part of the Data and Information module. The main goal of our implementation of security is to make an attempt to defend ourselves against the sort of vulnerabilities we have learnt about, and to show that we understand how we should prevent them.