Security analysis report

Group 22

- Jasper van Amerongen, s2384442
- Lola Solovyeva, s2207958
- Adamo Mariani, s2305399
- Nidanur Gunay, s2473593
- Albina Shynkar, s2207966
- Eda Yardım, s2472651

Software security analysis is undoubtedly an important part of the project because it enables the stakeholder to understand all of the risks regarding the software, the way they are prevented and to evaluate how secure the software is. First of all we thought about SQL injection prevention, as it is important to secure our data before entering the database. SQL injection prevention is better achieved using prepared statements, which are in our system and will be discussed later on. The next step to secure our system was cross-site scripting (or XSS) prevention, so that no javascript from the client could have affected the server.

- SQL injection prevention

The security of the database is a high priority for the project, no data should be vulnerable from within the database. Prepared Statements and input sanitization are ways to prevent SQL injection and in our project we have used both. User inputs are initially sanitized using the function called escapeSQL(), which is inherited by all Controller classes. In that function we are carefully sanitizing SQL queries by escaping all SQL-specific characters.

- ➢ In order to eliminate union attacks, prepared statements are used and user input is inserted in pre-defined ‘slots’ within all SQL queries

- ➢ The escapeSQL method searches for SQL characters in a string (“ ‘ ”, “\” etc.) and escapes them in order to prevent unwanted data insertion and deletion.

- XSS prevention

Cross-site scripting is another way of attacking a web application and it should be prevented. XSS enables attackers to inject client-side scripts into web pages viewed by other users, and this is prevented applying input sanitization. In our project we have implemented the escapeHtml() method in order to do so, and in that function we are carefully sanitizing potentially dangerous HTMLcode (e.g. HTML tags, hash mark, ampersands etc.).

- ➢ In order to prevent this kind of attack, we modify our query using that function to encode all the aforementioned characters in HTML format, so that they can be decoded once again when shown on the webpage, without them being interpreted as HTML code (or tags).