Security Analysis

GTA-VI - A Nedap Theft Dashboard

Antoine Moghaddar (S1880241)
Michael Raczkiewicz (S2524732)
Duru Koçak (S2486431)
Valeria Veravita (S2456036)
Sadat Ahmad (S2610736)

Module 4: Data & Information
University of Twente
2020-2021

Security Analysis

In terms of security, we have implemented a token system, hashing, and other methods to get rid of vulnerabilities that can be exploited by SQL injection and XSS.

LoginHandler.java class handles the logins and the passwords of our project and most of our security has been implemented there. The *login* method handles the login procedures and has a prepared statement that eliminates SQL injection. The password entered by the user is handled by adding the salt and hashing it. To hash the passwords we have used SHA512 in the *Hashing.java* class which returns the hash. If the hashed value is correct and matches the database, a token, permission, and timestamp are created in *login* which is added to the database. The timestamp enables the website to timeout and makes the token invalid after 15 minutes. This protects the website against cross-site forgery since the logged-in time of the user is limited. Another important detail is that while the token is stored locally, the permission is not. Therefore even if you can modify your local storage and your token, the permissions won't be changed.

While adding new users with the *newUser* method, we use the same security measures as the *login* method. prepared statements and the salt are similarly used. In this case of course, salt is being generated and saved in the database. We add an additional level of security, however, by using secure random instead of pure random while generating salt and tokens.

The *getStorId* method fetches the tokens and returns the storid and the permission level. We have also used prepared statements here to prevent SQL injection attacks by modifying tokens in the local storage.

deleteToken method also makes sure that the token is deleted from the database after logging out which prevents reloading the page after logging out.

Additionally, to prevent XSS, we made sure that the HTTP request is not directly in the URL but in its parameters. This prevented inserting malicious code into the URL which can be processed by our program.

In the end, the only security issue we have identified and still have is when you send the HTTP request the password is sent in plain text. Which makes the password vulnerable to attacks and information retrieval.