Security analysis

1. Approach

Our approach to security has been to secure the endpoints themselves instead of concerning ourselves with the transmission security since for the web app itself we will be using HTTPS so we need not concern ourselves with that. Thus we built an entirely custom authentication protocol to validate both the server, client and the request users make. Both the server and the client have their own set of RSA key pairs both are 2048 bit string and are generated using OpenSSL. There are also two variants of authentication, session-less and session-ed. First session-less.

2. Session-less Authentication

Session-less authorization means happens when the client has been loaded in a browser, but the user has not yet requested access to the dashboard. The servlets that use session-less authentication are ServIndex(Index of all servlets they labels and via which methods they are accessed), Login and UserCreation. These three services are needed for user to request to load the dashboard, make and account and of course for the frontend to know which services are available thus they cannot be accessed via within a session. This is also why the authentication protocol is checks the validity of the endpoint. For this upon a request the client sends in the headers a signed secret phrase called an app_key. This string is salted as follows ("+" + current_time + "+" + app_key) and then signed with the client private key and, along with the date used to concatenate the app_key, are both added to the headers and sent to the server.

Upon receiving the request the server first checks if the security headers are present and then the signed key is verified by taking the app_key from the server storage and concatenating it with the given date, then the client public key is used to verify it.Then the date is checked to see whether it is recent request after which the server returns the appropriate response along with it's own signed app_key this time signed with the server private key and following the same format

3. Session-ed Authentication

Session-ed authentication is when a user has already logged in thus having a session opened in the database. Session-ed authentication builds on top of the session-less authentication by adding what is called a session-token. This session-token is a JSON Web Signature that is generated by the server and handled only in the headers and not directly by the client. The JWS has two fields the sessionKey and id of user. Upon verifying the client the server then checks: if all three security headers are present, the given user id has a session open and that the session is still active. If any of these are false then a 401 error is sent, if all pass then the session expiration is set 30 minutes after the current date and the server returns the same values as before, but this time with the session-token

- Bozhidar Velinov - Veselin Daskalov
- Aamir Farooq - Frans de Boer
- Nikola Martino - Nick Shindler

4. **SQL Injection protection**

   To protect against SQL injections we have used only prepared statements for our server-to-database queries as well as closing all active connections, a feature of our servlet wrapper class.

5. **Data integrity**

   To protect against SQL injections we have used only prepared statements for our server-to-database queries as well as closing all active connections, a feature of our servlet wrapper class.