Chat sécurisé HERMES

Développeurs :

- Baptiste Hardelin : baptiste.hardelin.etu@univ-lille.fr
- Théo Ernould : theo.ernould.etu@univ-lille.fr

Descriptif :

Notre projet consiste, comme son nom l'indique, à mettre en relation plusieurs utilisateurs afin qu'ils puissent communiquer par le biais d'une Interface Homme Machine, en l'occurrence, un site web (HTML, CSS, Javascript). L'utilisateur doit tout d'abord se créer un compte qui sera stocké dans une base de données (SQLite côté serveur) avec un identifiant et un mot de passe, ce dernier sera haché avec un sel afin d'en garantir la sécurité. Puis, il devra s'authentifier à l'aide de ces deux derniers éléments pour arriver sur la page de discussion où figureront la liste des utilisateurs connectés avec qui il pourra dialoguer, ainsi qu'un chat général.

Les discussions privées (hors général donc) sont chiffrées avec l'algorithme de chiffrement asymétrique RSA-OAEP, lui-même utilisant l'algorithme de hachage RSA-256 avec une taille de clé de 4096 bits permettant de communiquer de façon sécurisée entre deux utilisateurs donnés, le serveur (NodeJS) n'a donc à aucun moment connaissance du contenu déchiffré du message car le chiffrement/déchiffrement a lieu côté client, il ne sert que de passerelle et les messages sont par ailleurs éphémères, à chaque redémarrage du serveur ils s'effacent, ainsi aucun message n'est stocké de façon persistante, évitant les actes malveillants. Notre site a une certification HTTPS (pas reconnue par une autorité de certification officielle) qui permet d'offrir un cran de sécurité en plus.

Enfin, nos utilisateurs sont protégés des injections XSS, des tentatives de vol d'identité par le blocage de l'accès à la page de discussion sans être authentifié et du fait de se connecter deux fois avec le même compte.

Difficultés rencontrées :

Au début, nous avons eu du mal à déterminer s'il fallait par exemple que les messages soient sauvegardés ou non, s'il fallait une authentification au lieu d'un usage anonyme et quel algorithme de chiffrement utiliser.

Néanmoins, par la suite nous avons appris de nombreux aspects de sécurité et mis en application certains du cours, comme l'algorithme RSA donc nous avions déjà connaissance ou bien les notions de sel et poivre que nous avons appris par des recherches, ou encore, la certification via OpenSSL et les injections XSS…

De plus, de nombreuses recherches nous ont permis de découvrir des outils disponibles via le catalogue NPM, comme Express pour la navigation entre les pages et PassportJS pour l'authentification.

Finalement, ce qui nous a donc plu a surtout été la recherche, la documentation, la mise en pratique des enseignements et l'aspect concret qu'apporte un projet. Toutefois, l'ambiguité dans la possibilité des technologies à utiliser au début du projet nous a un peu perturbés mais nous avons vite été fixés et avons pu développer le projet.

Etat d'avancement :

Ce qui fonctionne à la restitution du projet a été expliqué dans le descriptif, mais il reste quelques bugs, notamment quand on rafraichît la page et les messages sont parfois dans le désordre quand ils s'affichent.

Mode d'emploi :

1. Se placer à la racine du répertoire du projet.
2. Effectuer la ligne de commande suivante : npm i && npm start
3. Aller dans un navigateur quelconque et se connecter à l'adresse suivante : https://localhost:3000 ou https://[nommachine]:3000 à l'iut
4. Une page indiquant que la connexion n'est pas privée va s'afficher (à cause de l'autorité de certification qui n'est pas officielle pour le navigateur), cliquer sur « paramètres avancés » ou son équivalent et ensuite « continuer vers le site localhost »
5. Vous pouvez ensuite vous créer un compte, vous connecter, et en ouvrant une deuxième fenêtre et faire de même tester l'envoi de messages entre plusieurs utilisateurs.