Rapport : SealPass

CHAHI RABIE Ala Eddine, JACQUET Julien, BEN MALLEM Amir,
DEFLY Kossi Eloi Fabius

30 Avril 2020

SealPass est un coffre-fort de mots de passe gratuit en ligne qui permet aux utilisateurs d'héberger leurs différents mots de passe dans une base de données chiffrée et sécurisée.

Abstract

Dans un monde numérique en perpétuelle évolution jamais les exigences en matière de protection des données et des systèmes d'informations n'ont été aussi hautes, pour les utilisateurs les principaux garants de la sécurité de leurs données personnelles et de leurs vies privées sont les systèmes d'authentification et les gestionnaires d'accès tels que les mots de passe.

Depuis 2017, pas moins de 555 millions de mots de passe dérobés depuis des centaines de milliers de bases de données ont été publié sur le *dark web* par les pirates informatiques. Ces mots de passe sont disponibles en quelques cliques et permettent parfois un accès à tous les comptes liés.

En effet, un unique mot de passe est très souvent utilisé pour tous les services internet d'un utilisateur et c'est de cette habitude que découle les failles de sécurité les plus importantes. Il est par conséquent primordial de revoir cette pratique et d'essayer d'amorcer des systèmes intelligents permettant le stockage sécurisé de plusieurs mots de passe sans que l'utilisateur n'aie à les retenir. C'est pourquoi les coffre-forts numérique sont de plus en plus en vogue et commencent, à l'instar du Cloud, a devenir une nécessité pour utiliser le web.

Encore faut il que ces mots de passe stockés soit *forts* et résistants aux différentes attaques informatiques telles que les recherches exhaustives, les attaques par dictionnaires etc...

Alors comment créer des mots de passe sûr ? comment gérer des mots de passe sans les rendre vulnérables ? comment les stocker de manière à pouvoir les utiliser sans les mémoriser pour autant ?

Basé sur les recommendations de l'ANSSI [1], SealPass à pour pesant d'or de satisfaire tous ces besoins et se lance le défis de garantir un haut niveau de sécurité et une protection de la vie privée des utilisateurs.

1. Agence nationale de la sécurité des systèmes d'information

Table des matières

1 Description

1.1 Qu'est ce que SealPass :

Sealpass est une application web de gestion de mots de passe en ligne. Elle permet aux utilisateurs de se connecter et de récupérer leurs mots de passe précédemment stockés. SealPass offre également un service cloud et un portefeuille numérique de cartes de crédit.

1.2 Fonctionnalités

1.2.1 Inscription gratuite

SealPass est un gestionnaire de mots de passe gratuit accessible partout à travers le monde, aucune limite en matière de nombre de mots de passe stocker ou d'espace de stockage n'est imposé. Pour pouvoir en profiter il suffit de créer un compte avec une adresse email valide ainsi qu'un mot de passe et d'utiliser ces identifiants afin de se connecter.
Le mot de passe choisi lors de l'inscription est très important il doit être robuste car il permet d'accéder à tous les autres mots de passe stocker, il est fortement conseillé de suivre les recommandations de l'ANSSI afin de créer ce dernier.

1.2.2 Création et modification de mot de passe

La fonctionnalité principale de SealPass est le stockage de mots de passe, les utilisateurs ont donc a leur disposition une section où ils peuvent consulter ou ajouter de nouveaux mots de passe. Les utilisateurs peuvent bien évidemment ajouter l'identifiant associé ainsi que si nécessaire le site associé. Ils peuvent également modifier des mots de passe existants et en supprimer.

1.2.3 Stockage de mot de passe

Les mots de passe de nos utilisateurs sont stockés de manière sécurisée dans une base de donnée. Ceci garantit que les mots de passe sont accessibles aux utilisateurs depuis n'importe où, il leur suffit d'entrer leurs identifiants et ils auront accès à tous leurs mots de passe.

1.2.4 Génération de mot de passe

Pour aider nos utilisateurs dans le choix d'un mot de passe unique et sécurisé, notre application est livrée avec un générateur de mots de passe. Ce générateur suit les recommandations de l'ANSSI [2] pour la sécurité des mots de passe. Il est possible pour l'utilisateur de choisir la taille et le type de caractères (alphanumérique, numérique, ponctuation, majuscules, minuscules) souhaité.

1.2.5 Stockage de carte de crédit

SealPass permet également de stocker des informations de cartes de crédits, notamment le nom associé, la date d'expiration, et le CVC/CVV. Ces informations au même titre que les mots de passe sont ensuite sauvegardés de manière sécurisée dans la base de données.

1.2.6 Cloud et stockage de fichiers

SealPass fournit un service cloud pour stocker des informations jugées sensibles, ici les utilisateurs peuvent envoyer leurs fichiers vers la base de donnée, ces fichiers peuvent même être organiser dans des dossiers créé par les utilisateurs. Il est possible a tout instant de télécharger ou supprimer les fichiers stockés.

1.2.7 Plugin pour utiliser les mots de passe

L'application SealPass est livrée avec un plugin, ce plugin permet de compléter les mots de passe et identifiants associé au site sur lequel l'utilisateur se trouve.

1.2.8 Changement d'identifiants

Les utilisateurs peuvent changer les identifiants associé a leur compte par le biais d'une section "settings". Ils peuvent changer d'adresse e-mail et de mot de passe principal. Pour le mot de passe principal, leur ancien mot de passe leur est demandé, ils doivent également confirmer leur nouveau mot de passe.

2. ANSSI sécurité des mots de passe

2 Technologies

Nous présenterons dans ce qui suit le détail des technologies que nous avons utilisé dans notre application.

2.1 Front-end

Pour ce projet une bonne partie d'entre nous ne maîtrisait que le HTML, Css et le SQL. Nous souhaitions donc améliorer notre CV avec des technologies modernes et à la mode.

2.1.1 Javascript côté client :

Le choix de l'utilisation du JavaScript, parfois baptisé "Language de web", était évident. Nous avions attendu d'apprendre ce langage tout au long de notre cursus. Mais nous voulions aller encore plus loin, nous avons donc recherché quels "frameworks" et "librairies" étaient à la mode, nous avons comparé, Vue.js, AngularJS et ReactJS. Notre choix s'est porté sur ReactJS, Angular semblait trop compliqué à apprendre en plus de JavaScript, et React étant plus populaire que Vue.js nous avons été séduits.

Après avoir suivi un tutoriel JavaScript (un tutoriel différent pour chacun, afin d'apprendre le plus d'informations possible), nous nous sommes lancés sur le développement du front-end. Après un rapide tutoriel sur React, nous étions prêts.

2.1.2 ReactJs pour une apparence moderne :

Nous nous sommes servis de REACT pour avoir un affichage dynamique notamment au niveau du Cloud où on affiche une grille Css d'éléments (dossiers/fichiers) ces éléments sont des objets et leur affichage est dynamique (supprimer ou ajouter un objet modifie la grille). Avec React on se sert également du modal qui nous permet d'afficher une fenêtre flottante au-dessus de l'application, par exemple pour donner le nom d'un nouveau dossier ou importer un fichier.

React nous a également permis de réutiliser des Components déjà codés aux endroits propices. Par exemple un dossier contient la même structure d'affichage que la grille des fichiers, on a donc réutilisé le Component de cette grille dans le dossier.

Pour notre front-end la partie stylistique est assurée par notre propre CSS.

2.2 Back-end

2.2.1 NodeJs et AJAX pour maîtriser le serveur :

`NodeJs :`

La partie back-end de notre application web 'SealPass' a été réaliser avec NodeJs , ce choix n'est pas anodin, bien que seule une personne de l'équipe maitriser cet environnement d'exécution au départ, de nombreux arguments sont venus pencher la balance du coter de NodeJs au depend d'autres languages ou frameworks tel que Php ou Django (framework web de python) parmi ces arguments on retrouve :

— la capacité de NodeJs a supporté le traitement de données en temps réel en effet grace au moteur V8 utilisé par NodsJs qui fait de la compilation JIT(Juste in time) qui transforme le code Javascript très rapidement en code machine.

— L'architecture non bloquante de NodeJs est un véritable atout pour certaines fonctionnalités de notre API notamment le Cloud ou il peut y avoir des transferts de fichiers très volumineux cela évitera d'arrêter notre application le temps que le transfert soit effectué.

Un autre language comme le Php aurait très bien pu être utilisé d'autant plus que la bibliothèque cryptographique sodium est devenu une bibliothèque native au langage ce qui aurait facilité un bon nombre de choses étant donnée la nature de notre API mais, le déclin continue de son language au fil des années nous a éloigné de ce choix dans un souci de maintenance pour le futur de l'API, Ajoutons à cela la curiosité de l'ensemble de l'équipe apprendre NodeJs qui ne fait que monter en puissance au fil des ans ce qui apporte un plus-value à nos compétences au vu des tendances actuelles, bien que de ce point de vue 'tendances actuelles' le choix de Django le framework web de python aurait été pertinent mais NodeJs permet d'avoir une syntaxe homogène entre le front-end et le back-end ce qui est très avantageux au regard de la taille de l'équipe et au temps imparti à la réalisation de cette application web.

AJAX :

L'utilisation d'AJAX a été indispensable au développement de SealPass, indispensable pour l'envoi de requête depuis le client vers le serveur en arrière-plan et vice-versa.

L'utilisation de concept basé sur les techniques JSON, XML et Javascript a beaucoup apporté à notre application la rendant plus dynamique nous évitons ainsi de recharger entièrement nos pages.

Cependant l'utilisation d'AJAX s'accompagne de certaines bonnes pratiques à ne pas négliger tel que l'utilisation de POST au lieu de GET, ne pas envoyer de données sensibles à partir du serveur et l'utilisation des nonces, combiné AJAX à d'autres techniques de protection rendent les attaques de vol de sessions et de cookies moins efficaces.

Afin de réaliser des tests au fur et mesure de leur avancer l'équipe back-end a dû s'appuyer sur postman.

2.2.2 MongoDb : une BD NoSql facile d'accès et efficace

Pour débuter la création de SealPass il reste encore un choix à faire, celui du type de base de données un SGBD relationnelle (SGBDR) ou un SGBD NoSql, nous avons décidé de nous appuyer sur une base de données NoSql et cela pour une raison majeure une fonctionnalité de SealPass qui est le Cloud, en effet après plusieurs recherches nous avons constaté que le stockage de fichier directement dans une base de données relationnelles était fortement déconseillé pour un service tel que le Cloud, cela affectait considérablement les performances de la DB.

Pour cette raison nous avons finalement opté pour MangoDB un système de gestion de base de données orientés documents.

SealPass est le type d'application web où il aurait été parfait d'avoir une base de données hybride une combinaison entre un SGBDR et un SGBD NoSql, mais le temps jouait en notre défaveur il a donc fallu trancher, cette hybridation reste envisageable pour les perspectives d'amélioration.

2.3 Sécurité

La réalisation de SealPass se déroule dans le cadre du module application web et sécurité et, comme l'intitulé du module l'indique l'aspect sécurité important dans notre API, il était donc nécessaire de faire une application web qui peut résister aux attaques

les plus courantes et qui répond standards de sécurité actuels. Pour cela nous avons défini les attaques de bases auxquelles SealPass devait résister afin de les contrer, parmi ces attaques on retrouve les injections Sql, XSS, CSRF et le vol de session.

Nous avons choisi dès le départ de partir sur Express et Ajax pour partir sur de bonnes de bases de sécurité. Nous détaillerons dans ce qui suit les mesures prises à cet effet.

2.3.1 TLS

SealPass utilise énormément de requêtes HTTP, c'est la raison pour laquelle nous avons dû penser a un protocole de sécurisation de donnée et nous nous sommes penché sur TLS pour assurer la confidentialité, l'intégrité et l'authentification. `TLS en pratique sur SealPass :`

Nous avons tenter d'appliquer au mieux les directives du protocole en l'occurrence :

1. le chiffrement de données sensibles s'est fait à l'aide de l'algorithme cryptographique à clé secrète l'AES (en mode opératoire CBC).

2. les mots de passe sont automatiquement hashé à l'aide de la bibliothèque Bcrypt qui utilise des fonctions de hachages à jour et non obsolètes tel que sha2.

3. Aucune donnée sensible ne circule dans le serveur en clair.

4. Utilisation de modules et bibliothèque à jour.

2.3.2 Gestion d'attaques :

`XSS :`

Les failles XSS ont été géré dans un premier temps par le choix de fonctions qui protège de ce type d'attaques tel qu'*innerText* au lieu d'*innerHTML*. Dans un second temps nous avons mis en place un système d'expressions régulières empêchant l'insertion de script en enmpêchant à titre d'exemple l'utilisation des chevrons et autres

caractères suspects.

`Injections Sql :`

L'utilisation de la base de donnée MongoDB de type NoSql empêche par construction les attaques de types injections Sql

`Injections NoSql :`

Tout input dans SealPass est contrôlé à l'aide d'expressions régulières qui emêche l'insertion de caractères spéciaux tel que $, ainsi l'utilisateur ne peut introduire des injections NoSql.

`CSRF :`

SealPass est construit en AJAX pour être mieux protégés contre CSRF, aussi certains points ont été mis en place pour éviter au mieux ce genre d'attaques tel que :

1. l'utilisation de POST pour les requêtes qui déclenchent des actions.
2. Contrôle systématique des headers
3. Faire expirer les sessions sous 5 minutes d'inactivité.
4. Ajouter d'informations reliées à la session dans les URLs.

`Vol de sessions :`

Les cessions de l'application sont basées sur les tokens qui expirent chaque 5 mintues d'inactivité, si le token est corrompu la signature change et ce dernier est rejeté. L'application demande une authentification avant de pouvoir exécuter toute actions, cette partie n'a pas encore été assez travailler. En effet pour l'instant les tokens sont générés il reste cependant la mise en place du système de garde qui vérifie le token à chaque nouvelle requête. Nous avons contacter monsieur Rotella qui nous a permis de travailler encore sur l'application avant la soutenance c'est pourquoi cette partie sera effective d'ici la soutenance.

2.3.3 Améliorations :

Mr Rotella nous a permis de travailler un peu plus sur le projet d'ici la soutenance c'est pourquoi nous allons essayer de mettre en

place le HTTPs et les certificats, et essayer de protéger SealPass contre d'autres attaques plus poussées.

3 Comparaison des objectifs

Nous allons à présent vous présenter rapidement le déroulement de notre projet, ainsi que les objectifs atteints ou non atteints. Enfin, nous parlerons des perspectives d'améliorations de notre application et critiquerons ce que nous aurions pu changer dans notre manière de travailler.

3.1 Déroulement du projet

Tout d'abord un mot sur le déroulement de notre projet, avant le début du confinement les rôles de notre projet changeaient de semaines en semaines, mais avec le confinement nous avons choisis de suivre une nouvelle approche.

Nous devrions toujours faire au minimum un point sur la situation globale une fois par semaine, nous avons mis en place un Trello pour savoir qui faisais quoi à quel moment, et ainsi mieux nous coordonner. Nous avons continué de communiquer sur WhatsApp comme nous le faisions auparavant. Et nous avons élu un "chef" de groupe permanent, Julien JACQUET, chargé de faire en sorte que le projet continue d'avancer malgré les nouvelles contraintes.

On peut dire que le projet a continué de progresser à bonne allure sur tous les fronts, à l'exception d'un point, la partie Cloud. À l'heure actuelle les parties front-end et back-end du Cloud sont fonctionnelles, mais le retard engendré suite au départ d'Abdoulaye BALDE ne nous a pas permis d'intégrer toutes les fonctionnalités du Cloud à temps.

Pendant nos conversations nous avons réévalué les objectifs que nous avions donnés dans le cahier des charges lors de la séance du 12.03.20. Le Cloud ne faisait plus partie des perspectives d'améliorations, mais des objectifs, pas forcément principaux, mais importants tout de même. La partie multilingue de l'application n'était plus un objectif, mais une amélioration possible, nous pensons que cet élément a peu de valeur technique et un intérêt limité dans le

contexte de ce cours.

Au cours de nos discussions, nous avons créé (Fabius KOSSI) des "wireframes" qui nous ont aidé à avoir une idée plus concrète de comment notre site serait organisé.

Tout au long de ce projet l'aspect sécurité de notre application est resté un point important, d'un côté parce que c'est l'un des objectifs de ce cours et de l'autre à cause de la nature sensible des informations que nous stockons.

Durant la période de confinement nous avons également choisi de scinder notre équipe en deux plus petites équipes, une équipe front-end (Fabius KOSSI, Julien JACQUET) et une équipe back-end (Ala Eddine CHAHI, Amir BEN MALLEM). Cette configuration nous a permis de fonctionner de manière plus autonome et est relativement similaire à la configuration retrouvée en entreprise.

Enfin, est venu le moment de l'intégration et de la rédaction du rapport, à ce moment les deux équipes ont été réunies.

Mais le travail ne s'arrête pas là, dans la semaine à venir, il va s'agir de préparer la présentation, de compléter la partie cloud, ainsi que tout autre problème subsistant.

3.2 Objectifs

3.2.1 Objectifs atteints

- Création d'utilisateurs.
- Connexion d'utilisateurs.
- Stockage de mots de passe.
- Modification de mots de passe stockés.
- Suppression de mots de passe stockés.
- Changement d'adresse mail associé a un utilisateur.
- Changement de mot de passe principal, avec demande de confirmation de mot de passe et d'ancien mot de passe.
- Génération de mot de passe en suivant les recommandations de l'ANSSI.
- Correctement sécurisé.

3.2.2 Objectifs non atteints

— Connexion du plugin à la base de données pour le rendre fonctionnel, le plugin est prêt mais non connecté.
— Intégrations du Cloud, par manque de temps.

3.2.3 Objectifs Supplémentaires

— Stockage de cartes bleues.
— Extension SealPass.

3.3 Améliorations

3.3.1 Perspectives d'améliorations d'ici la soutenance

— Améliorer et intégrer le Cloud complètement.
— Renforcer la sécurité contre des attaques plus coriaces.

3.3.2 Perspectives d'améliorations

— Support de plusieurs langues
— Synchronisation de n'importe quel service web. Lors de la connexion à un site qui n'est pas présent dans le coffre-fort de l'utilisateur, l'extension propose à l'utilisateur de sauvegarder ses identifiants.
— Application mobile.
— Edition en direct des fichiers dans le Cloud.
— Continuellement améliorer la sécurité de l'application.
— Ajouter plus de réglages personnalisés.

4 Conclusion

Le développement SealPass fut l'occasion pour tous les membres d'acquérir de nouvelles compétences sur le plan technique mais aussi sur le plan humain, sur le travail en équipe avec des personnes qu'on ne connaissait pas auparavant.
Dès la première réunion l'ensemble de l'équipe était d'accord pour dire que la réalisation de SealPass reposait sur deux axes l'organisation à travers une répartition équitable des tâches et une bonne communication, bien qu'un événement inattendu se soit produit (confinement) nous avons su réinventer notre organisation interne afin d'atteindre les objectifs que nous nous sommes fixés.

Le projet s'est avéré être un point indispensable dans notre apprentissage il nous a permis non seulement d'approfondir nos connaissances sur le développement d'applications web mais aussi de nous introduire aux domaines de sécurité de ces applications que nous connaissions que très peu.

Liens utiles : Git, Rapport LaTeX

5 Sources

Nombre de mots de passe dérobés source NIST
Recommendations de l'ANSSI sécurité des mots de passe
Plus d'infos sur Javascript
Plus d'infos sur VueJS
Plus d'info sur ReactJs
Plus d'infos sur AngularJs
js frameworks benchmark
Génération mot de passe aléatoire
NodeJs crypto methods
Web Cryptography API pour génération mdp
Choix du language ou framework en back-end
Documentation sur crypto-js
Benchmark de l'API WebCryptoAPI pour finaliser le choix
Etude des performances de differentes bibliothéque crypto
Etude de Stanford Javascript Crypto Library
Recherche informtation sur les differences entre nodeJs et php
Tendences actuelle des frameworks Js front-end
Lecture documentation de VueJs ainsi que compartif avec React
MSR JavaScript Cryptography Library
Comparatif framworks JS
Documentation sur php sodium
Comparatif React et VueJs selon critére des tendances actuelles
Libsodium en javascript
Php sodium
Multer pour le cloud
Plus d'infos sur AJAX
Plus d'infos sur express
Plus d'innfos sur pyhton Django
Comparatif NodeJs et Django
Plus d'infos sur les SGBDR
Plus d'infos sur les SGBD NoSql
Mongodb cloud
Plus d'infos sur Mysql
plus d'infos CSS
Git de DOMPurify
Infos sur les différentes attques
Alternative pour AJAX